Overview

- This tutorial will show you how to create a fun and interactive multiplication table game.
- This comes with a starting world where the objects are already set up, so the programmer will focus on writing the methods.
- In the game, the player is prompted to answer 121 multiplication questions in a different order each time they play.
- Programming concepts covered include arrays, nested loops, Whiles, animation, and other basic Alice constructs.

Starting world

- If you haven’t already done so, download the stating world from the website.
- Look at the object tree: everything is already set up for you.
  - Click on each object in the tree to make sure you understand what they are referring to.
  - The numberObjects folder holds all of the hidden answers in the table.

Understanding fillRandomly

- Click on world in the object tree, and click on its methods panel.
- Click edit next to fillRandomly.
- The allNumberObjects array holds all of the objects in the numberObjects folder in order from 0x0 to 10x10.
- We will fill the allAnswers array so that it holds all the answers.
Adding in the answers

- Drag a **Loop** into the method and select **other...** and then type **11**, and click **show complication version**.
  - This outer loop represents each row, so rename it **row**.
- Drag another **Loop** into the **Do Nothing** and select **11** again, and click **show complicated version**.
  - This inner loop represents each column, so rename it **column**.
- Drag **allAnswers** into the **Do Nothing** and select set item <index> to <item>, expressions, row, expressions, and **row** (for now).

- Notice that there are 25 items before the green 2x3 square.
- Since we start counting from 0, 2x3 is square 25.
  - 25 = 11*2 + 3, or square number = number of full rows before it * number of squares per row + number of squares before it in its own row.
  - Try this with other numbers to see this pattern.

- We want each answer to be its row multiplied by its column.
- Click on the first **row** and select **math**, row * expression, column.
- Drag the **purple box** you just made to the second time you see row.

- Which item number in the array are we trying to change?

- The item we’re looking for is the row number multiplied by 11 plus the column number, or row*11 + column in terms of our variables.
- Click the first **row** in set item line and select **math**, row *, 11.
- Click the arrow after (row*11) and select **math**, (row*11) +, expressions, column.

- Click on **more** and change the duration to 0 seconds.
Randomization

- To create a random order of multiplication problems in the game, we need to shuffle the elements in both of our arrays.
- Drag in a Loop, select other, and type 121 because that is the number of squares we have in the table.
- Drag a Do together into the Loop.
- Drag a Do in order into the Do together.

Understanding the shuffle

- Suppose we want to swap the yellow and green squares.
- We move the green to a temp spot, put the yellow in its place, then put the green in the yellow’s former place.

Random number

- Drag rand above the Do together, and select set value and 1, for now.
- From the world’s functions tab, drag random number onto the 1.
- We want a random integer between 0 and 120 since there are 121 objects in the array.
- Click on the more of random number more and select minimum, 0.
- Click on more again and select maximum, 121.
- Click on more again, select integerOnly and then true.
• We want to switch the item at the index of the Loop and the item in the rand place.

• Drag allNumberObjects at the bottom and select set item <index> to <item>, then expressions, index, and square, for now.

• Drag allNumberObjects onto square and select ith item from array, expressions, and then rand.

• Now drag allNumberObjects below that line and select set item <index> to <item>, then expressions, rand, expressions, tempObject.

<table>
<thead>
<tr>
<th>Do in order</th>
</tr>
</thead>
<tbody>
<tr>
<td>tempObject  set value to</td>
</tr>
<tr>
<td>set item index to</td>
</tr>
<tr>
<td>set item rand to</td>
</tr>
</tbody>
</table>

• Below that drag allAnswers again and select set item <index> to <item>, expressions, rand, expressions, tempNumber.

• Important: make sure to change the duration to 0 (or false) in each line in this loop.

<table>
<thead>
<tr>
<th>Loop</th>
<th>index from 0 up to (but not including) 121 times incrementing by 1 show simple version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do together</td>
<td></td>
</tr>
<tr>
<td>rand set value to</td>
<td>randomNumber minimum = 0 maximum = 120 integerOnly = true in more...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Do in order</th>
</tr>
</thead>
<tbody>
<tr>
<td>tempObject  set value to</td>
</tr>
<tr>
<td>set item index to</td>
</tr>
<tr>
<td>set item rand to</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Do in order</th>
</tr>
</thead>
<tbody>
<tr>
<td>tempNumber  set value to</td>
</tr>
<tr>
<td>set item index to</td>
</tr>
<tr>
<td>set item rand to</td>
</tr>
</tbody>
</table>

**Shuffling allAnswers**

• Drag in another Do in order below the first one.

• Create a variable of type number called tempNumber.

• Drag tempNumber into the Do in order and select set value and then 1, for now.

• Drag allAnswers onto the 1 and select ith item from array, expressions, and then index.

• Drag allAnswers below that line and select set item <index> to <item>, expressions, index, and 1, for now.

• Drag allAnswers onto the 1 and select ith item from array, expressions, and rand.

**Hiding the instructions**

• Now that everything is shuffled, we want to hide the instructions and start the game.

• Drag a Do together into the bottom of the method.

• Click the + next to instructions in the object tree to see where the instructions are.

• Click on Line1 under instructions, and drag isShowing from its properties tab into the Do together, select false, and change the duration to 0 seconds.

<table>
<thead>
<tr>
<th>Do together</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line1 set isShowing to false duration = 0 seconds</td>
</tr>
<tr>
<td>Line2 set isShowing to false duration = 0 seconds</td>
</tr>
<tr>
<td>Line3 set isShowing to false duration = 0 seconds</td>
</tr>
</tbody>
</table>

• Do the same for Line2, 3, 4, 5, and 6.
Starting the game

- Let's have a square move to the box in the table that we are going to fill next.
- Drag a Loop below the Do together, select 121.
- Click on square in the object tree; go to its methods.
- Drag square move to into the Loop and select camera. Drag allNumberObjects onto camera, and select ith item from array, expressions, index.
- Drag in square move and select up, .1 meters, and set the duration to 0 seconds.

Asking the question

- Create a new variable called `playerAnswer` of type other, String.
- Drag `playerAnswer` below the square move up line and select set value, default string.
- From the world’s functions tab, drag ask user for a string onto default string and select Enter a String, for now.
- Drag what as a string from world’s functions onto Enter a String, and select camera, for now.
- Drag allNumberObjects onto camera and select ith item from array, expressions, index.

Create method: checkAnswer

- We want to make a method that checks if the player enters the correct answer.
- Create a new world method called checkAnswer.
- Create 3 new parameters: `numberOfObject` of type object, `playerAnswer` of type string, and `correctAnswer` of type string.
- Drag an If/Else statement into the method and select true.
• Drag playerAnswer onto the true and select playerAnswer ==, expressions, correctAnswer.
• Alice stores all numbers as decimal numbers, so we will need to add a “.0” to our numbers before comparing them to the correct answer.
• To correct this, drag a joined with b from world’s functions onto playerAnswer, select other, and type “.0”.

![Image of code block](image)

**Game over**

• Click Add Objects, select Create 3D Text Object, and type “GAME OVER”.
• In the object tree, rename 3D Text to “gameOver”.
• Resize it, and move it in the middle of the screen.
• Go to gameOver’s properties tab, and change isShowing to false.
• Drag isShowing right before your Wait 1000 seconds line, and select true.

![Image of code block](image)

• Drag numberObject into the last Else and select set color to, red, and set the duration to 0.
• There are two options left: the player was incorrect or he wants to see the answer.
• Drag an If/Else statement below that line and select true.
• Drag playerAnswer onto the true and select playerAnswer ==, other, and type answer.
• Copy the numberObject isShowing line and paste it into the first Do nothing.

![Image of code block](image)
• Drag playerAnswer into the Else *Do Nothing* and select *set value, default string*.

• From world’s methods, drag *ask user for a string* onto default string, select other, and type “Try again: ”.

• Also from world’s methods, drag a joined with b onto the Try again, and select default string.

• Drag what as a string onto default string and select expressions, numberObject.

While...

• We want the player to keep trying until he gets it right, or types “answer” or “end”, so we want to run this method until one of those conditions is met.

• Create a *Boolean* variable named keepRunning, and set its initial value to *true*.

• Drag a *While* into the beginning of the method, and select expressions, keepRunning.

• Drag keepRunning into the *While*, and select false.

• Drag keepRunning right below the last line (player set value), and select true.

• Now drag the entire method (the big If/Else) into the *While*, so the method is just one big While.

Connecting the methods

• Open your *fillRandomly* method.

• Drag checkAnswer from world’s methods into the end of the Loop at the end, and select camera (for now), expressions, playerAnswer, expressions, correctAnswer.

• Drag allNumberOfObjects onto camera and select ith item from array, expressions, index.

• Open my first method.

• Drag *fillRandomly* into the *Do Nothing*.

• From gameOver (in object tree)’s properties tab, drag isShowing into the method and set it to true.

Make sure all your code is correct:
Finished!!
Play your world and see how many multiplication problems you can answer correctly!