Modeling a Girl

1 Introduction

This activity will involve some modeling. This means that we are going to take a real life situation and simplify it so that we can represent or simulate it on the computer. The first activity that you did with termites and wood chips was also a model. We are going to try to model something more complex though!

We will have one main character: a girl. She is studying math and English (probably among other subjects). She is going to move around your `Graphics` window and bump into various people that she knows, including friends, family, and teachers. We would like to model how her interactions with different people affect how much she enjoys math and English and how well she does at each subject.

Please think about the following questions and write some ideas down for each.

- Think of the subjects that you study in school. What are some of them? What are your favorite subjects? What do you like about these? What are your least favorite subjects? What do you dislike about these?

- Are there times when you don't enjoy your favorite subjects? Why? Are there times when you do enjoy your least favorite subjects? Why?

- What subjects do you think you are good at? What subjects do you think you are not as good at? How do these lists compare to the lists of subjects that you enjoy or dislike? What makes you feel like you are good at a subject? What makes you feel like you are bad at a subject?

- Are there times when you are confused by something in a subject that you are good at or feel like you really understand something in a subject that you are not as good at? Why?

Here are some of my answers. When I was in high school I liked math a lot. I also enjoyed science, particularly biology and physics. I liked English about as much as math. Social studies and Spanish were pretty good most of the time. Gym was okay depending on the unit we were doing. Health wasn't terribly exciting.

I enjoy subjects that I do well in, that my friends and family are interested in and value, and that I have received particular encouragement in. I also like subjects that challenge me somewhat. I do not enjoy so much subjects for which my knowledge of the subject is derided by my teachers or classmates and subjects that I am not as good at. I particularly dislike subjects where the classes that I have taken have been taught poorly or by teachers who didn't seem to care whether or not I was interested in the subject.

I do well at subjects that I have been encouraged in over long periods of time. I also do well in subjects that I enjoy, and subjects that are similar to or use other subjects that I do well at. I do not do as well in classes that are boring and taught poorly (unless they are not particularly challenging).

I have used parts of my responses to model a girl. She will begin with a certain level of enjoyment of each of math and English, as well as with a certain level of performance in each subject. For a simple model, she will be able to interact with family, friends, and teachers. Each person in her environment has varying levels of supportiveness in math and English, which will affect our girl's enjoyment of each subject and her performance in each subject.

2 Our Heroine

Let's begin writing a program. We would like to call our girl a girl, and not a turtle, so first we will create a breed called "girl". At the top of the `Procedures` window, type

```
breeds [girl]
```

We want to keep track of some information about the girl, so we need to create some turtle variable. You can do this by typing

```
turtles-own [enjoy-math perform-math enjoy-english perform-english]
```

also at the top of the `Procedures` window.

Now we want to create one girl, and give her some initial preferences about math and English. We will have to create a procedure to do this.

```
to setup
  ca
  create-girl 1
```

```
  setenjoy-math 50
  setperform-math 50
  setenjoy-english 50
  setperform-english 50
end
```

Change the initial values of the variables as you wish to create your own unique girl.

So that our model is easy to use, you can create a button in the `Interface` window that runs the `setup` procedure.

3 Other People

To experiment with the model, it will be handy to be able to create different people and to have an easy way to set how supportive they are of math and English. To do this, we will create some sliders. Click on the third button from the left in the `Interface` menu: this button should have a green object on it. As when you create a button, you will have to click and drag in the `Interface` window to tell StarLogo where to put the slider and how big to make it. Then a dialog box will pop up.

Let's start with the math supportiveness parameter (aka variable). Under `Observer Variable` in the slider dialog box type `math-supp`. Under `Min` type `-10`, under `Max` type `10`, and under `Current` type `0`. Now click on `OK`. This will automatically create a global variable `math-supp` which is given the value shown on the slider. We can set the value of `math-supp` to be anywhere from -10 to 10, and to start out the value is set at 0. Create a second slider with the same `Min`, `Max`, and `Current` values for the variable `english-supp`.

When we populate our girl's environment with people, we would like to be able to specify whether each person is a family member, a friend, or a teacher in addition to their supportiveness in both academic subjects. We can do this with a slider also. However, the sliders can only take integer values, so we will have to choose an integer to represent each category. Create a third slider, and call the variable `who?`. Set `Min` at `1`, `Max` at `3`, and `Current` at `1`. We will use a slider setting of 1 to specify that we want to create a family member, 2 to create a friend, and 3 for teachers.

Now that we have some parameters from the slider settings, we need a procedure to create a person. People in the girl's environment will be represented by patches in the `Graphics` window. So we can use the following procedure (type this in the `Procedures` window).

```
to create-person
  settemp-x ((random screen-size) - screen-edge)
  settemp-y ((random screen-size) - screen-edge)
  if (pc-at temp-x temp-y) - 0
   [make-color
    osetsupport-math-at temp-x temp-y math-supp
    osetsupport-english-at temp-x temp-y english-supp]
end
```

This procedure uses two as yet undeclared variables, `temp-x` and `temp-y`, to store the x and y coordinates for the new person. To declare these variables, type

```
globals [temp-x temp-y]
```

at the top of the `Procedures` window.

In this procedure we also want to store the current settings of the sliders in some variables associated with the patch representing the person we have just created. To declare a patch variable we return to the top of the `Procedures` window and type

```
patches-own [support-math support-english]
```

This declares the patch variables `support-math` and `support-english` that are used in the procedure `create-person`.

The procedure `create-person` also calls `make-color`, which we haven't yet defined. Each person-patch will be a particular color, depending upon if the person is family, friend, or teacher of our girl. We will need to know which color goes with each category. We can store this information in three global variables: `family-color`, `friend-color`, and `teacher-color`. To declare these variables, add them to the list of global variables at the top of the `Procedures` window.

```
globals [temp-x temp-y family-color friend-color teacher-color]
```

We can add some lines to the `setup` procedure to set these colors.

```
to setup
  ca
  setfamily-color red
  setfriend-color green
  setteacher-color blue
  create-girl 1
```

You are welcome to change these colors to ones that you prefer!

When we create a new person, we want the procedure `make-color` to change that patch to the correct color. We can do this as follows.

```
to make-color
  if who? = 1
   [osetpc-at temp-x temp-y family-color]
  if who? = 2
   [osetpc-at temp-x temp-y friend-color]
  if who? = 3
   [osetpc-at temp-x temp-y teacher-color]
end
```

That takes care of all the details: now we can create people for our girl to interact with! To make this easy, create a button in the `Interface` window that runs the procedure `create-person`.

4 How to Interact

We have a girl and we have an environment full of different people. Our next task is to tell the girl how to move around, and to tell StarLogo how to model her interactions with other people.

For our very simple model, we can have our girl move around her environment randomly. She can move forward, and every so often turn a random amount. Each time she moves, we would like to check if she has met a person (that is, if she is on a colored patch). The following procedure will move our girl and will call a procedure to check each patch that she moves to.

```
to go-girl
  lt (random 360)
  repeat 5
   [fd 1
```

```
    at-patch
    wait 0.1]
end
```

Here we pause after each move so that our girl doesn't whiz around the screen so fast that we don't know where she is.

The procedure to check a patch is as follows.

```
to at-patch
  ifelse (pc-at xcor ycor) = family-color
   [meet-family]
   [ifelse (pc-at xcor ycor) = friend-color
     [meet-friend]
     [if (pc-at xcor ycor) = teacher-color
       [meet-teacher]]]
end
```

I would like to digress a moment to compare the structure of `at-patch` with the structure of `make-color`. In both procedures we want to check which of three cases we are in. In `make-color`, since we have three separate `if` statements we check each case, even if we have already found one case to be true. This is good if multiple cases could be true at the same time. In `at-patch`, we check the first case: if this case is true we stop and do not check any more cases. So in `at-patch` we check a case only if the preceding cases were all false. This is good as long as only one case can be true at a time. In fact, this saves the computer a little time, since when either the first or second cases are true it doesn't have to check as many cases.

For only three cases on the fast computers that we have today, we will not be able to notice a difference. However, if we had a very large number of cases, checking them with separate `if` statements, like we do in `make-color`, would usually be noticeably slower than checking them with nested `ifelse` statements, like we do in `at-patch`.

Let's get back to the procedure `at-patch`. Depending on which case is true (whether the patch is a family, friend, or teacher person), we either call `meet-family`, `meet-friend`, or `meet-teacher`. These three procedures tell us how much and in what ways the interaction with a person affects our girl's enjoyment of and performance in math and English. These three procedures will look very similar, but will have some different numbers in them.

```
to meet-family
  setenjoy-math (enjoy-math + .5 * support-math)
  setperform-math (perform-math + .3 * support-math)
  setenjoy-english (enjoy-english + .5 * support-english)
  setperform-english (perform-english + .3 * support-english)
end

to meet-friend
  setenjoy-math (enjoy-math + .3 * support-math)
  setperform-math (perform-math + .1 * support-math)
  setenjoy-english (enjoy-english + .3 * support-english)
  setperform-english (perform-english + .1 * support-english)
end

to meet-teacher
  setenjoy-math (enjoy-math + .3 * support-math)
  setperform-math (perform-math + .5 * support-math)
  setenjoy-english (enjoy-english + .3 * support-english)
  setperform-english (perform-english + .5 * support-english)
end
```

Each procedure changes our girl's enjoyment and performance variables by an amount that depends on how supportive the person she has met is of math and of English, and on a factor that reflects the relative weight that our girl gives to the opinions of people from this category (family, friend, or teacher).

We can also create a button in the `Interface` window to perform the simulation. This button will call the procedure `go-girl`. For this button, when creating and naming the button, click on the box marked `Forever` in the button dialog box. Now the procedure `go-girl` will run repeatedly until you click on the button a second time to make it stop running.

5 Where Are Our Results?

Now the program runs a simulation, but we are still missing one thing: how do we find out the results of our model? Remember that we wanted to know the effects of interactions with other people on our girl's enjoyment of and performance in both math and English. So we would like to know the values of the variables `enjoy-math`, `perform-math`, `enjoy-english`, and `perform-english` after the program has been running for a little while.

To keep track of these variables, we can create monitors in the `Interface` window. Click on the third button from the right in the `Interface` menu to create a monitor. When you reach the monitor dialog box, under `Instruction` type `enjoy-math-of 0` (since our girl has ID number 0), and select 1 decimal place. Create additional monitors for `perform-math` (enter the instruction `perform-math-of 0`), `enjoy-english`, and `perform-english`. The monitors may not display the correct values immediately: to fix this, restart the simulation by clicking on your setup button, re-populating the girl's environment, and clicking on your button to run the `go-girl` procedure.

6 The Complete Code

```
globals [temp-x temp-y family-color friend-color teacher-color]
patches-own [support-math support-english]
turtles-own [enjoy-math perform-math enjoy-english perform-english]
breeds [girl]

to setup
  ca
  setfamily-color red
  setfriend-color green
  setteacher-color blue
  create-girl 1
  setenjoy-math 50
  setperform-math 50
  setenjoy-english 50
  setperform-english 50
end

to create-person
  settemp-x ((random screen-size) - screen-edge)
  settemp-y ((random screen-size) - screen-edge)
  if (pc-at temp-x temp-y) = 0
   [make-color
    osetsupport-math-at temp-x temp-y math-supp
```

```
    osetsupport-english-at temp-x temp-y english-supp]
end

to make-color
  if who? = 1
   [osetpc-at temp-x temp-y family-color]
  if who? = 2
   [osetpc-at temp-x temp-y friend-color]
  if who? = 3
   [osetpc-at temp-x temp-y teacher-color]
end

to go-girl
  lt (random 360)
  repeat 5
   [fd 1
    at-patch
    wait 0.1]
end

to at-patch
  ifelse (pc-at xcor ycor) = family-color
   [meet-family]
   [ifelse (pc-at xcor ycor) = friend-color
     [meet-friend]
     [if (pc-at xcor ycor) = teacher-color
       [meet-teacher]]]
end

to meet-family
  setenjoy-math (enjoy-math + .5 * support-math)
  setperform-math (perform-math + .3 * support-math)
  setenjoy-english (enjoy-english + .5 * support-english)
  setperform-english (perform-english + .3 * support-english)
end

to meet-friend
  setenjoy-math (enjoy-math + .3 * support-math)
```

```
  setperform-math (perform-math + .1 * support-math)
  setenjoy-english (enjoy-english + .3 * support-english)
  setperform-english (perform-english + .1 * support-english)
end

to meet-teacher
  setenjoy-math (enjoy-math + .3 * support-math)
  setperform-math (perform-math + .5 * support-math)
  setenjoy-english (enjoy-english + .3 * support-english)
  setperform-english (perform-english + .5 * support-english)
end
```

7 Improving the Model

This is a very simple model. How accurate to you think it is? What would you change about the model? What would you leave the same? Here are some ideas for extending the model.

- We don't move around randomly. Instead, we seek out some people and avoid others, based on how they treat us. How would you make the girl do this? Who would she seek out? Who would she avoid?
- We have personal traits that also affect how much influence our interactions with other people have on our enjoyment of our classes and how well we do in our classes. These traits could include how stubborn and persistent we are, or how independent we are or how much we desire acceptance from our peers. What other such traits can you think of? How would you program the girl to have these traits?
- I mentioned earlier that how much I enjoy a subject affects how well I do at a subject, and vice versa. Should this be true of our girl also? How would you program this?
- Many people that we meet do not fall into the categories of family, friends, and teachers. How would you add more categories of people to the model? Who would you add? Likewise, additional subjects of study could be added.

- Some of your friends and acquaintances are also girls, and could be modeled by our program. How would you add more girls to the model? How would they interact with each other?
- These different ideas could also be combined. For example, suppose that we had multiple girls who each tried to hang out with supportive people and avoid unsupportive people. If one girl found a cluster of supportive people she might begin to draw a line behind her, leaving a trail so that other girls could follow and also enjoy the benefits of a supportive group of people, thereby helping other girls when she is in a situation where she can do so. This would be a different sort of interaction than the interactions between girls and non-girl people in their environment. How would you program this?

If you have time, try one of these suggestions, or an idea of your own for extending the model. Plan your change on paper ahead of time, and write out what procedures and variables you will need. This will make the programming much easier. Use the reference manual to look up any new commands that you need. If you can't figure out how to do something, try to brainstorm ideas with other girls. Most importantly, have fun!