Practice of adding numbers

Description

This is a simple program that allows a preschooler practice adding single digit numbers. Basically, it presents 2 numbers in countable picture form and asks how much they add up to. The user needs to perform as many additions as possible in 30 seconds. At the end the program presents the score: how many were right and wrong.

Although the program does something very simple, it requires some effort to design as described below.

Scratch and CS Concepts Used

When we design this program, we will make use of the following Scratch and CS concepts. Learn these concepts if you don't know them before proceeding further.

- Algorithms
- Animation using costumes
- Arithmetic
    - Basic operators (+, -, *, /)
- Backdrops – multiple
- Concurrency
    - Running scripts in parallel
    - Synchronization using broadcasting
- Conditional statements:
    - Conditions: YES/NO questions
    - Relational operators (=, <, >)
    - Conditionals (IF)
    - Conditionals (If-Else)
- Data types – basic
    - Integers
- Data types – strings
    - String operations (join, letter, length of)
- Events
- Looping (iteration)
    - Looping - simple (repeat, forever)
    - Looping - nested
    - Looping - conditional (repeat until)
- Program output
    - Text
- Random numbers
- Sequence

- STAMP - creating images
- Stopping scripts
- User input
    - Text
- Variables
    - Simple
    - As timer

Explore the program:

If you want to play with my final program to get a feel for how it works, click the link given at the end of the article. Try not to peek at the scripts yet, since we want to design them ourselves below.

1. Click on the “Green flag” and read the instructions.
2. Perform as many additions as you can until the timer expires.
3. The program will announce your points and stop.

High Level Design

This is where we take a step back and try to get our arms around the task of writing this program. We will first understand the basic features and the flow of operation.

There are two parts to our program. The back-end (or invisible) part does the following:

- It generates 2 random numbers for each addition.
- It verifies user's input with the actual sum.
- It repeats these operations until the time is up.
- It keeps score of correct and incorrect.

The front-end does the following:

- It presents each pair of numbers as graphical objects (animal pictures).
- It accepts user input for each addition and passes it to the back-end.

The back-end is fairly simple: it just involves 3 variables to manage each addition, and 2 more two keep the score.

For the front-end, we will use the "stamp" feature of Scratch to create graphical images on the screen. Since the two numbers to be added are single-digit, the max number of images would be 9 in the left half of the screen plus 9 in the right half. Each set of 9 images can be presented as 3 rows of 3 each. Thus, we will need a script that can mange this 3x3 grid of images.

For animal images, we could endow a single sprite with lots of animal costumes. For each addition, we could pick one of these costumes (at random) and use it to create the "stamp" images.

That is really all the work we need to do, right?

Objects:

As usual, we will use object-oriented design techniques to build our program. The first step is to list the required objects. Here is our initial list:

- Back-end logic (some invisible sprite or the stage)
- Animal sprite (with costumes of various animals)

Global data:

The numbers involved for each addition and the score would be all global variables, since both front and back-end will need them.

Let us now build the program feature by feature and design the objects incrementally. We will build all backend feature ideas in the initial version and frontend ideas in the final version.

Feature idea #1: The backend logic

Create code for the backend object to create numbers for each addition and keep score. Ask user for each addition repeatedly until timer expires. Present the final score.

Design:

Create 2 random numbers for each addition and compare their sum with user's input. Keep score and present it after timer loop is finished. We will use the stage to host the backend logic.

Feature idea #2: Help screen

Include a help screen to assist the user in playing the game.

Design:

This is just a matter of creating another backdrop with the appropriate instructions. The program will present it first when Green flag is clicked.

Save as Program Version 1

Let's save this project before continuing to the remaining ideas. Compare your program with my program below.

Solution: picture-addition-1.sb2

How to run the program:

1. Click on the "Green flag" and read the instructions.
2. Perform as many additions as you can until the timer expires.
3. The program will announce your points and stop.

Final Version

In this next version of the program, we will work on the following feature ideas:

- Create code for the front-end, i.e. code to create animal images to represent the numbers to be added.

Feature idea #3: Graphic animal images

Create animal images to represent the two numbers to be added.

Design:

As mentioned earlier, we will use a single sprite and use its costumes to create the graphic images. We will use the "stamp" feature of Scratch to create multiple images on the screen. Each number would be presented as a 3x3 grid of images, since it would be between 1 and 9.

Since there are 2 numbers, we would use the same code to draw the first number in the left half of the screen and the second number in the right half.

The main challenge is to work out the algorithm for setting up animal images in a 3x3 grid pattern. For example, if the number is 4, we need to draw one row of 3 images followed by another row of 1 image. If the number is 7, we need to draw two rows of 3 images each followed by a row of 1 image.

Here is one possible algorithm:

```
Input: N (number 1 to 9)

Y = starting Y position of top row
Repeat until N = 0
      X = starting X position of a row
      If N <= 3
            Paint a row of N images using STAMP
            N = 0
      Else
            Paint a row of 3 images using STAMP
            N = N - 3
      End if
      Y = Y - row size
End repeat
```

Save as Program Version "Final"

Congratulations! You have completed all the main features of the program. Compare your program with my program at the link below.

Solution: picture-addition-final.sb2
Link: https://scratch.mit.edu/projects/358373934/

How to run the program:

1. Click on the "Green flag" and read the instructions.
2. Perform as many additions as you can until the timer expires.
3. The program will announce your points and stop.

Author: Abhay B. Joshi (abjoshi@yahoo.com)
Last updated: 10 January 2020