PLTW COMPUTER SCIENCE

Activity 1.1.5

Variable Roles I

Introduction

Computers often store information while running a program. Some of this information is stored on disk to be used after the program is finished, like data files or a Microsoft® Word® document. Some information is not stored on disk and probably never will be — such information is to be used while the program is still running. These are variables.

The information in a variable can be simple, like a single number. The information in a variable can also be complicated, like a multidimensional grid of values.

The reasons for using variables can also be simple or complicated. A handful of simple reasons, however, address most of the reasons we use variables.

Think about notes you've made to yourself to help you remember something. Are there a few notes that come up over and over again?

Materials

- Computer with microphone and camera (built-in or external)
- Scratch™ 2 Offline Editor with the resource 1.1.5 sourceFiiles
  - or -
  Scratch account and project links as follows:
  FixedMostRecent: **http://scratch.mit.edu/projects/22630182/**
  Accumulator: **http://scratch.mit.edu/projects/22638577/**
  Aggregator: **http://scratch.mit.edu/projects/22638600/**
  AccumulatorAggregator: **http://scratch.mit.edu/projects/22638599/**
  RolesChallenge **http://scratch.mit.edu/projects/22629898/**

Resources

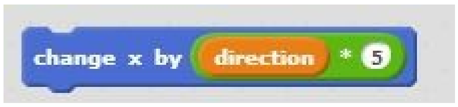
1.1.5 sourceFiles.zip

Procedure

Part I: “Fixed Value” and “Most Recent” Roles

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

1. Form pairs as directed by your teacher. Meet or greet each other to practice professional skills.
2. Open the Scratch project `1.1.5.a.A FixedMostRecent.sb2` in the Scratch 2 Offline Editor or in a browser with http://scratch.mit.edu/projects/22630182/
3. Familiarize yourself with the project.
    - What are the names of the two sprites?
    - How many scripts do the stage and each sprite have?
4. Run the program by clicking the green flag. Discuss with your partner why the script causes the behavior you observe. Record pseudocode here.
5. Explain why a programmer would want to use a variable like `MOVE_INCREMENT` instead of just putting `5` in the relevant code:



```
change x by (direction * 5)
```

6. Explain what the variable `direction` does. What does the value of `1` mean? What does the value of `-1` mean?
7. Observe the following features of the Scratch GUI.
    - When “seeing inside” a project, clicking on sprites while moving the mouse will be interpreted as a drag, which will move the sprite instead. Sprites will move when the mouse is dragged on the small stage but not when the mouse is dragged on the full screen stage.
    - Double clicking on a monitor changes its appearance. When a monitor appears as a slide, you can change the value of the variable even while the program is running.
8. On the stage click the `direction` slider. What happens if the `direction` is `0`?

9. The variable `direction` is playing the role of "most recent". This means that the most recent value is somewhat unpredictable, especially if it is affected by user input. All variables store the value most recently assigned to them, but sometimes the phrase "most recent" is the best explanation for *why* we are using a variable. Explain why this variable's role is best described as the "most recent" role.

10. Change the program so that the cat moves faster. Strategize together with your partner and then code and test in small iterations. Save your finished version as `1.1.5 yourNames_step10` or as directed by your teacher.

11. Move the wall sprite closer to the left side of the screen. Change the program so that the cat moves back and forth between the wall and the right side of the screen. Strategize together with your partner and then code and test in small iterations. Save your finished version as `1.1.5 yourNames_step 11` or as directed by your teacher.

Part II: "Accumulator" and "Aggregator" roles

12. Open the Scratch project `1.1.5.b.A Accumulator.sb2` in the Scratch 2 Offline Editor or in a browser with **http://scratch.mit.edu/projects/22638577/**

13. This program should look similar to the last one that you worked with. Take a moment to look for any differences. What are the new variables in this program?

14. Run the program by clicking the green flag. Try clicking the cat a few times. Discuss with your partner what you think any new variables introduced in this version of the code might be doing. Record your thoughts here.

15. What is the purpose of the variable `accumulator` in this code?

16. Open the Scratch project `1.1.5.c.A Aggregator.sb2` in the Scratch 2 Offline Editor or in a browser with **http://scratch.mit.edu/projects/22638600/**

17. The `aggregator` list also shows up in an on-stage monitor. As you click on the cat, what does its purpose appear to be?

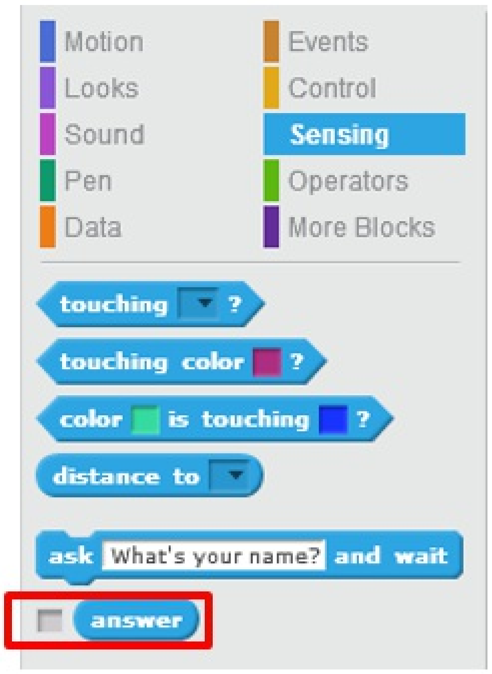
Part III: "Accumulator" and "Aggregator" practice

You will create a practice program that asks the user how much money they are gaining. You will be given a program that asks the question five times. You will modify the program to accumulate a running total and to keep a list aggregating all values.

18. Open the project `1.1.5.d AccumulatorAggregator.sb2` in the Scratch 2 Offline Editor or in a browser with **http://scratch.mit.edu/projects/22638599/**.

19. Complete the program by adding appropriate code into the **repeat** loop to store the user-

entered values in both `total` and `inPocket`. This will require the use of the **answer** shown below.

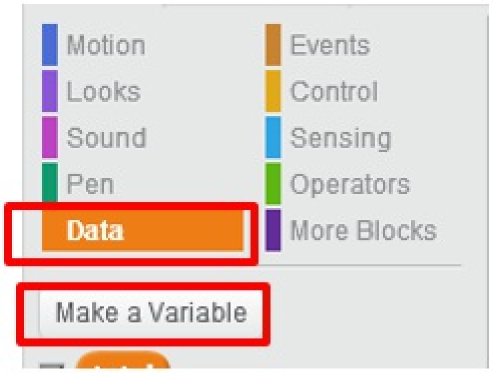


Part IV: "Accumulator" and "Aggregator" practice

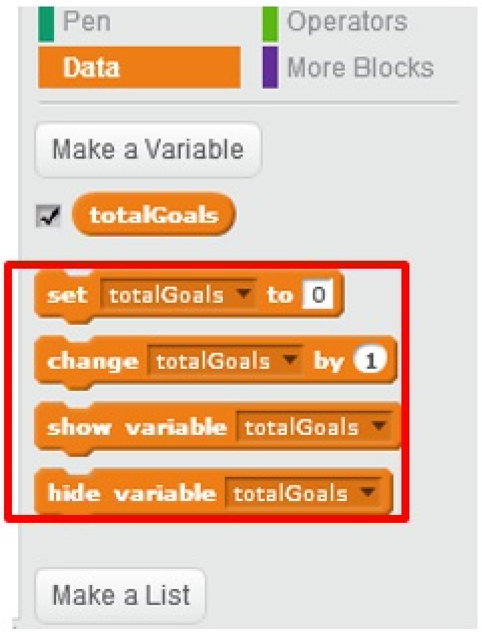
For additional practice you will create a program that scores the user's mouse clicks on a soccer ball sprite. You will be given a program that moves the sprite. You will modify the program to score the clicks on the ball and to keep a list of the times at which the clicks (goals) were made.

20. Open the Scratch project `1.1.5.e.A RolesChallenge.sb2` in the Scratch 2 Offline Editor or in a browser with http://scratch.mit.edu/projects/22629898/.

21. Create a new **accumulator** variable called `totalGoals` by clicking the **Make a Variable** button in the **Data** category as shown below.



Once you have done this, you will see several new blocks as shown below to use for that variable:

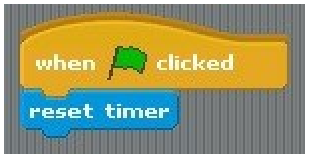


22. Modify the program so that when the sprite is clicked, the `totalGoals` changes by `1`.

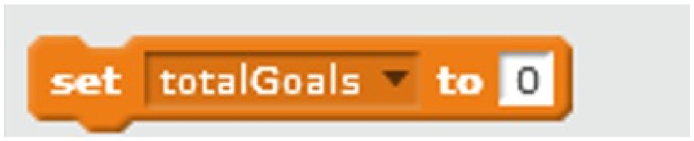
23. Now create a list using the **Make a List** button. Name it `timesOfGoals`.

24. You will see several new blocks that you can use to modify the contents of the list. Modify your program using one of the blocks so that when the ball is clicked, the time at which the click occurred will be added to the list (*Hint: the "thing" to add to the list is the* `timer` *in the **sensing** palette*).

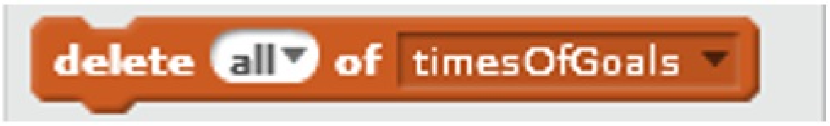
25. When you use variables, it is important to make sure that they **initialize** at the beginning of your code. To initialize a variable, set it to a value at the beginning of your program. The stack of blocks below shows how the `timer` is initialized.


```
when green flag clicked
reset timer
```

Add the code to initialize `totalGoals` and `timesOfGoals`. The variable `totalGoals` should start as 0, and `timeOfGoals` should start out empty. The blocks that perform these functions are shown below.



```
set totalGoals to 0
```



```
delete all of timesOfGoals
```

26. Test your code to see if it collects the times at which the ball is clicked. If it doesn't work the way you wanted, strategize, code, and test in additional iterations.

Conclusion

Remember that variable roles refer to why a variable is being used. In this activity we have examined four of the eight most common variable roles: fixed value, most recent, accumulator, and aggregator.

1. What do you see as the primary advantage of using a **fixed value** variable?
2. Write a pseudocode segment that shows usage of a **most recent** variable that is significantly different than the example given in this activity.
3. Explain the **accumulator** role:
    - List two examples of software you have used that you suspect might have used an accumulator.
    - Choose one of your examples from part (a) and explain what you think the accumulator in the program might have been doing and why this function leads you to classify the object as an accumulator.
4. Explain the difference between the accumulator role and the **aggregator** role.