Last updated on 2013-04-25

Math on a Sphere

Topic 9: Variables

Sometimes we want to use variables in our code. By variables, we means terms like *x* and *y* that we learn about in algebra. We use these variables to represent numbers (and sometimes other things, like words). Variables are very useful in computer programming.

Variables can help us to (1) improve the readability of our code, (2) condense our code, or (3) tell the computer to follow different instructions depending on the state of some variables(s).

One simple example is the following:

```
repeat 5 {
      forward 10.40595729638936397
      right 47.29476739298124
      }
```

Reading this code is kind of confusing, because the long numbers are distracting. So by using variables we can make things more readable:

```
stepSize = 10.40595729638936397   // how far turtle moves each time
turnAngle= 47.29476739298124    // how far turtle turns each time
repeat 5 {
      forward stepSize
      right turnAngle
}
```

If our code was even longer, and we used these variables even more, they would be all the more helpful.

When we use variables, we want to give them useful names, or else things can get confusing. (And if things get confusing, we've just made our lives harder instead of easier!) Also, it can be helpful to use comments, like those shown above. To make comments we type "//" and then write notes for ourselves or other people working on our code. The turtle ignores any text after the "//".

The previous example showed how we could use variables to improve readability. We can also use variables to condense code. This may be confusing, so let's use an example to make things easier to understand. Let's say we want to make a spiral.

One way to make a spiral is as follows (see Figure 1):

```
right 90
forward 20
right 20
forward 20
right 21
```

```
forward 20
right 22
forward 20
right 23
forward 20
right 24
forward 20
right 25
forward 20
right 26
forward 20
right 27
forward 20
right 28
forward 20
right 29
forward 20
right 30
forward 20
right 31
forward 20
right 32
forward 18
right 33
forward 18
right 34
forward 18
right 35
forward 18
right 36
forward 16
right 37
forward 16
right 38
forward 16
right 39
forward 16
right 40
forward 16
right 41
forward 16
right 42
forward 14
right 43
forward 14
right 44
forward 14
right 45
forward 14
right 46
forward 14
right 47
forward 12
right 48
forward 12
right 49
forward 12
right 50
```

```
forward 10
right 51
forward 10
right 52
forward 10
right 53
forward 8
right 54
forward 8
right 55
forward 6
penup
forward 80
```

Figure 1. Making a spiral without using repeats

As you can see from Figure 1, this code works well. However, the code is really long and takes a while to write. We can make it shorter if we notice there's a pattern. Now, it's not a pattern like the one we talked about before, when we discussed the `repeat` command in Topic 5. The pattern here is that we repeat the same general instructions, but every time we change how far the turtle walks forward and how far the turtle turns.

Let's analyze the code to figure out the pattern. The first line (right 90) does not actually draw anything, it just rotates the turtle so that the spiral is drawn on the part of the sphere we're looking at. So ignoring that line and looking at the next 10 lines of code, we get:

```
forward 20
right 20
forward 20
right 21
forward 20
right 22
forward 20
```

```
right 23
forward 20
right 24
```

So in these 10 lines of code there are 5 pairs of instructions. Each pair has the following form:

```
forward #
right #
```

In each of the 5 pairs, the turtle always walks forward 20 steps. However, the amount the turtle turns to the right changes. The first time the turtle turns, it turns 20 degrees. After the first time, the turtle turns 1 degree further than the previous time. It is easy to express these instructions with a mathematical expression. Let's say we created a variable called `turnAngle`, and gave it an initial value of 20 degrees by saying `turnAngle = 20`. Now each time we move on to a new pair of instructions, we'd need to increase the value of `turnAngle` by 1 degree. We could do this by saying `turnAngle = turnAngle + 1`.

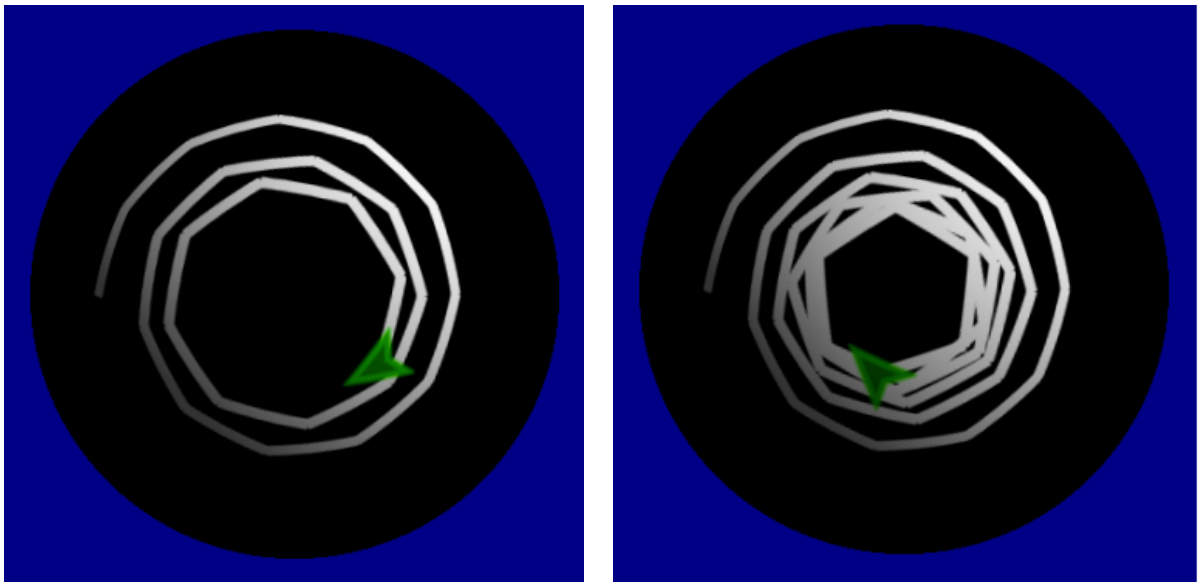
So that's a quick example of how we can represent things, like the angle we want the turtle to turn, with variables. But how do we change the value of `turnAngle`, so that the turtle turns a different amount each time? To accomplish this, we can use the `repeat` command we discussed in Topic 5. Here's how we can do this

```
turnAngle = 20
repeat 5 {
     forward 20
     right turnAngle
     turnAngle = turnAngle + 1
     }
```

In the first line we create our variable and give it its initial (starting) value of 20 degrees. Then we give the turtle the same 5 pairs of instructions of before. Except now, we also tell the computer to increase the value of `turnAngle` by 1 degree, every time it executes a pair of instructions.

The code above can generate a simple spiral. If we increase the number of repeats to 25, we get the results shown in Figure 2A. If we increase the number of repeats to 45, we get the results shown in Figure 2B.

Looking back at this code, we see that using variables allowed us to do two things. By using variables, we condensed the code. Using the `turnAngle` variable also let us instruct the turtle to do different things, depending on what part of the spiral the turtle is drawing. This is one example of why variables are so useful.



(A) (B)

Figure 2. We can make spirals using variables. (A) 25 repeats and (B) 45 repeats

If you look at Figure 2B careful, you see that if we increase the number of repeats to 45, the spiral starts touching itself. This is because as we move towards the inside of the spiral, the turtle is going forward too far each time. We can fix this by changing both the forward step length and the turn angle.

The following is an example of code that generates a better spiral (see Figure 3). The variable names are now different. We're using *x* for the amount the turtle walks forward. And we're using *a* for the amount the turtle turns to the right. Every time the turtle executes a pair of instructions, *x* is decreased by 0.33 degrees and *a* is increased by 0.5 degrees.

```
x = 20
a = 20

right 90

repeat 60 {
    forward x
    right a
    x = x - 0.33
    a = a + 0.5
}

penup
forward 50
```

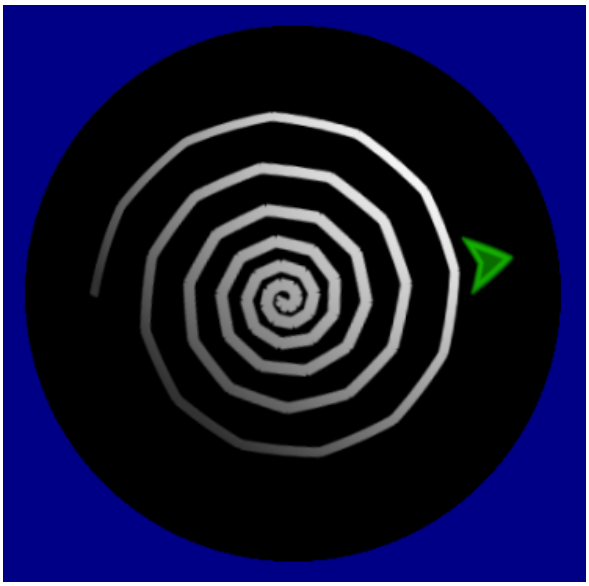


Figure 3. Another spiral produced by using variables

Variables are useful in so many different designs. What do you want to use them for?