Last updated on 2013-04-25

Math on a Sphere

Topic 5: Repeating Instructions

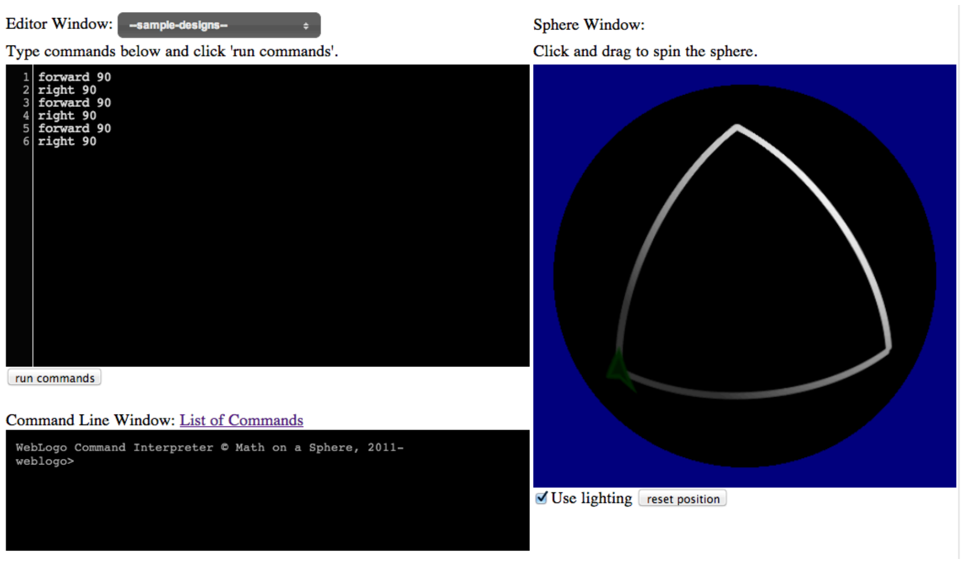
Writing many lines of code can become tiring and boring, so computer programmers often try to simplify their code. Condensing code in this way also makes it easier for others to read.

As an example, let's look at how we could create a triangle in a faster way.

One way to make a triangle is to use a series of `forward` and `right` commands, as shown below:

```
forward 90
right 90
forward 90
right 90
forward 90
right 90
```

As shown in Figure 1, this set of instructions produces a triangle. *(If this seems confusing, consult Topic 3: Lines and Shapes on the Sphere.)*


Editor Window:
--sample-designs--
Type commands below and click 'run commands'.
run commands
Command Line Window: List of Commands
Sphere Window:
Click and drag to spin the sphere.
Use lighting
reset position

```
forward 90
right 90
forward 90
right 90
forward 90
right 90
```

```
WebLogo Command Interpreter © Math on a Sphere, 2011-
weblogo>
```

Figure 1. One way to make a triangle

Now, let's try to figure out how we could simplify/condense the above code. There are different ways to simplify blocks of code. One way is to use the `repeat` command. To use the `repeat` command, we first need to identify the lines of code we want to repeat. Then, we figure out how many times we need to repeat it.

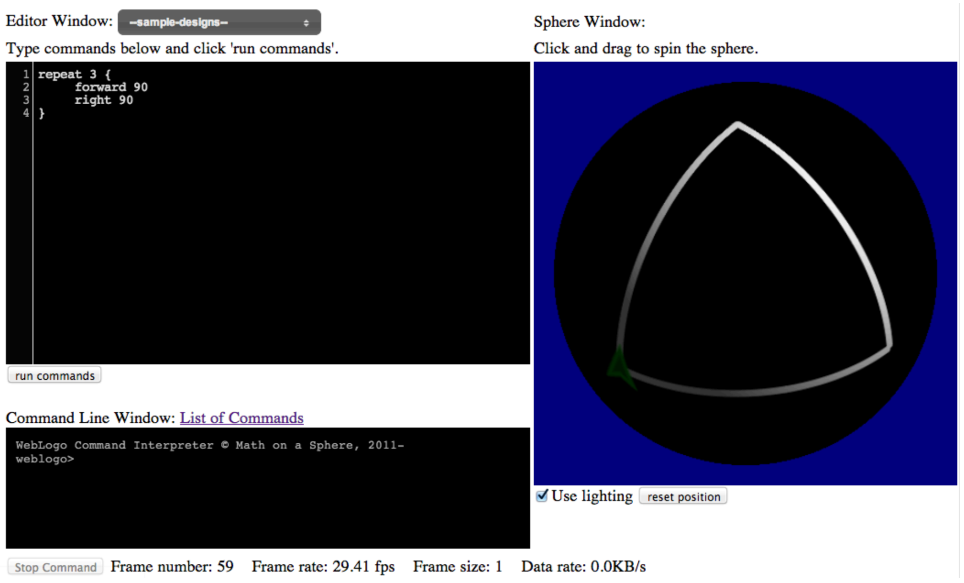
Look at the original code. What set of commands is being repeated?

```
forward 90
right 90
```

How many times do we repeat this pair of commands? 3 times. Now that we know what instructions we're repeating, and the number of times we're repeating the commands, we can implement the `repeat` command, as follows:

```
repeat 3 {
      forward 90
      right 90
}
```

As you can see, these instructions produce the same design as before (Figure 2).


Editor Window:
--sample-designs--
Type commands below and click 'run commands'.
run commands
Command Line Window: List of Commands
Sphere Window:
Click and drag to spin the sphere.
Use lighting
reset position
Stop Command

```
repeat 3 {
      forward 90
      right 90
}
```

```
WebLogo Command Interpreter © Math on a Sphere, 2011-
weblogo>
```

Frame number: 59 Frame rate: 29.41 fps Frame size: 1 Data rate: 0.0KB/s

Figure 2. A second way to make a turtle, which uses the `repeat` command

Now let's look at the syntax of the `repeat` command a little more closely. (The *syntax* refers to the way in which we write the commands – see http://en.wikipedia.org/wiki/Syntax_(programming_languages) for more information.)

```
repeat 3 {
     forward 90
     right 90
}
```

The first thing we write is "repeat". The `repeat` command tells the turtle that it is going to have to repeat a number of commands. The turtle doesn't know what these commands are yet or how many times she'll have to repeat them – she just knows that she's going to repeat the instructions. So, it's basically just notifying the turtle – "hey turtle, get ready, because you're about to have repeat the set of commands I'm going to give you".

After the word "repeat" comes the number 3, which tells the turtle that she'll have to repeat the instructions 3 times.

Next comes the left curly bracket "{". The left and right curly brackets sandwich the instructions that the turtle is going to repeat. They're necessary because we often combine the repeat command with other code. So, the turtle needs to know which instructions she's going to repeat. This is an example of such a situation:

```
repeat 3 {
     forward 90
     right 90
}
forward 50
```

If we didn't have the {} symbols, the turtle wouldn't know which lines of code to repeat.

So to review, we first tell the turtle which command we're giving it (`repeat`). Then, we tell the turtle how many times it's going to repeat the commands. Then, we use {} to sandwich the set of instructions the turtle is going to repeat.

A helpful aspect of the turtle language is that it doesn't care too much about the exact syntax of the `repeat` command. You must have the necessary information in the correct order, but it doesn't care too much about spaces or which lines things are on. For example, you could move the first curly bracket to a new line:

```
repeat 3
{
     forward 90
     right 90
}
```

Or, you could put everything on one line.

```
     repeat 3 { forward 90   right 90 }
```

However, this last example -- where everything is on the same line -- isn't recommended because it's harder to read, especially when you are repeating a bunch of commands.

Now you're an expert at using the `repeat` command. What fun designs can you make with your new knowledge?