Review of Scratch Concepts

- instructions (blocks) are composed into scripts
- the order in which instructions are executed is determined by
  - control blocks
  - principle of sequential execution
- variables can ease data management and describe actions on data

Clicker Exercise: Order of Execution

- in what order are the labeled blocks executed?
  A. X, Y, Z
  B. X, Z, Y
  C. Z, X, Y

Clicker Exercise: Using Variables

- if dist is set to 50 in the Stage script, where is the triangle after the "start" script is executed?
  A. (x= 0 , y=50 )
  B. (x= 50, y=0 )
  C. (x=-50, y=0 )
  D. (x= 0, y=-50)

Clicker Exercise: Model and Predict

- how much does the triangle turn on the second iteration of the repeat loop?
  A. 0 degrees
  B. 5 degrees
  C. 10 degrees
  D. none of the above
**Clicker Exercise: Model and Predict**

- how many turns does the triangle make in total?
  
  A. 7 turns  
  B. 8 turns  
  C. 40 turns  
  D. 41 turns

**Exercise: Extending the Scripts**

- what code changes are needed so that when the green flag is clicked, the South triangle starts in its “home base” and then does all three of the moves?

**Exercise: Extending the Scripts**

- if we duplicate the South triangle and rename as the North triangle, what changes do we need to make so that North is positioned and moves as we want?
- when all four triangles (North, South, East, West) are in place, what would change if we detached the “grow” hat block from the rest of the script from the South triangle?

**Properties of Variables in Scratch**

- the “dist” variable is *shared* among all triangles and the stage
- the “my-y-position” variables of South and North are *private*
Clicker Exercise: Model and Predict

• If “dist” = 110 and in South we change

   [Image: set my-y-position \(\uparrow\) to 0 - dist]

   to

   [Image: set my-y-position \(\uparrow\) to 50]

   then the home base of North also changes

   A. True
   B. False

Exercise: Extending the Scripts

• challenge: change the Stage script so that the triangles dance forever (i.e., until the red stop sign is clicked), picking from the three available moves at random at each dance step

• suggestion: use the “forever”, “if”, “if/else” control blocks and the “pick a random” operator

Scratch: Concluding Remarks

• you can find many project examples on the Scratch website (scratch.mit.edu/)

• Scratch provides support for programming of text as well as images and motion (e.g. for simple data analysis of biological data)

• if you like programming, consider a Scratch class project!

Quiz 2: What to Expect

• covers material in Module 2 notes, Don Norman article, Scratch lab, lectures up to today with the following exceptions:
  • exclude material on the Unix sort command and material on or after the slide starting with "expressiveness is provided through..."
  • however, Unix directory pathnames, the ls, cp, and cd commands, and the proper use of the space character are within the scope of the quiz
  • The material excluded from the quiz may be tested in the midterm or final!
Quiz 2: What to Expect

- questions on quizzes (and midterm and final exams) will derive from course learning goals!
- Material in the guest lecture (on networking) may be covered in the quiz (but no more than one question) as well as in the midterm and final exam.

Review of Scratch Concepts

- **instructions** are represented as blocks
- **variables**
  - have a value; can be private or shared
  - “set [...]” instructions can change a variable’s value
- **flow of control constructs** determine the order in which instructions are executed, and include:
  - **loops**, e.g. repeat blocks
  - **conditionals**: if and if/else blocks
  - **events**, e.g., “broadcast [...] and wait”, “when [flag] clicked”, and “when I receive [...]” blocks
  - principle of sequential execution

Quiz 2: What to Expect

*learning goal for Scratch*

- **learning goal**: recognize the use of and predict the behaviour of instructions, variables and flow of control constructs in short Scratch programs (this is a preview of one of the learning goals for the Processes module, to be presented in the next class)
- **sample question**: see recent clicker questions!

Quiz 2: What to Expect

*recall four learning goals for Interfaces*

- **learning goal 1**: explain how tools augment and constrain our power to think and act, define the “myth of human error” and give examples that dispel this myth
- **sample question**: “Tools that enhance our ability to explore data at different scales augment our power to think.”
  Give an example of a digital tool that supports or refutes this claim and justify your choice. Describe also how your chosen tool could further support exploration of data at multiple scales.
Quiz 2: What to Expect

- **learning goal 2**: explain strengths and weaknesses of human-computer interfaces, referring to concepts such as familiarity and consistency, mappings and metaphors, feedback, negative transfer, or additional concepts that you identify

- **sample question**: list two principles of good graphical interface design from the Module 2 notes. For one of the principles that you list, describe a concrete example of how the Scratch user interface succeeds or fails in adhering to the principle.

Quiz 2: What to Expect

- **learning goal 3**: navigate unfamiliar interfaces through effective use of conceptual models...

- **sample question**: Unix provides the following command (similar to the `cp` command) for moving a file:

  ```
  mv file1 file2
  ```

  One model for this command, when file2 already exists, is that file1 replaces file2; another model is that the contents of file1 are appended to file2. Describe an experiment to test which of these models is implemented in Unix.

Quiz 2: What to Expect

- **learning goal 4**: use basic features of Unix [...] with knowledge of the ways that special symbols are interpreted

- **sample question**: suppose that a Unix user’s current directory within the illustrated directory structure is `dir2`. Circle the current directory after the command:

  ```
  cd ~/dir1
  ```

```
    user  (home directory)
   /|
  dir1 dir2
 /|
|   |   
|   |   
|   |   
| dir1 dir3
```

```