SCRATCH MATH GAME

OVERVIEW

Subject: Math
Grades: 4-5 (can be adapted for older students as well)

Time: 2-3 hours, depending on Scratch experience level

Understandings:

- Students will know how to use conditional statements (if/then/else) to change the flow of a program
- Students will know how to use variables to keep track of values in a program

Objectives:

- Students will be able to code a game in Scratch that quizzes users about math facts

Standards:

- MA DLCS 3-5.CT.d.1 Individually and collaboratively create, test, and modify a program in a graphical environment
- MA DLCS 3-5.CT.d.4 Recognize that programs need known starting values (e.g., set initial score to zero in a game).

Implementation

Prior knowledge

I created this lesson for 5th grade students who completed my Choose Your Own Adventure in Scratch project in the previous school year. The math game project is pretty beginner-friendly, but it is written with the assumption that students have explored Scratch before and are familiar with the concept of conditional statements (if/then/else). However, this project can still be adapted for beginner students who are not familiar with Scratch using the tips below.

Scratch exploration

This project lends itself to Scratch exploration and therefore does not require a lot of prior experience with the Scratch interface. However, if your students are new to Scratch and time permits, it can be helpful to give them an opportunity to explore the platform. Scratch also has built-in tutorials for students to explore.

Conditionals

If your students do not have prior experience with conditionals, it may be useful to introduce this concept first without variables. Even if you do not plan to do the Choose Your Own Adventure project, check out these slides for an introduction to conditionals in Scratch. Give students time to create a game that asks a few questions and tells the user whether they are right or wrong. Once they are comfortable with this, go back and introduce variables and have students update their code to also keep track of the score.

Introduction and Coding Concepts (45 minutes)

These slides contain visuals and text for this part of the lesson.

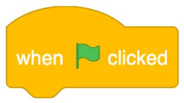
Preview game (5 minutes)

- Demonstrate the sample game (slide 3).
  - What happens when the player answers correctly?
    - Sound effect
    - Score goes up
    - Wizard congratulates player
  - What happens when the player answers incorrectly
    - Wizard says "WRONG"
  - What happens at the end of the game?
    - If the player got all of the questions right, the wizard tells them they won
    - Wizard gives instructions to play again
- Let students know that they can customize the game
  - Custom story, characters, and setting
  - Do you want to award gold coins instead of points?
  - Do you want to ask a bonus question if the score is high enough?
  - Do you want to create a custom sprite to use as a prize for the winner?
  - Endless possibilities!

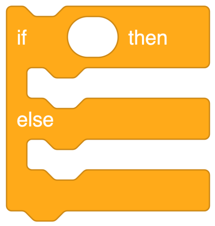
Scratch and conditionals review (5 minutes)

This portion of the lesson assumes students have prior experience with Scratch and conditionals. If not, see Scratch Exploration and Conditionals above for ways to expand on these areas.

- Questions for students:
  - What do you know/remember about Scratch? (slide 4)
  - What block do you (almost) always start your code with? (slide 5)



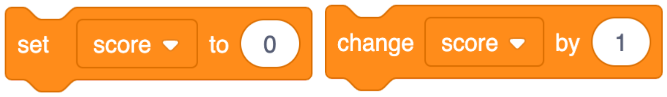
  - In programming, what is a *conditional statement*? (slide 8)



    Checks if a *condition* is true. If the condition is true, runs the code in the "then" section. Otherwise, runs the code in the "else" section.
  - What were some places *conditional statements* could have been used in the example game?
    - Checking if answer was right
    - Checking if player won

Variables (15 minutes)

- Ask students if they've heard the word variable before and know/have guesses for what it means (slide 11)
- **Variable**: a container that holds a value (information) (slide 12)
- *Optional: practice with variables (slides 13-22)*
  I found that using all of these examples front-loaded the information in the lesson too much. However, I've left slides in if you'd like to go more in-depth or pick different examples to show students.
- If you're making a game, what could you use a variable to keep track of? (slide 23)
  - Not limited to just the math game - what are examples from other games? Video games or real life
  - Example ides: score, number of lives, count of items in inventory
- Show "set" and "change" blocks for variables in Scratch (slide 24)



  - Draw a box on a whiteboard labeled "score". Ask students what happens when you run "set score to 0"
    - the value of score is 0 - write 0 in the box
  - What happens when you run "change score by 1"?
    - the value of score goes up by 1 - write 1 in the box
  - Give students more practice with predicting the value of the variable with different values inside of "set to" and "change by"
  - What do you do if you want the score to get lower?
    - Change by a negative number
- What is the difference between "set to __" and "change by __"? (slide 25)
  - "Set to _" does not care what the value of the variable is - it just set the variable to be the same as the specified value
  - "Change by _" does care what the value of the variable is - it uses that as a starting point and changes from there

Code planning (5 minutes)

- Ask students: what is the "flow" of the game? (slide 27)
  - We don't need to think about the code yet - walk through the major parts of the game in human terms
    - Introduce game (explain premise and rules)
    - Ask math questions, keeping track of score
    - Tell the player if they won or lost
- Pass out code planning sheet, one per student
  - Planning sheet works as a checklist/starting point for students to follow when they start in Scratch
  - There are spaces for students to fill in missing information. Work through each section as a class before students begin using Scratch
    - Choose 1 backdrop and 1 sprite (slide 28)
      - I like to give a time limit for this in the beginning to prevent students from spending the entire time customizing sprites. Once their code is done and they have at least 5 questions, they can go back and customize sprites and backdrops.

- Make a variable (slide 29)
    - Remind students that they will need to make a score variable before they are able to set/change the score
    - They can name this something other than "score" if it would work better with their game
        - Gold coins, goals, peanuts, etc.
- Start your code (slide 30)
    - What do you need to set the score variable to at the beginning of the game? 0 (students should fill this in)
    - Remind students that they can use more "say" blocks than there are on the planning sheet when introducing the game
- Ask the player questions (slide 31)
    - Ask students what should go into the "ask" and "if" blocks (math question and correct answer). They should label this on their code planners
    - Ask students what the "then" and "else" sections represent in the game (player got the question right or wrong). They should label this on their code planners
    - This is a good opportunity to differentiate for more advanced students to explore other blocks like animations, sound effects, additional sprites, etc. Students who are newer or less comfortable with Scratch can stick with just "say" blocks
    - Students will need to repeat the blocks in this section of the planner for each question they want to ask
        - If anyone gets annoyed about adding the same blocks over and over, this could be a good time to introduce making blocks in Scratch. I only did this on a case-by-case basis if it came up for individual students because this project already has a lot of new information to digest.
- Ending the game (slide 32)
    - Ask students what the condition should be to check if the player won (check if the score is equal to the number of questions). They should label this in their code planners.
    - Ask students what the "then" and "else" sections represent in the game (player won or lost). They should label this in their code planners.
    - This is another opportunity for advanced students to explore blocks aside from the "say" blocks when the player wins or loses.
    - Ask students why the instructions to play again are *after* the conditional (as opposed to the *then* or *else* section). What would happen if it was in one of those sections?
        - If the instructions were in the "then" section, they would only show up if the player won. If they were in the "else" section, they would only show up if the player lost. The player should see the instructions whether they win or lose - it does not belong inside the conditional statement because it does not rely on the condition.

Getting started with coding (15 minutes)

Use the last few minutes for students to start creating their games. They can begin by setting up accounts (if necessary). See this blog post for information about creating teacher/student accounts in Scratch.

Once students are up and running, they should follow the checklist on their code planner.

If the code planning pages are printed in black and white, it may be helpful to display this slide. It shows the blocks in color and labels the categories in which they're located in Scratch. This will help students find the blocks as they learn to navigate Scratch.

Coding (45+ minutes)

After the first day, you can spend as much or as little time coding as you'd like. With the template, some students will likely finish the basic game quickly. They can use any extra time to add additional features such as animations, sound effects, more backdrops and sprites, and whatever else they desire.

Continue to display the "Where do I find...?" slide to help students find the blocks in their code planners.

Tip: Depending on your comfort level with Scratch, students may ask questions to which you don't know the answers. That's ok! Direct students to Scratch's tutorials - these have a lot of useful information and may be able to help students get unstuck. You can also ask students to ask their peers - it's likely that someone in the room has the answer! I like to use "expert" cards to help facilitate peer collaboration in this way.

Materials

- Devices for students to access Scratch
- Scratch materials (e.g. Tutorials from Scratch)
- Math game presentation
- Code Planner Handout (for students)
- Where do I find...? Presentation to help students find blocks in Scratch if their code planners are printed in black and white
- Scratch Educators account (if you'd like to create accounts for students to save work)
- "Expert" cards to help facilitate peer collaboration
- Scratch Blocks for Google Docs and Slides add-on if you'd like to create your own images of Scratch blocks for lesson materials