Last updated on 2013-04-25

Math on a Sphere

Topic 10: If Statements

So far we have focused mainly on straightforward commands for the turtle. We ventured into more complex commands when we discussed the random command, but we have not dived into complicated syntax. Here, we will begin to introduce some of the more complicated code structures that computer programmers routinely use. In this topic we will discuss what *if statements* are, why we would want to use them, and how we implement them.

We can describe how *if statements* are used by considering the childhood game of "duck duck goose", which is like a complicated game of tag. In "duck duck goose", all the children -- except one -- sit in a circle. Let's call the standing child Sally. Sally's job is to walk around the circle and pat each of the other children on the head. Every time she pats a child on the head she says either "duck" or "goose". If she says duck nothing happens, and she continues on to the next child. If she chooses to say "goose" she starts to run quickly around the circle, and the child whose head she just tapped jumps up and tries to chase and catch her.

Notice that the explanation of this game used two statements of the following form: "If [a happens] then [b happens]". Computers can also understand this kind of language. Just as we can "program" a child to understand how to play this game, we can program a computer to understand the game.

So what commands do we need to give the computer?

> If Sally says "duck", then Sally keeps walking around the circle AND the rest of the children stay seated.
>
> If Sally says "goose", then Sally starts running AND the child Sally just touched stands up and starts running after Sally

When we send instructions to a computer it's very important that we follow the syntax that the computer expects. For example, we saw in Topic 6 that when we use the `setposition` command the syntax is as follows: `setposition [a,b]`. It's important that setposition comes first and that there's a space after `setposition`. Furthermore, it's important that we use square brackets around `a,b`. Similarly, when we use *if statements* we need to follow the proper syntax. To do this, we convert the English sentences above into computer language.

The syntax for *if statements* in Math on a Sphere is as follows:

```
if (Sally says "duck") {
     Sally keeps walking around the circle
     Rest of the children stay seated
     }

if (Sally says "goose")  {
     Sally starts running
     Child Sally just touched stands up and starts running
     after Sally
     }
```

As you can see, we are giving the computer the same instructions that the children receive. We just use computer syntax instead of English grammar. For the computer syntax, we put the thing we are wondering about in parentheses `()`. For this example, the thing we are wondering about is whether Sally says "duck". Then we put the instructions for each case in curly braces `{}`. The syntax by itself looks like this (where … indicates missing text.)

```
if (...) {
     ...
     }

if (...)  {
     ...
     }
```

Figure 1 shows how you can use *if statements* to make cool designs. In the code for Figure 1 (provided at the end of this document), you can see that we use if statements like the one below:

```
if (counter == 1)  {
    set color pink
    repeat 30 {  rt 12 fd 2 }
    }
```

The use of the double equal signs == is very important. It tells the computer that we are not trying to set the counter variable equal to one. Instead, we are trying to determine *if* the value of the counter variable is currently equal to one.

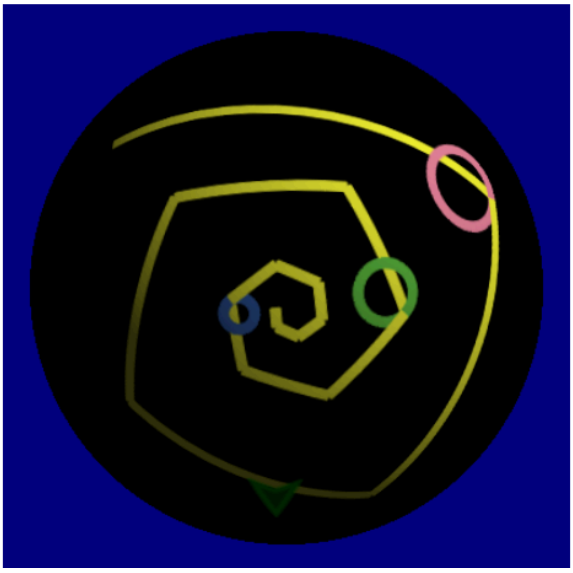

Figure 1. Using *if statements* in designs

If you have only two options, as above, you could alternatively use the following syntax:

```
if (...) {
      ...
      }
else (...)  {
      ...
      }
```

However, your designs will not come out as intended if you combine multiple *if statements* with an *else statement*, as shown below:

```
if (...) {
      ...
      }
if (...) {
      ...
      }
else (...)  {
      ...
      }
```

If you try to use the syntax shown above, the instructions for the `else`(...) case will override most of the instructions, except for the instructions immediately proceeding the else. An example of this is shown in Figure 2.

Note: many languages support `elseif` *statements. Math on a Sphere currently does not support* `elseif` *statements.*

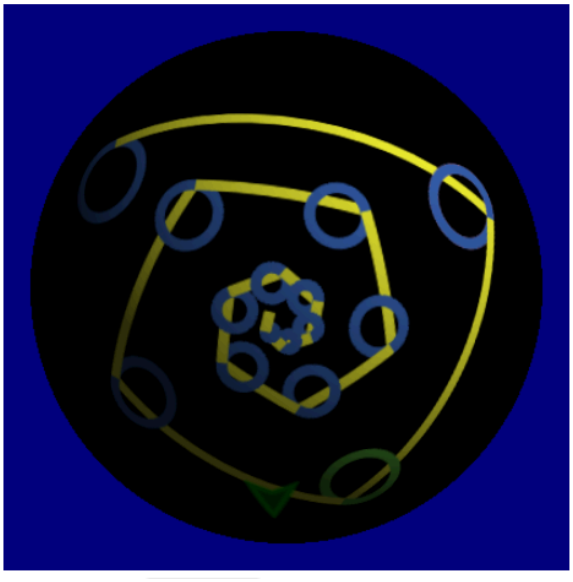

Figure 2. Be careful when using *if statements*. It doesn't work to combine an *else statement* with multiple *if statements*

In the previous examples, we were always checking whether something was equal to something else. An example: `if (counter == 1)`. But, we can also use other Boolean algebraic operations. For example, we could use `if (counter < 1)`.

Can you think of some ways to incorporate *if statements* into your designs?

Code for Figure 1:

```
x = 96
a = 60
counter = 0

penup
forward 20
left a
forward 30

setheading a

pendown
repeat 15 {

    if (counter == 1)  {
      set color pink
        repeat 30 {  rt 12 fd 2 }
        }

    if (counter == 6)  {
      set color green
        repeat 30 {  rt 12 fd 1.4 }
        }

    if (counter == 9)  {
      set color blue
        repeat 30 {  rt 12 fd 0.75 }
        }

    set color yellow
    forward x
    right a
    x = x * 0.8

    counter = counter + 1
}


penup
setheading 180
forward 50
```

Code for Figure 2

```
x = 96
a = 60
counter = 0

penup
forward 20
left a
forward 30

setheading a

pendown
repeat 15 {

    if (counter == 1)  {
      set color pink
        repeat 30 {  rt 12 fd (2 - counter/8) }
        }

    if (counter == 2)  {
      set color green
        repeat 30 {  rt 12 fd (2 - counter/8) }
        }

    else {
      set color blue
        repeat 30 {  rt 12 fd (2 - counter/8) }
        }

    set color yellow
    forward x
    right a
    x = x * 0.8

    counter = counter + 1
}


penup
setheading 180
forward 50
```