CMPS 10 Lecture Notes: Lecture 12 (2-11-2016)

SNAP! Review Day!

Loop Review
Let's start with creating a for loop and a repeat until, both at the same time, and try to count certain numbers.

- Let's say we have the numbers 6, 7, and 8, and we want to use a for loop and a repeat until to basically count them.
- Repeat until and for loop basically have the same functionality, though repeat until needs you to implement its own incriminator. That is, we have a change i by 1 block.
- So, in a for loop, we see that it goes from 1 to 10, so with the repeat until loop, we begin by setting i to 1. Then we have repeat until equals 11 (because with the for loop, you'll go inside the loop even when it is 10.
- And so as soon as i became 11, it stopped counting, so the repeat until block when from 1 to 10.
- And same thing with the for loop.
- Now let's add summation to this example.

For the summation, imagine that you have a bag. And you want to make sure that your bag is empty.

- We do that by first adding a script variable block, and adding a sum variable to it.
- We then use a set block at the beginning to set the value of sum to 0
- And then inside the loop, we can use change sum by i to actually handle the incrementing of sum.
- And then outside the for loop at the end, we want to say what the value of sum is.

One useful trick is to actually write out the steps that your program is going to do:

- So, sum begins at 0
- and we know that i will loop from 1 to 5.
- First, we say what i and sum is. So i is 1, and sum is 0.
- Then we change sum by i (so sum goes up by 1).
- Then we say what i is (it is still 1) and then we say what sum is (now it is 1)
- And that is the first run through the loop!
  - The second time through the loop: now i is 2 (it got incremented by 1), and sum is still 1, as it hasn't changed yet.
  - Then we change the sum by i (that is, we add 2 to the sum)
  - Then, when we print out i, at this point, it is still 2, but sum is going to be three now.
    - Then if we go one more round...
    - i goes up by 1, so now i is 3. Sum remains three the top of the loop.
    - Then we add i to sum, that is, we add three to three, so our sum becomes 6.
    - And when we print it out, we see that i is 3 and sum becomes 6.

Lets do a similar thing with multiplication.

- We make a new script variable, called multiply.
- We set multiply to 1 (because if we set it at 0, that will lead to problems down the line since multiplying things by 0 always yields 0).
- Then inside the for loop, we want to say something like set multiply to multiply X i.
- So you end up with 1 X 2 X 3 X 4 X 5, which ends up being 120.

And so at the end of the day, we just have a lot of different implementations of addition and sum, using repeat until and for loop.
Now we are going to move on to for loops, moves, and turns.

- Lets take a moment to look at how the change of the degree is going to give us different shapes.
- Also, note that there is a nice glide block, that will make the sprite move smoothly across the screen.
- We put our pen down

- We have a for loop with i = 1 to 50.
- Then we move four steps and we turn 180 degrees.
- We see that since we are turning 180 degrees it is just going to be a straight line.
- And if we replace move 4 steps with move l steps, then we get a line that gets bigger and bigger.
  - What if we changed the turn to 120 degrees instead of 180 degrees.
  - Well, as we may remember from earlier in class, turning 120 degrees helps us to make a triangle!
  - And then we get a really cool looking triangle.
  - What if we put in 240 degrees?
    - Ah, it does the same thing, but from a different angle! It is as if your triangle has been rotated a little.
  - What if we put in 121 degrees?
    - It is still a triangle, but it is as if it is moving. You are trying to draw a triangle, but it is incomplete!
    - And because you are moving the i inside your for loop, it gets bigger.

Remember how we made it so that the sprite would follow the mouse?

- If you do a forever loop, you can do some cool stuff with moving your sprite and turning some number of degrees. You can make a little starburst pattern.
  - This will be useful to know for the midterm!!!

BREAK

Broadcasting and multi sprite communication.

- From kaleidoscope example we learned we can have multiple sprites and they can function individually by us writing code for each of them.
- And each of them can have its own code.
- But it isn't clear how we can make these sprites interact with each other.
  - So one can be responsive to the other one's behavior.
  - So in order to make them interact with each other we will use a concept called broadcast.
  - One sprite will broadcast a message, and the other one will catch it and respond to it.
  - You can drag images in to change the color of your sprite.
  - So for your assignment you can download the duck and Anlonso sprites used in lecture.
  - Let's say you want to make the sprites have a conversation with each other.
  - There is a difference between broadcast and broadcast and wait the first is you broadcast and you don't care if anybody responds. The second is you don't do anything until someone else responds.
- Under control, there is a "when I receive" block which is how you behave when you hear a broadcast.
  - So, when you click start, duck will broadcast something (or vice versa).
  - And you GIVE a name to the broadcast message, and you need to make sure that you give the receive message that SAME name.
  - And then voila! you are doing great! they can now say hello to each other.
    - and to make more lines, you can broadcast with another block,
      - And then the duck gets another receive block. NOTE that the new receive block does not get attached to the existing block.
      - For homework: dialogue is NOT enough. Some amount of movement inside the screen is important.
      - Or you can make a different game, that involves conversation with the user. i.e. you ask for user input and then based on that it changes the flow of conversation.

You can make sprites respond to keyboard input with blocks that listen for it.

- So there is are blocks in the sensing panel that says key UP ARROW pressed.
- And then you can use the motion block 'change y by 3'
- And so you can change y by 3 if you push up arrow, and change y by 3 if you push down arrow, change x by 3 if you push left, and change x by 3 if you push right.

- AND we want the duck to ALWAYS be response to our movements, so we need to loop the whole thing in a forever loop.

What if we want our sprite to move like a ball?

- Lets do this to the sprite.
- Lets say as soon we hear the user is ready we want Alonso to start moving like a ball.
  - We'll bring in a move block into Alonso after the message 2 Alonso.
  - and we want him to bounce off the walls.
    - And there is block that already exists that we can use to achieve that: if on edge, bounce.
    - And we can point in direction based on which edge we bounce off of.

There was a bug that we were facing where both sprites had their own forever loops.

- Apparently the solution was to have the forever loops be inside of their own receivers.
- Seems like a hack, but okay!

How do you do collision detection? How do you do if sprite 1 touches touch 2?

- There are two different types of touch. One is touch of color, the other is touch of sprite.
- So you can make it so that if two sprites touch, then you do the pause all command, indicating that the game is over.
- OR you can make it so that if it just touches ANYTHING of a particular color, then the game ends.

So, you have two options:

- Design an interactive story with some actions. that is not everything is just dialog based.
- OR design a game!

Due Feb 22nd 8am.