The Matrix

By Melissa Dalis
Professor Susan Rodger
Duke University
June 2011
Overview

• This tutorial will show you how to create an interactive matrix multiplication tool in Alice that you can use to multiply any two 2x2 matrices.

• This world will help reinforce each step in the easily forgotten multiplication process.

• Programming ideas covered include arrays, variables, helper methods, billboards, and other basic Alice concepts.
Starting world

• Make sure you’ve downloaded the starting world from the website.

• Go to the object tree, and notice that eqnBrackets, ansBrackets, multSign, and multSignSpot form the setup for the matrices.

• Click on the + sign next to the leftMatrix folder.
• Click on `left(0,0)` in the object tree and go to its properties tab.
• Change `isShowing` to `true`.
• Click on `left(0,0)Spot` and change its `opacity` to `1` (100%: fully opaque).
• Notice that `left(0,0)` is the number object, and `left(0,0)Spot` is the green square behind it.
• Undo what you just did so that the two objects are again invisible.
Matrix Notation

• Look at other objects in the folders and see that they work the same way.
• Notice that the first item in the left matrix is called left(0,0). In computer programming, we usually start counting from 0. So, this means that left(0,0) is in the 0th row and 0th column. We always refer to the row in the first index and the column in the second.
Adding instructions

• Make sure you’ve downloaded the matrixInstructions file from the website.
• At the top of Alice, click File, Make Billboard, and find matrixInstructions.

• Move and resize matrixInstructions so that it covers most of the screen like in the picture.
• Read the instructions so that you understand what goes on in the game that you are creating.
Deciding when to start the game

• Click on world in the object tree, and go to its properties panel.

• Click create new variable, and name it start, make it type Boolean, and set its starting value to false.

• Create a new event in the top right corner, and select When a key is typed.

• Click on any key, and select Space.

• Drag a Do in order onto the Nothing.
• Click on `matrixInstructions` in the object tree, and go to its properties tab. Drag `isShowing` into the Do in order in the event that you just made, and select false.

• From `world’s properties` panel, drag `start` into the Do in order, and select set value, true.

• Drag a While into my first method, and select true.

• Drag `start` onto the true, and then click on it, select logic, not start.
When a number is clicked...

- We want to be able to tell when the player clicks on a number in one of the matrices.
- Create a **new object variable in world’s properties** named `numberClicked`.
- Create a **new event for When the mouse is clicked on something**.
- Change anything to `leftMatrix, left(0,0)`.
- Drag `numberClicked` onto the Nothing and select **set value, leftMatrix, left(0,0)**.
Replicating events

• Drag the event you just made for left(0,0) onto the clipboard at the top corner to make a copy.
• Drag the clipboard into the Events box to paste, and change left(0,0) to left(0,1).
• Do this 10 more times for left(1,0), left(1,1), right(0,0), right(0,1), right(1,0), right(1,1), ans(0,0), ans(0,1), ans(1,0), and ans(1,1).
generateNumbers

• Click on world in the object tree, and go to its methods.

• Click create new method, and name it generateNumbers.

• This method will be called 8 times to fill all of the spots in the left and right matrices.

• Create 3 new parameters: numberObject (other, 3D text), spot (object), and index (number).
  – Make sure you’ve downloaded the latest version of Alice if 3D text doesn’t come up as an option.

• Create one new number variable in the method called numberEntered.
• We want a green square to show up beneath the number we are asking the user to enter, all of the squares (left(0,0)Spot, etc.) are set to 0 opacity.
• Drag the variable you created named spot into the method and select set opacity to, 1.
• Click more, duration, other, 0.
• Drag numberEntered into the method, and select set value, 1, for now.
Click on **world** in the object tree and go to its **functions**.

- Drag **ask user for a number**, select **other**, and type **“Enter a number for the green spot in the matrix.”**
- We want to keep track of all the number values.
- In world’s properties tab, click **create new variable**, name it **matrixNumbers**, of type **number**, check **make a List**, and change **List** to **Array**.
- Click **new item** 8 times so that you have items 0-7, which will represent all of the numbers in the matrices.
• Drag matrixNumbers into the method, select set item <index> to <item>, expressions, index, expressions, numberEntered, set duration to 0.

• Find left(0,0) in the object tree, drag text from its properties tab into the method, and select default string, for now.

• Drag numberObject onto left(0,0).

• From world’s functions tab, drag what as a string onto default string, and select expressions, numberEntered.

• To make the number show up as an integer rather than a decimal, drag int a as a String (from world’s functions) onto numberEntered, and select expressions, numberEntered, and set duration to 0.
Making the number appear

- After we change the text of the number object, we want it to appear in the matrix in place of the green square that was previously there.

- Drag **spot** to the end of the method, select **set opacity to, 0**, and change the **duration** to **0**.

- Since **isShowing** doesn’t automatically show up in the menu when we would drag **numberObject** into the method, find **isShowing in ground’s properties** (in object tree), drag it into the method, and set it to **true**.

- Drag **numberObject** onto **ground**, and set **duration** to **0**.
calculateAnswers

• Create a new world method named calculateAnswers.

• First, we want calculateAnswers to call generateNumbers for each of the numbers and spots in the matrices.

• Create a new world variable called numberObjects of type other, 3D Text, check Make an Array, and click new item until you have items 0-7.

• Set the items to left(0,0), left(0,1), left(1,0), left(1,1), right(0,0), right(0,1), right(1,0), and right(1,1), in that order.
Since we also need to loop through all the spots, create another **world variable** named `allSpots`, and make it an **object array**.

- Add items until there are items 0-7, and set the items to `left(0,0)Spot`, `left(0,1)Spot`, `left(1,0)Spot`, `left(1,1)Spot`, `right(0,0)Spot`, `right(0,1)Spot`, `right(1,0)Spot`, and `right(1,1)Spot`, in that order.

- Drag a **Loop** into `calculateAnswers` that runs 8 times, and click on **show complicated version**.

- From **world’s methods**, drag `generateNumbers` into the **Loop**, and select `camera`, `camera`, `expressions`, `index`, for now.
• From world’s properties, drag `numberObjects` onto the first camera, and select `ith item from array, expressions, index`.

• Also from world’s properties, drag `allSpots` onto the second camera, and select `ith item from array, expressions, index`.

• So now that we’ve generated all of the numbers, we want to calculate the answers.

• Create 4 new world number variables: `#ans(0,0)`, `#ans(0,1)`, `#ans(1,0)`, and `#ans(1,1)`.
How to multiply matrices

\[
\begin{bmatrix}
0 & 1 \\
2 & 3
\end{bmatrix}
\ast
\begin{bmatrix}
4 & 5 \\
6 & 7
\end{bmatrix}
=
\begin{bmatrix}
0 \ast 4 + 1 \ast 6 & 0 \ast 5 + 1 \ast 7 \\
2 \ast 4 + 3 \ast 6 & 2 \ast 5 + 3 \ast 7
\end{bmatrix}
\]

• Each of the numbers above represents the number object in that index of our numberObjects array.

• In other words...

  – \( \text{ans}(0,0) = \text{left}(0,0) \ast \text{right}(0,0) + \text{left}(0,1) \ast \text{right}(1,0) \)
  
  – \( \text{ans}(0,1) = \text{left}(0,0) \ast \text{right}(0,1) + \text{left}(0,1) \ast \text{right}(1,1) \)
  
  – \( \text{ans}(1,0) = \text{left}(1,0) \ast \text{right}(0,0) + \text{left}(1,1) \ast \text{right}(1,0) \)
  
  – \( \text{ans}(1,1) = \text{left}(1,0) \ast \text{right}(0,1) + \text{left}(1,1) \ast \text{right}(1,1) \)
So how do we code that?

- Drag \#ans(0,0) into the method, and select set value, 1.
- Drag matrixNumbers onto the 1, and select ith item, 0.
- Click on the last white arrow in the line and select math, *, 1.
- Drag matrixNumbers onto the 1, and select ith item, 4.
- Click on the last arrow in the line and select math, +, 1.
- Do the same so that you have (item0*item4)+(item1*item6)
- Change 0 to 4 and 1 to 6.
• Use the clipboard to copy and paste this line 3 times to calculate all of the answer spaces, keeping in mind the graphic on slide 21:

– Replace 0,4,1,6 with 0,5,1,7, then 2,4,3,6, then 2,5,3,7.
– Replace world.#ans(0,0) to world.#ans(0,1), (1,0), then (1,1).
Changing the text

• Find \texttt{ans(0,0)} from the \texttt{answerMatrix} folder in your \texttt{object tree}, and go to its \texttt{properties} tab.

• Drag \texttt{text} to the line below the first answer set value line, and select \texttt{default string}, for now.

• From world’s functions, drag \texttt{what} as a string onto \texttt{default string}, and select camera.

• From world’s functions, drag \texttt{int a} as a \texttt{String} onto \texttt{camera}, and select expressions, \texttt{#ans(0,0)}.
• Make 3 copies of that line, and drag each of them below an answer set value line so the answer set value and ans(0,0) set text lines are alternating.
• Change the second ans(0,0) to ans(0,1), the third to ans(1,0), and the fourth to ans(1,1), so that each couple of line refers to the same answer.
• Make sure all durations are set to 0 (or false) seconds.
answerAnimation

• Create a **new world method** named answerAnimation.
• This method will help the user calculate the answer.
• In the method, create **8 new object parameters** for the number objects involved in calculating their answers and their respective spots: num1, num1Spot, num2, num2Spot, num3, num3Spot, num4, num4Spot.
• To deal with the answer, create **2 object parameters**: ans and ansSpot. Also create a number parameter #ans to keep track of the actual number answer.
• Drag **ansSpot** into the method, and select **set opacity to, 1**, and set the **duration to 0** seconds.

• For each number object that we’re dragging to the equation at the bottom, we want the correct spot in the equation to turn green to indicate where the number should be dragged, wait for the user to click on a number to be moved, move the number, and then turn the equation spot to gray.

• Create a helper (world) method called **answerAnimationHelper** so that we can do this for all of the numbers.
• Create 2 object parameters for this helper method: numberObject and equationSpot.
• Drag equationSpot into the method, and select set color to, green, and set duration to 0.5.
• Drag a While into the method and select true.
• From world’s properties, drag numberClicked onto the true, and select numberClicked !=, expressions, numberObject.
• Drag numberObject into the method, and select move to, expressions, equationSpot.
• Drag equationSpot into the method, select set color to, light gray, and set the duration to 0 seconds.
• (Code on next page)...
Check your code:

- `world.answerAnimationHelper (Obj numberObject, Obj equationSpot)`
- No variables

- `equationSpot` set color to `duration = 0.5 seconds` more...
- `While` `world.numberClicked` `!=` `numberObject`
  - (Do Nothing)
- `numberObject` move to `equationSpot` more...
- `equationSpot` set color to `duration = 0 seconds` more...
Using animationHelper

• Go back to your `answerAnimation` method.
• From world’s methods, drag `answerAnimationHelper` into the method, and select expressions, `num1`, expressions, `eqnSpot1`.
• Do the same for `num2` and `eqnSpot2`, `num3` and `eqnSpot3`, `num4` and `eqnSpot4`.
  – `eqnSpots` are in the object tree

```
world.answerAnimationHelper numberObject = num1  \ equationSpot = eqnSpot1  \\
world.answerAnimationHelper numberObject = num2  \ equationSpot = eqnSpot2  \\
world.answerAnimationHelper numberObject = num3  \ equationSpot = eqnSpot3  \\
world.answerAnimationHelper numberObject = num4  \ equationSpot = eqnSpot4  \\
```
Asking user for answer

• From the equationStuff folder in the object tree, drag eqnAnsSpot into the method, and select set color to, green.

• Create a number variable in the method called playerGuess.

• Drag playerGuess into the method, and select set value to, 1, for now.

• From world’s functions, drag ask user for a number onto the 1, and type “Answer to equation?”.
Try again

- Drag in a **While**, and select **true**.
- Drag `playerGuess` onto the **true**, and select `playerGuess !=`, **expressions**, `#ans`.
- Click on the arrow at the end of the line, and select **logic**, `playerGuess !=` and, **true**.
- Drag `playerGuess` onto the **true**, and select `playerGuess !=`, **other**, `-1`.
- Drag `playerGuess` into the **Do Nothing**, and select **set value**, `1`.
- From world’s functions, drag `ask user for a number` onto `1`, and type “Try again or type -1 to see answer.”
• Below the While, drag in eqnAnsSpot, select set color to, light gray, and set duration to 0.
• Drag in ans, and select set opacity to, 1.
• Drag in ansSpot, and select set color to, green.
• Drag in a While, and select true.
• From world’s properties, drag numberClicked onto the true, and select numberClicked !=, expressions, ans.
• Below the While, drag in ans, and select move to, ansSpot.
Moving things back

- Drag \texttt{ansSpot} into the method, select \texttt{set opacity to}, 0, and set the \texttt{duration} to 0.
- Drag in a \texttt{Do together}.
- Drag \texttt{num1} into the \textit{Do Nothing}, and select \texttt{move to}, \texttt{expressions}, \texttt{num1Spot}.
- Do the same for \texttt{num2} and \texttt{num2Spot}, \texttt{num3} and \texttt{num3Spot}, \texttt{num4} and \texttt{num4Spot}. 
Connecting the methods

• Go back to world.my first method.
• From world’s methods, drag in calculateAnswers.
• Now we want to call answerAnimation for all 4 of the items in the answer matrix.
• Remember from earlier that:
  – \( \text{ans}(0,0) = \text{left}(0,0)\times\text{right}(0,0) + \text{left}(0,1)\times\text{right}(1,0) \)
  – \( \text{ans}(0,1) = \text{left}(0,0)\times\text{right}(0,1) + \text{left}(0,1)\times\text{right}(1,1) \)
  – \( \text{ans}(1,0) = \text{left}(1,0)\times\text{right}(0,0) + \text{left}(1,1)\times\text{right}(1,0) \)
  – \( \text{ans}(1,1) = \text{left}(1,0)\times\text{right}(0,1) + \text{left}(1,1)\times\text{right}(1,1) \)
So let’s code that

• From world’s methods, drag `answerAnimation` into the method, and choose `left(0,0), left(0,0)Spot, right(0,0), right(0,0)Spot, left(0,1), left(0,1)Spot, right(1,0), right(1,0)Spot, ans(0,0), ans(0,0)Spot, #world.ans(0,0).
  – These can be find in `leftMatrix`, `rightMatrix`, `answerMatrix`, and expressions/world’s properties.

• Drag in `answerAnimation` again, and choose `left(0,0), left(0,0)Spot, right(0,1), right(0,1)Spot, left(0,1), left(0,1)Spot, right(1,1), right(1,1)Spot, ans(0,1), ans(0,1)Spot, #world.ans(0,1).

• According to your knowledge of this program and multiplying matrices, try doing the next two lines yourself.
Check to see if you’re right

- **3rd line**: \( \text{left}(1,0), \text{left}(1,0)\text{Spot}, \text{right}(0,0), \text{right}(0,0)\text{Spot}, \text{left}(1,1), \text{left}(1,1)\text{Spot}, \text{right}(1,0), \text{right}(1,0)\text{Spot}, \text{ans}(1,0), \text{ans}(1,0)\text{Spot}, \text{world.}\#\text{ans}(1,0). \)

- **4th line**: \( \text{left}(1,0), \text{left}(1,0)\text{Spot}, \text{right}(0,1), \text{right}(0,1)\text{Spot}, \text{left}(1,1), \text{left}(1,1)\text{Spot}, \text{right}(1,1), \text{right}(1,1)\text{Spot}, \text{ans}(1,1), \text{ans}(1,1)\text{Spot}, \text{world.}\#\text{ans}(1,1). \)
Finishing touches

• Click on **light** in the **object tree**, and go to its **properties** tab.

• Change the **brightness** from 1 to 2.

• Click on **goodJob** in the object tree, and go to its **properties** tab.

• Drag **isShowing** into the bottom of **my first method**, and select **true**.
Good job!!
Play your
world!