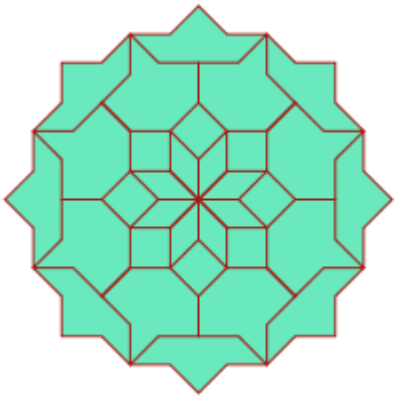
Visual Patterns

with Kojo



by
Lalit Pant and Anusha Pant

Version: September 16, 2017

License: Creative Commons *Attribution-NonCommercial-ShareAlike 4.0 International*
CC BY-NC-SA 4.0
© 2010–2017 Lalit Pant (lalit@kogics.net)
http://www.kogics.net

A Note for Facilitators and Teachers

This book contains a series of activities for kids to play with.

Most activities contain a fully defined program and a picture of the output of the program. For such activities, ask a kid to type in the program inside the script editor, run it, and then check that the actual output of the program matches the output shown in the book. Then, ask the the kid to do some reflection, i.e., think about and discuss what was just learned.

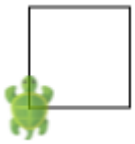
Some activities contain an incomplete program, with the incomplete areas marked with `???`, and a picture of the output of the (complete) program. For such activities, ask a kid to type in the program inside the script editor, fill out the incomplete portions of the program, run it, and then (as before) check that the actual output of the program matches the output shown in the book.

At every step, encourage the following:

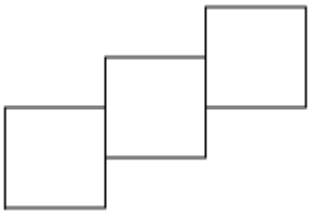
- exploration, discovery, and a sense of play.
- perseverance in the face of unexpected results, and joy in the process of figuring out what went wrong.
- reflection and discussion about what was learned.
- digressions and diversions from the provided sequence of activities.

It is not important to finish all the activities. But it is vitally important to spend time with, go deep into, enjoy, and learn from each activity!

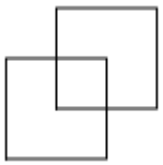
```
def square() {
    repeat(4) {
        forward(50)
        right()
    }
}
clear(); setSpeed(medium)
setPenThickness(1); setPenColor(black)
square()
```



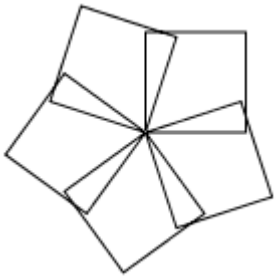
```
def square() { /* same as before */ }
def block() {
    square()
    right(90)
    hop(50)
    left(90)
    hop(25)
}
cleari(); setSpeed(medium)
setPenThickness(1); setPenColor(black)
repeat(3) {
    block()
}
```



```
def square() { /* same as before */ }
def block() {
    square()
    ???
}
cleari(); setSpeed(medium)
setPenThickness(1); setPenColor(black)
repeat(2) {
    block()
}
```



```
def square() { /* same as before */ }
def block() {
    square()
    ???
}
cleari(); setSpeed(medium)
setPenThickness(1); setPenColor(black)
repeat(5) {
    block()
}
```



Hands on with Forty figures

On the next two pages you will see forty different figures (these are Kojo drawings of the figures presented in a wonderful book by Barry Newell called Turtle Confusion). Some of these figures are building blocks for patterns, while others are the patterns themselves. Use the ideas learned while working through the previous page to make the patterns. Each program that you write to make a pattern should contain:

- the definition (via a `def`) of the fundamental **shape** inside the pattern. As you define this shape, make sure that you think about the starting position and direction (or heading) of the turtle within this shape.
- the definition (via a `def`) of the pattern building-**block**. This includes the fundamental shape, and some commands to set the position and direction of the turtle to make the next building block in the pattern. This position and direction relates to the starting position and direction of the fundamental shape.
- A **repeat** command that makes the pattern building-**block** a **certain number of times** to create the full pattern.

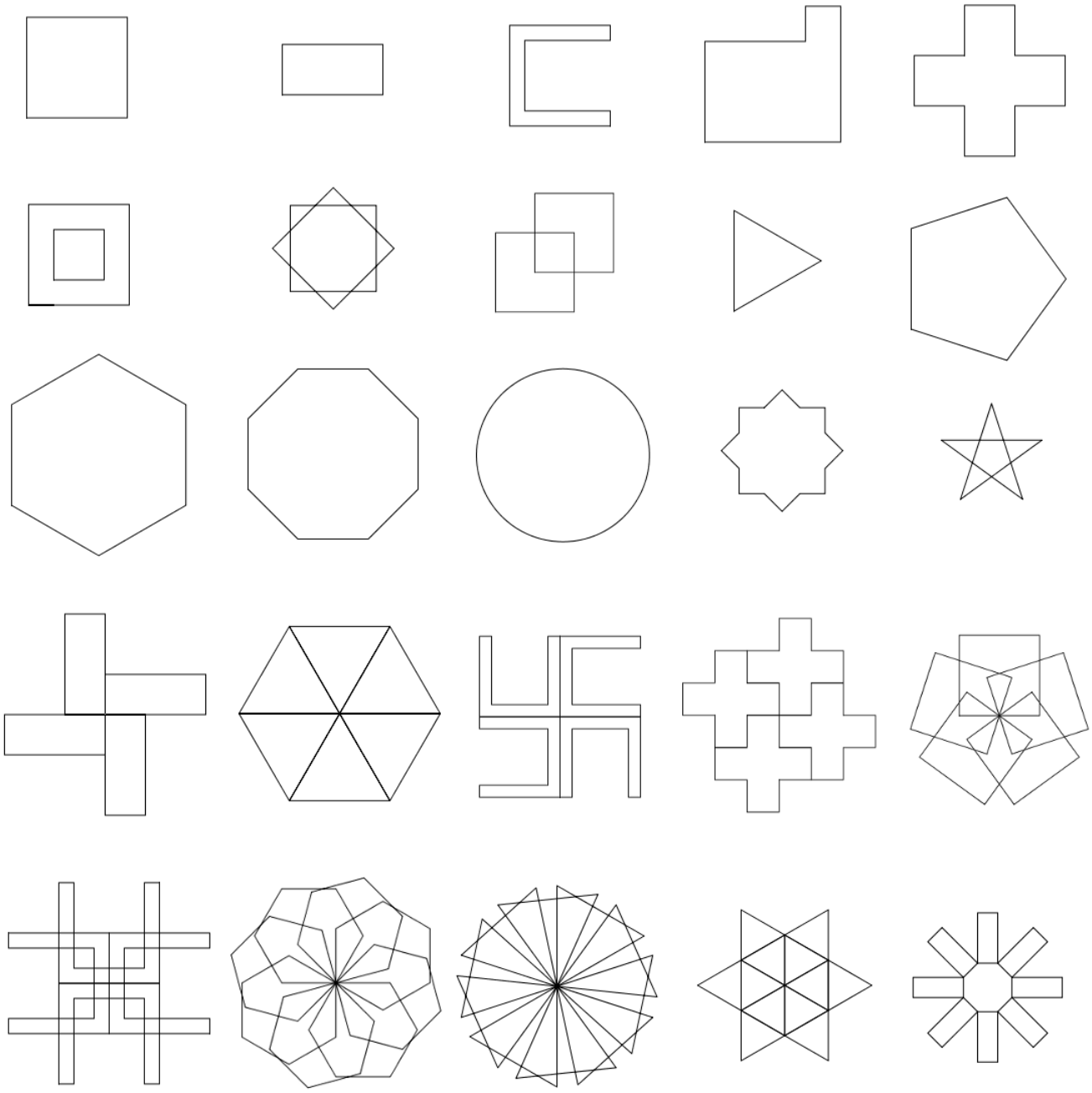
Here's a template for the kind of code you should write to make the figures:

```
// define the fundamental shape present inside the pattern
def shape() {
}

// define the building block of the pattern,
// based on the fundamental shape
def block() {
    shape()
    // position the turtle for the next block
}

cleari(); setSpeed(medium)
setPenThickness(1); setPenColor(black)
// change the turtle's direction if needed
// and then repeat the block to make the pattern
repeat(8) {
    block()
}
```

Figures 1 to 25



Figures 26 to 40