Table of Numbers

In this project, we will learn how to display numbers in a table format.

Concepts used in this program:

- Algorithms
- Arithmetic operators (floor, mod, ceiling, sqrt)
- Conditional looping (repeat until)
- Conditions: YES/NO questions
- Costume-based animation
- Lists
- Logic operators (AND, OR, NOT)
- Nested looping
- Procedures with inputs
- Relational operators (=, <, >)
- Simple looping (repeat, forever)
- STAMP
- String operations (join, letter, length of)
- String variables
- Synchronization using broadcasting
- User input (ASK)
- Variables - local/global scope
- Variables – numbers

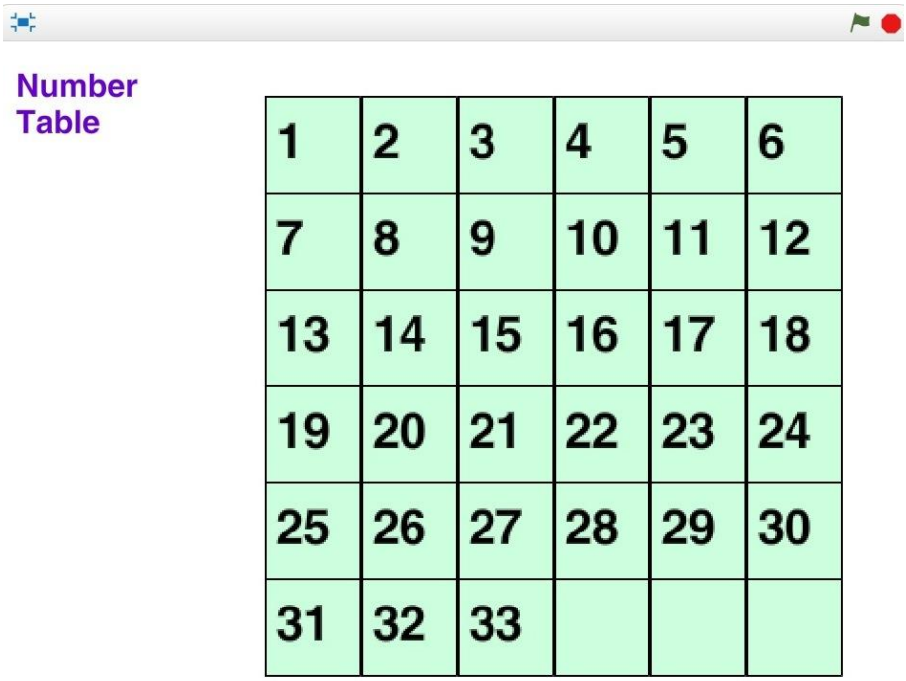
Program Description

Several programs require the display of numbers in a table format, i.e. in rows and columns. This program allows you to create a square table big enough to display natural numbers (i.e. > 0) up to a max that you specify. For example, if you specify the max as 75, the program will show a table with 9 rows and 9 columns, and display numbers 1 to 75 in this table.

Do you want to check out a working Scratch version of this program? Click on the image below (or the URL just below it). I encourage you to explore the program and its various features. But, don't look at the Scratch scripts yet; you want to design this program yourselves!

How to run the program:

1. Click on the "Green flag": everything is reset to original state.
2. When asked, enter the upper limit.
3. The program will then display numbers 1 up to the limit.



https://scratch.mit.edu/projects/318322805/

Scratch and CS Concepts Used

When we design this program, we will make use of the following Scratch and CS concepts. I assume that you are already familiar with these concepts.

Program Versions

A "program version" is nothing but a copy of our Scratch project with a unique name. In the process of designing the "number-table" program step by step, we will develop multiple program versions, such as, number-table-1, number-table-2, etc. As the version number increases, the program will have more and more features.

Program Design

Let us now see how we can build this program step by step.

Design Process:

We will build the program incrementally, i.e. we will use the following methodology. In this, we call our complete program as the "big idea" and break it down into smaller ideas.

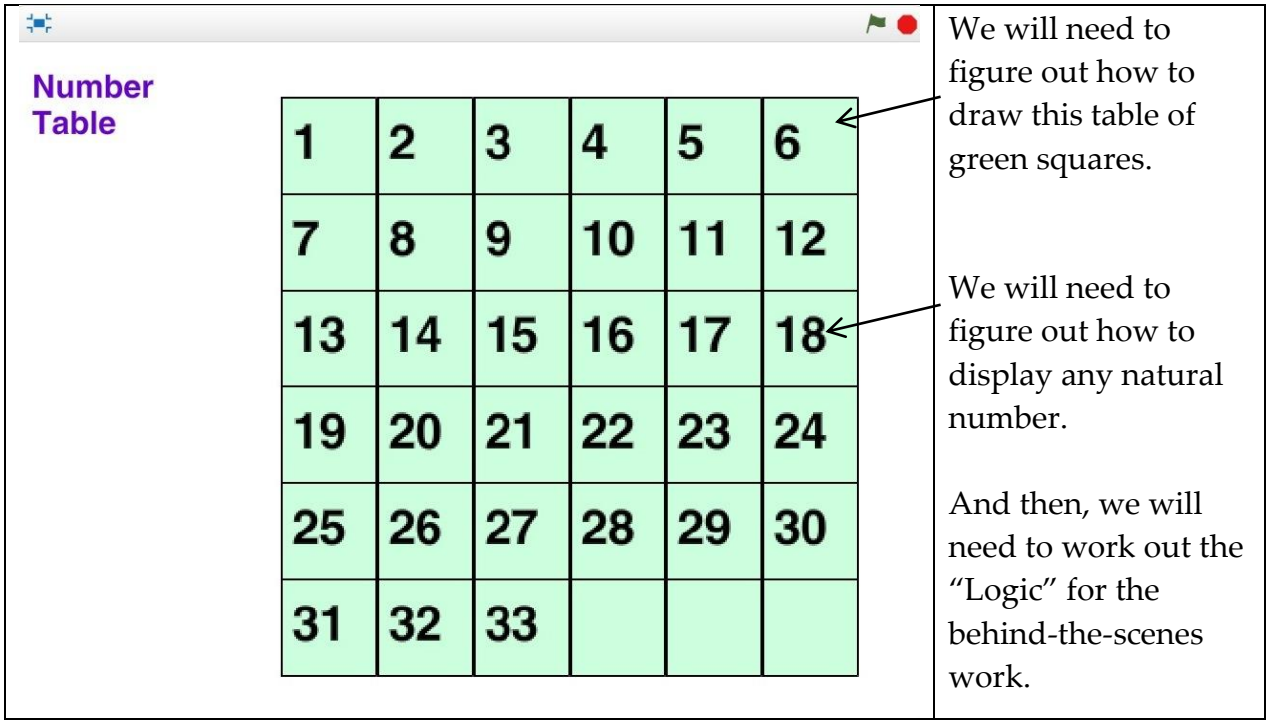
1. Take a smaller idea (a part of the "big idea").

2. Think about how it can be done (experiment in Scratch if necessary).
3. Write the program.
4. Test. Go back and forth in this process (steps 2, 3, and 4) until the idea works as expected.
5. Take up another smaller idea (another part of the "big idea") and go to step 2. Continue until all required features have been added to the program.

High Level Design:

This is where we take a step back from the computer (literally!), analyze the problem in our mind (and on a piece of paper if necessary), and break it down into multiple smaller ideas which can be programmed separately.

Let's take a look at the main screen of the game and try to point out the different pieces.



When we take a step back and consider how this problem can be solved, we see the following steps:

1. Create a list containing N numbers. (Example: 1 thru 10)
2. Draw a square table of (say R number) of rows and columns such that R*R >= N. (Example: a 4x4 table will be required to hold 10 numbers).
3. Display the numbers from the array sequentially in this table row by row. (Example: in the 4x4 table drawn above, we could show numbers 1,2,3,4 in the first row, 5,6,7,8 in the second row, and so on.)

Initially, to simplify things a bit, we will just show numbers 1 thru 16 in a 4x4 table. We will first draw a fixed 4x4 table. This could be achieved by stamping a green square sprite 16 times.

Displaying numbers in this table is a tricky challenge. First, we will work on how to display a single digit 0 to 9 (anywhere really). Then, we will work on how to compose and display a 2-digit number such as 13 by combining the single digits. Finally, we will solve the riddle of printing these numbers sequentially in the square table.

So, let's get rolling with these various ideas one by one.

For each idea, I will state the requirement, propose a "design" that will allow us to program this idea, and finally present a "solution" that presents details of Scratch scripts. You should try to think of your own design and solution before reading the ones that I propose.

Idea # 1

Draw an NxN table of filled green squares.

Design:

Refer to the program "Matrix-1" in the book "Practice CS Concepts with Scratch". It explains how this feature can be designed.

Idea # 2

Figure out how to display any natural number within a cell of the table.

Step 1:
Write a script to display a number anywhere on the screen.

Design:

Refer to the program "Show number-1" in the book "Practice CS Concepts with Scratch". It explains how this feature can be designed.

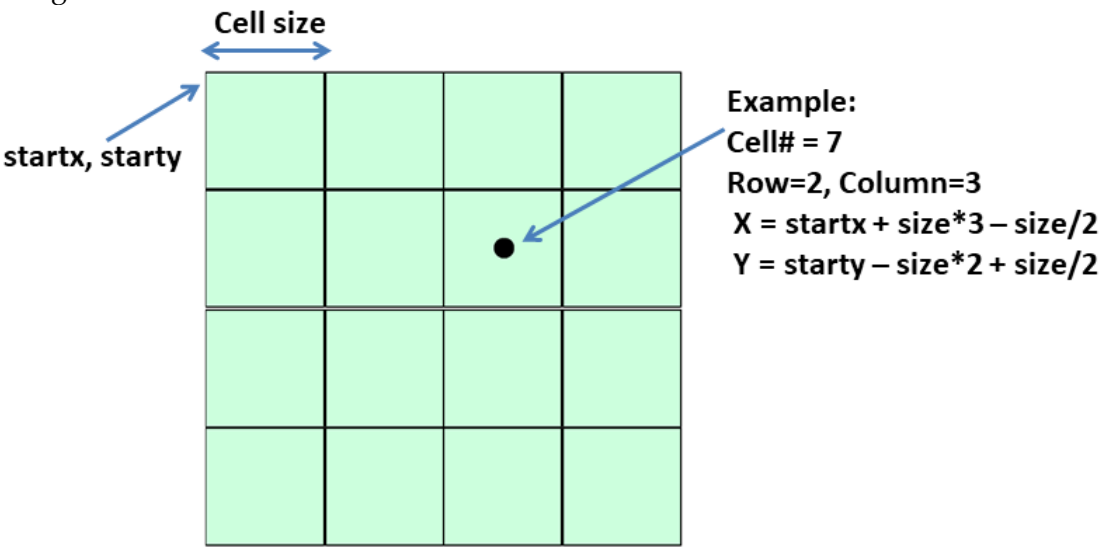
Step 2:
Finally, we need to figure out how to display a number within any cell of the table.

Design:

In Step 1, we display the number near the mouse pointer. Now, we need to change that to display instead at the X and Y of the given cell.

We will first assume a fixed 4x4 table, so we will adjust the size of the digit costumes such that a 2-digit number (e.g. 16) will fit exactly in any cell of the 4x4 table. Later we will see how we can draw a table of arbitrary size and still be able to fit numbers in its cells.

See the figure below.



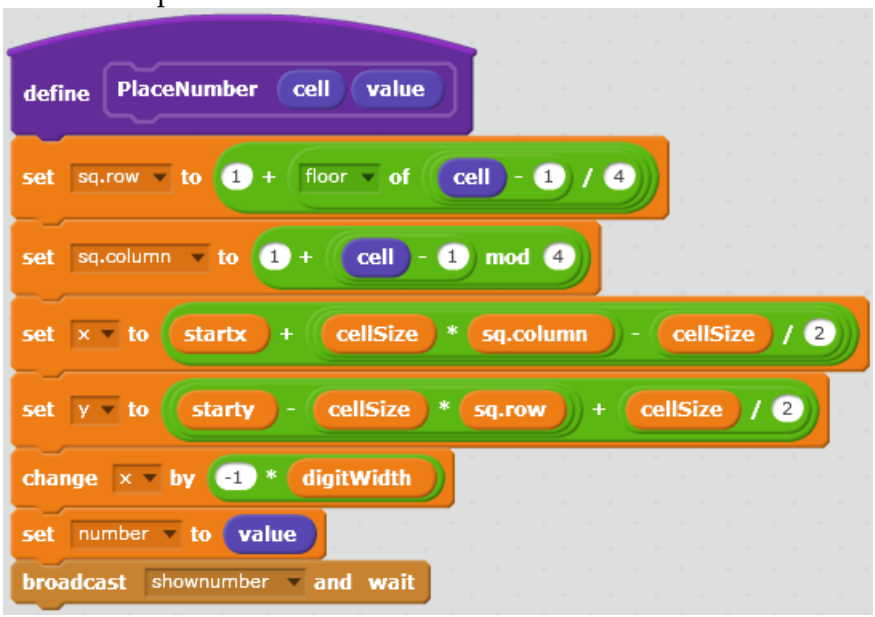
Here is the algorithm to determine X and Y of the center of a cell:

PlaceNumber: Algorithm to place a number in a cell:
Given:
Cell = which cell? = same as the location of the number in the array.
Value = number to be placed
Startx, Starty = x and y of the top-left corner of the table.
CellSize = width or height of a single cell.

```
// First, derive row and column numbers from "Cell".
Row = 1 + (Cell-1)/4        // integer division, discard remainder
Column = 1 + (Cell-1) mod 4 // remainder
// Now, calculate X and Y
X = startx + CellSize*Column - CellSize/2
Y = starty - CellSize*Row + CellSize/2
// To fit the longest number in the cell we will shift left
X = X - digitSize
Call DisplayNumber // defined above
```

Solution:

Script for the "Number" sprite:



Save as Program Version 1

Before continuing to the next idea, make a copy of your project under a different name. (For example, I am calling my copy as Number-table-1). This way, you have a backup of your project that you can go back to if required for any reason.

Compare your program with my program at the link below.

Number-table-1: includes ideas 1 and 2 explained above.
Link: https://scratch.mit.edu/projects/318321601/

Idea # 3

Now include step 1 of the main design steps.

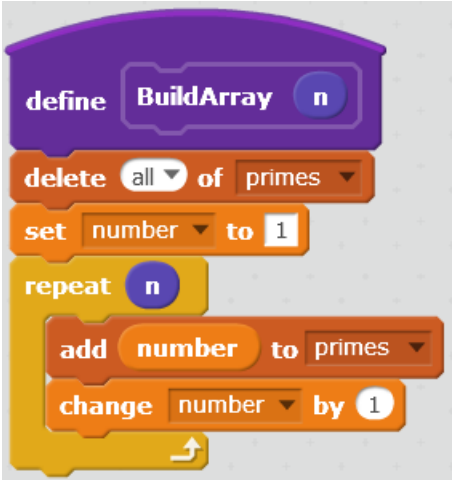
Design:

This is straightforward: we can just use a List variable to create the list of numbers.

Go ahead and write your own script to implement this algorithm, and then compare it with the script given below.

Solution:

Script for the "Number" sprite:



Idea # 4

Display the numbers!

Design:

Earlier, we saw how to display a number within any cell of the table. We made the assumption that we have a fixed 4x4 table and the size of the digits is such that a 2-digit number (e.g. 16) will fit exactly in any cell of the 4x4 table.

Now, we will see how we can draw a table of arbitrary size and still be able to fit numbers in its cells.

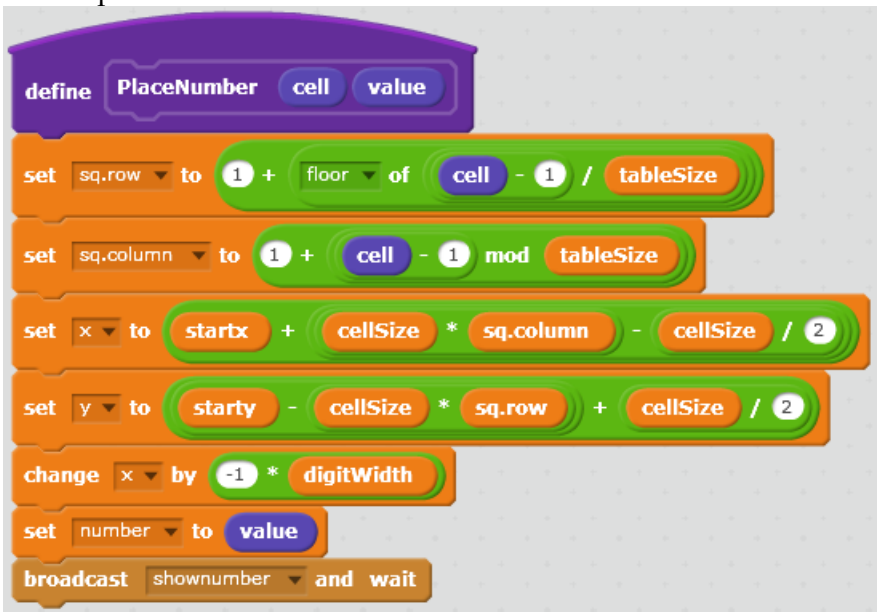
First, we will slightly modify the algorithm PlaceNumber. Instead of using 4 as the number of rows (and columns), we will the variable TableSize so that the algorithm will work for tables of any size.

PlaceNumber: Algorithm to place a number in a cell:
Given:
Cell = which cell? = same as the location of the number in the array.
Value = number to be placed
TableSize = number of rows (or columns) in the table
Startx, Starty = x and y of the top-left corner of the table.
CellSize = width or height of a single cell.

```
// First, derive row and column numbers from "Cell".
Row = 1 + (Cell-1)/TableSize   // integer division, discard remainder
Column = 1 + (Cell-1) mod TableSize // remainder
// Now, calculate X and Y
X = startx + CellSize*Column - CellSize/2
Y = starty - CellSize*Row + CellSize/2
// To fit the longest number in the cell we will shift left
X = X - digitSize
Call DisplayNumber // defined above
```

Solution:

Script for the "ball" sprite:



Save as Program Version "Final"

Congratulations! You have completed all the features of the game. As before, let's save this project under a different name. (For example, I am calling my copy as Number-table-final).

Compare your program with my program at the link below.

Number-table-final: includes all the features listed above.
Link: https://scratch.mit.edu/projects/318322805/