7-5-2022

Using Magic to Teach Computer Programming

Dale F. Reed
University of Illinois at Chicago, reed@uic.edu

Ronald I. Greenberg
Loyola University Chicago, Rgreen@luc.edu

Recommended Citation

Dale F. Reed and Ronald I. Greenberg. Using magic to teach computer programming. In EDULEARN22 Proceedings, 14th International Conference on Education and New Learning Technologies, pages 3240--3248. IATED, July 2022.

USING MAGIC TO TEACH COMPUTER PROGRAMMING

Dale F. Reed[1], Ronald I. Greenberg[2]

[1]University of Illinois Chicago (UNITED STATES)
[2]Loyola University Chicago (UNITED STATES)

Abstract

Magic can be used in project-based instruction to motivate students and provide a meaningful context for learning computer programming. This work describes several magic programs of the "Choose a Number" and "Pick a Card" varieties, making connections to underlying computing concepts.

Magic tricks presented as demonstrations and programming assignments elicit wonder and captivate students' attention, so that students want to understand and replicate the work to show it to friends and family members. Capturing student interest and curiosity motivates them to learn the underlying programming concepts.

Two "Choose a Number" programs are shown where the computer is able to identify a user's choice among many alternatives:

1. Magic Letter: A table of numbers with associated random letters is displayed. Participants make choices and the computer identifies their chosen letter.
2. Number Boxes: The user chooses a secret number, for example the age at which their business becomes successful and is sold. The user indicates which boxes displayed on the screen contain their age, and the computer then identifies their secret number.

Two "Pick a Card" varieties are shown, where board pieces represent cards, the user selecting one of them that the computer later identifies:

1. Secret Tile: A grid of black/white tiles are displayed, where the computer verifies which secret tile selected by a volunteer has been flipped.
2. Twenty-one Card Magic: Twenty-one cards are selected from a shuffled deck and displayed face-up, with a volunteer choosing a secret card. After repeated stacking and re-dealing, the secret card is identified.

These examples illustrate computing concepts and can be used in classroom explanations and programming assignments. Links are given to online playable versions, assignment descriptions, and magic-themed resources in computing.

Keywords: Magic, Computer Programming, STEM education, Binary, Parity, Search, Number Guess

1 INTRODUCTION

The science fiction writer Arthur C. Clarke [1] wrote that "any sufficiently advanced technology is indistinguishable from magic." As technology continues to grow at an exponential rate, new applications regularly arise that surprise us in what they can do. The term "magic" used in our context, means *tricks*, where misdirection and unexpected results suggest abilities and features that are not really there, relying not on "sleight of hand" but rather "sleight of mind".

Curzon and McOwan [2] point out that magic tricks can arouse curiosity and give a sense of delight, pointing out that advanced technology has been and continues to be the basis for magic illusions [3]. Teaching math using curious properties of numbers and using discrepant events in physics instruction have long been used to "to stimulate interest, motivate students … and promote higher-order thinking skills" [4]. We want to capitalize on this student interest to have them practice basic programming skills using variables, loops, decision statements, input and output. The "Choose a Number" and "Pick a Card" programs using these basic skills are presented in turn below.

2 CHOOSE A NUMBER

In these problems, multiple values are displayed or described, with the user making an undisclosed choice. After some interaction the computer mysteriously is able to reveal the user's choice.

2.1 MAGIC LETTER

In Magic Letter, a table of numbers and associated letters are displayed. The user is invited to select a position in the table and then follow the instructions given, as shown in *Fig. 1*:

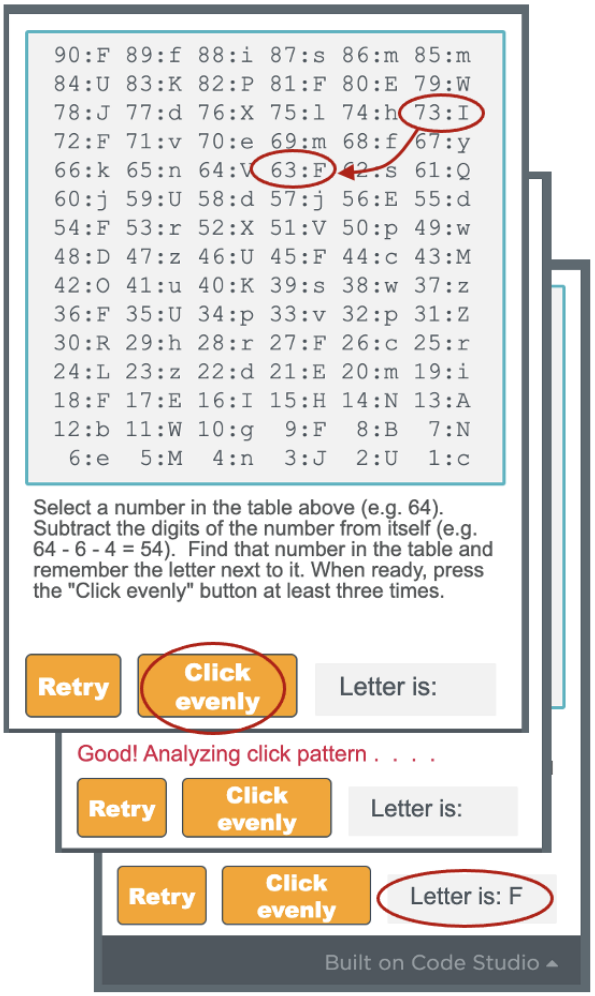


Figure 1. Magic Letter playable online at bit.ly/uicmindreader

Let's say the user chose position 73:I, which is circled in *Fig. 1*. The instructions then indicate to subtract the digits in the selected number to find a new location, so for 73 we subtract 7 and subtract 3, giving: 73 - 7 - 3 = 63, making note of the letter in this new location, which in this case is '**F**'. The user is prompted to press the "Click evenly" button at even intervals, showing the second screen in *Fig. 1*. Finally in the last screen the program reveals the user's letter.

Before giving away the secret, we encourage you to give it a try yourself a few times!

The secret to the program (shh... don't tell!) is that the program itself chooses a random "special character" ahead of time, which in the case of the table above is the character 'F'. This is coupled with the mathematical fact that for any positive integer, if you subtract its individual digits from itself, you end up with a number that is evenly divisible by 9. Thus 29 - 2 - 9 = 18, which is 9 x 2, and 61 - 6 - 1 =

54, which is 9 x 6, and so on. When the table is created, all the numbered locations below 90 where 9 is a factor (9, 18, 27, 36, 45, 54, 63,72, 81) are set to the special character 'F' that was chosen ahead of time. For all the rest of the characters, a random character is chosen. The whole notion of “clicking evenly” so the program can “analyze the click pattern” is just misdirection.

Likewise a purely text-based version (*Fig. 2*) is readily accessible to beginning programmers, requiring knowledge of Input/Output, variables, assignment, a decision (if) statement, and a loop. The most complicated part of the program is how to use a random number generator to give random characters. Brief replicable examples can be provided of how to generate random numbers and how to use them to get a random upper or lower-case character. In the experience of the authors, students are excited to get the program to work so they can show their friends. The text-based version can simply be printed or displayed for one-time use, omitting the final reveal of the secret letter, which the presenting “magician” can provide.

```
Program #3: Mind Reader
BTT CS 111: Program Design I in Python

Our subconscious get expressed in different ways, including
through choices we make, how we type, and how quickly we type.
Python libraries include Artificial Intelligence (AI) neural
network tools that can recognize patterns.
For this program choose the algorithm analysis level. Higher
levels take longer, but are more accurate. Level 4 works most
of the time.
Enter level > 1: 0

99:y 98:i 97:P 96:o 95:u 94:g 93:p 92:m 91:T 90:y
89:w 88:I 87:H 86:D 85:k 84:a 83:U 82:A 81:E 80:M
79:G 78:N 77:A 76:H 75:O 74:P 73:h 72:E 71:H 70:H
69:O 68:J 67:a 66:R 65:u 64:F 63:E 62:X 61:j 60:x
59:X 58:W 57:N 56:v 55:j 54:E 53:S 52:P 51:q 50:S
49:b 48:H 47:m 46:v 45:E 44:L 43:W 42:V 41:l 40:O
39:q 38:y 37:Q 36:E 35:m 34:P 33:a 32:b 31:W 30:T
29:s 28:u 27:E 26:F 25:H 24:A 23:G 22:r 21:M 20:L
19:s 18:E 17:O 16:I 15:R 14:x 13:M 12:X 11:E 10:Y
 9:E  8:g  7:b  6:L  5:r  4:q  3:P  2:l  1:l  0:E

1. Choose any two-digit number in the table above (e.g. 73).
2. Subtract its two digits from itself (e.g. 73 - 7 - 3 = 63)
3. Find this new number (e.g. 63) and remember the letter next to it.
4. Now press 'a' to analyze responses: a

Your letter is: E
```

Figure 2. Magic Letter Text Version

At [5] see videos of the regular and extra-credit versions of this program running, along with a sample assignment write-up. The extra-credit version introduces a delay between the user’s input and the program revealing the secret letter, supposedly using this time to “build a neural network”. In reality this additional time is unnecessary and is an example of misdirection that students are encouraged to incorporate into their programs.

2.2 Number Boxes

A second “Choose a Number” example involves inviting a student volunteer to imagine that after graduation they start a tech company and build it up over some years, finally going public and selling it to investors. The question is “At what age do you sell your company?” A similar guessing game using

number boxes has appeared in various places [6] [7]. Screen captures from an interactive example of this can be seen in *Fig. 3*:

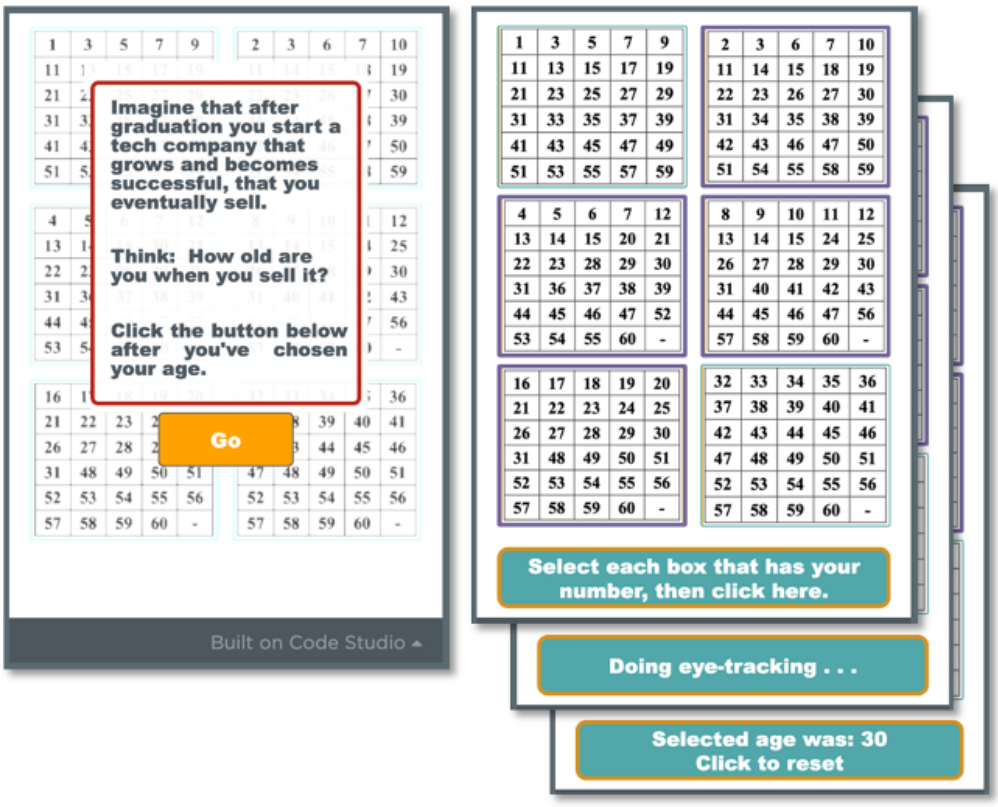


Figure 3: Number Boxes playable online at bit.ly/magicnumberboxes

In the above example the sequential screens in running the app are shown, starting from the left, where the user chooses a number. In this case let's assume the user thought of the secret number 30. On the next screen the user selects all the boxes that include the number 30, where selected boxes show up with a darker outline. To help in this selection, numbers in each box are displayed in order. Lastly, clicking on the button at the bottom leads to a message about "eye-tracking" and then displays the number the user was thinking of.

Like the previous *Magic Letter* example, this one can also be done statically using a print-out such as the six squares shown in *Fig. 4*. First a volunteer can be asked to choose the age after graduation at which they sell their company, writing it down on a hidden piece of paper. This narrative helps cast choices into the range of values less than 60. Then the presenter points to squares one at a time, successively asking "Is your number in this square?" Each time the volunteer says "yes" the presenter *remembers and sums the value in the upper-left corner* of that box, until all boxes have been considered. The order in which the boxes are considered doesn't matter.

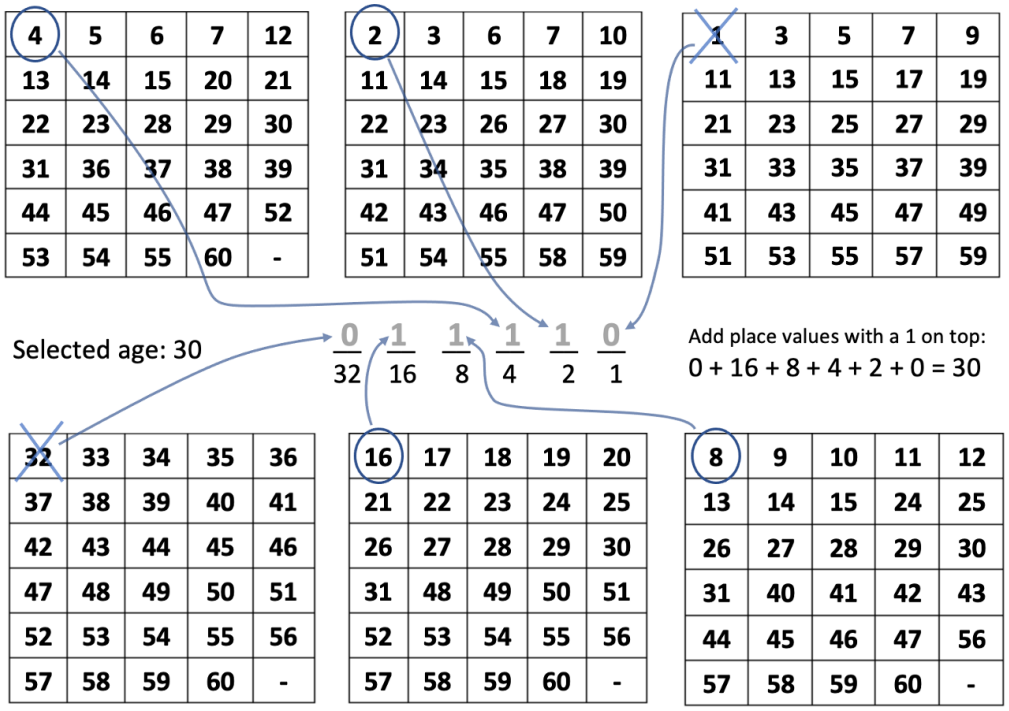


Figure 4: Number Boxes Process

In the example shown in *Fig. 4* let us suppose the volunteer chose 30 as their secret age. (Under normal circumstances only the six boxes with numbers are shown, since the other markings and values shown here in the center are for explanation purposes only.) The presenter first points to the box in the lower-left, asking "Is your number in this box?" and the volunteer would say "no", since 30 is not in that box. This "no" answer means the box upper-left corner value (32 in this case) can be ignored, representing a 0 in the 32's place in the binary number being constructed.

Next the presenter would point to the bottom-middle box, again asking "Is your number in this box?" and this time the volunteer would answer "yes", since 30 *is* in this box. The "yes" answer means this position value in the binary number, the 16's place, would be a 1, so now the presenter sums the box upper-left corner value (16 in this case), giving a total of 0 + 16 = 16 so far. Next up is the lower-right box, again eliciting a "yes" answer from the volunteer since 30 is again present, so the presenter now sums this box upper-left corner value (8 in this case) giving a total of 16 + 8 = 24 so far. This process continues, summing the upper-left corner values for all boxes where 30 is found, resulting in 16 + 8 + 4 + 2 = 30.

This problem is an excellent introduction to the idea of binary numbers. After doing the trick a couple of times to arouse students' curiosity, a more in depth explanation of binary numbers can be given, culminating at the end in the explanation of how to do the trick.

2.3 Resources And Variations

An important part of many magic tricks is misdirection, where extra information or actions divert the audience's attention away from what is really going on. In the *Number Boxes* example above, the presenter could pretend to ponder and then suggest that subconscious cues influenced the choice of the number, suggesting that "3" may be one of the values since there are 3 columns, and "0" may be another part of the answer since 0 rows have been skipped, resulting in the value "30".

If audience members start feeling this is a stall tactic and the trick isn't so hard if they have enough time to examine possible answers, then the magician can instead begin to give "lightning" fast answers. Such an approach can be especially impressive in a larger version of the trick in which audience members are asked to pick a secret number between 1 and 125. A static printout for this version can be found at rig.cs.luc.edu/~rig/1to125.pdf.

A variation of the *Number Boxes* example at cs4fn.org [8] has a combination of numbers and letters in each box. That version also explains how the numbers in each box are half of the overall set, namely those that have a 1 in a particular binary number place value.

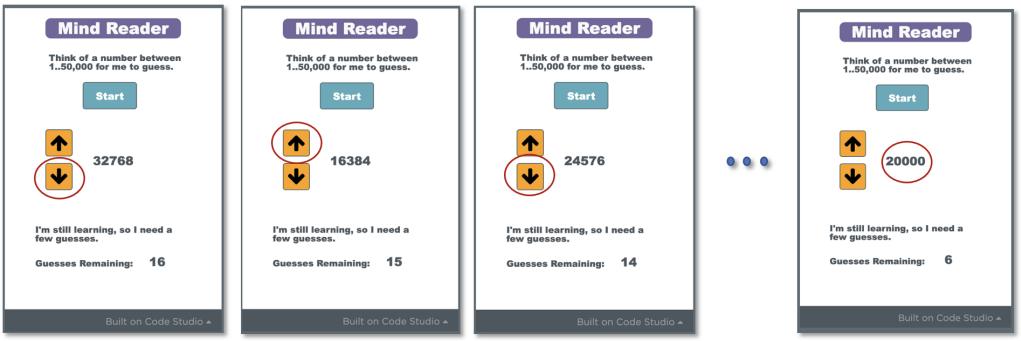


Figure 5: Guess a Number between 1 and 50,000 playable online at bit.ly/uicbinarysearch

We can use "guessing a secret number" as an introduction to binary search, as shown in *Fig. 5*. Here a volunteer selects some integer number between 0 and 50,000. Imagine they pick 20,000. The computer then makes up to 16 guesses, halving the search field on each guess by having the volunteer indicate if their secret number is "higher" or "lower" than the computer guess currently displayed. Again, this can easily be implemented as a text-based version. In our experience only a fraction of students immediately understand how this is working, so small group discussion helps *all* students understand, where the instructor can challenge students to consider what is the smallest possible problem they could use to explain the concept to each other.

Another properties-of-numbers example is called *Reversing Numbers*. Here the user is prompted for a 3 digit positive integer with unique digits. Subsequently subtracting and then summing the reverse of the digits always results in the same final value. To enhance this trick it helps for the presenter to write down the value 1089 on a folded slip of paper ahead of time. The presenter could then ask a volunteer to suggest a three digit number, asking that the digits all be distinct, to "make it as hard as possible." Imagine the volunteer chooses 457. The presenter reverses that number, giving 754, and does a subtraction starting with the largest of the two numbers, giving 754 - 457 = 297. The resulting number is again reversed, giving 792, and this time the two numbers are added, giving 297 + 792 = 1089. You then unfold the slip of paper, revealing this as the correct forecast. A description of this implemented as a text-based introductory programming project is given at [11]. A sample run of this program is shown below in *Fig. 6*.

```
Welcome to the number guessing game!
If you concentrate, sometimes you can connect to the electrons in the computer!
Let's try it.  Think of a three digit number. (To make it harder, make the digits
all different from each other). Type in your number: 275

I'll help you with the math.  Lets randomize those digits by reversing them, and do a subtraction:
   572  (The reversed digits)
 - 275  (The original number)
 =====
   297

Press 'Y' to continue or 'X' to exit... y

Now lets again scramble the numbers by reversing them, and adding them this time:
   297
 + 792
 =====
     ?

Before you continue, take a look at my number guess written down on paper.

Press 'D' to display the answer or 'X' to exit... d
Answer is 1089.
```

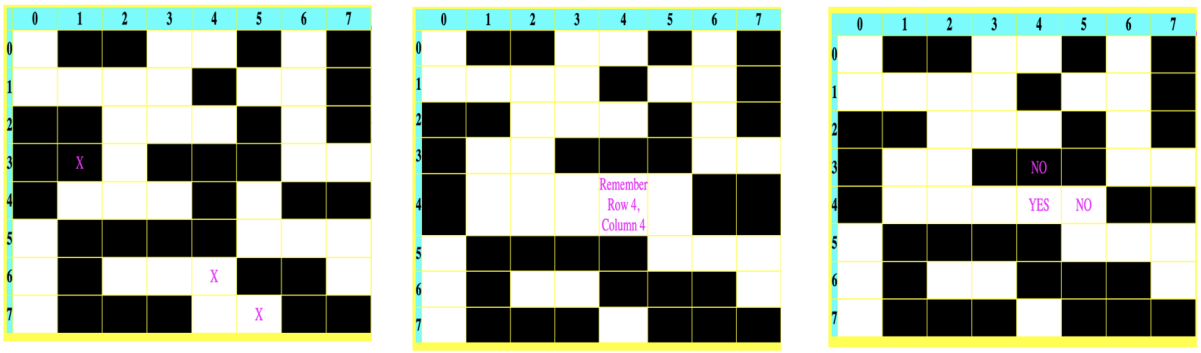
Figure 6: Reversing Numbers

3 PICK A CARD

"Pick a Card" tricks can be done using a deck of cards or can be simulated in a program using an array of tiles, where cards' front and back are represented by black or white tiles on the screen.

3.1 SECRET TILE

Many versions of the *Secret Tile* problem have been inspired by the excellent CS Unplugged error-detection card trick [10]. Successively complex versions of this idea developed by Greenberg [11] [12] are described and are playable online [13], where a volunteer selects a single square that the computer later can verify. Part of the sequence of stepping through this program is shown in *Fig. 7*.



a. Board created, Prompts for tile flips b. Volunteer selects secret tile c. Computer identifies secret tile

Figure 7: Pick a Secret Tile, playabler online as version 2a at bit.ly/tileflip

First (*Fig. 7a*) a randomized board is created with the help of a volunteer. The computer then identifies at most three tiles that "offend its sense of organization", and the user is prompted to click on them to flip them. Next (*Fig. 7b)* the presenter turns away from the screen, and the volunteer selects a secret tile. Finally (*Fig. 7c)* the computer gives feedback on whether or not the presenter was able to correctly select the secret tile. The underlying CS concept is the notion of *parity*, represented by whether a row or column has an even or odd number of black tiles. The computer-prompted tile flips are chosen to be the minimum changes necessary to ensure that the first seven rows have the same parity and the first seven columns have the same parity. Subsequent flipping of a single tile by a volunteer then breaks this pattern for a single row or must have occurred in the last row, and similarly for the columns. The intersection of the relevant row and column indicates the tile flipped. In *Fig. 7b*, each of the first seven rows had an *even* number of black tiles. Flipping the tile at row 4, column 4 then creates *odd* parity for only that row. Similarly, each of the first seven columns had an *odd* number of black tiles, and flipping the tile at row 4, column 4 creates *even* parity for only that column.

Versions of this "Pick a Card" program disguised as "Memory Trainer" apps can be played online either with a friend or solo, shown in *Fig. 8*. In both Memory Trainer versions the board is randomized and then the user tries to memorize what it looks like. The user then looks away while a partner flips a single square (on the left in *Fig. 8*) or the computer chooses one (on the right in *Fig. 8*). The user then has to figure out which square has been flipped. In reality it is possible to figure out which square has been flipped by noticing that the rows and columns have alternating even/odd parity. For instance in *Fig. 8 Memory Trainer Solo* the left (first) column has an *even* number of gray pieces (2), the second column has an *odd* number (3), the third again has an *even* number (4), and so on alternating between even and odd. The flipped square uniquely breaks the even-odd pattern for the row and column that intersect at the flipped piece.

This example can alternatively be implemented as a text-based program by clearing the screen and reprinting each new board displayed using 'X' and 'O' pieces as play progresses.

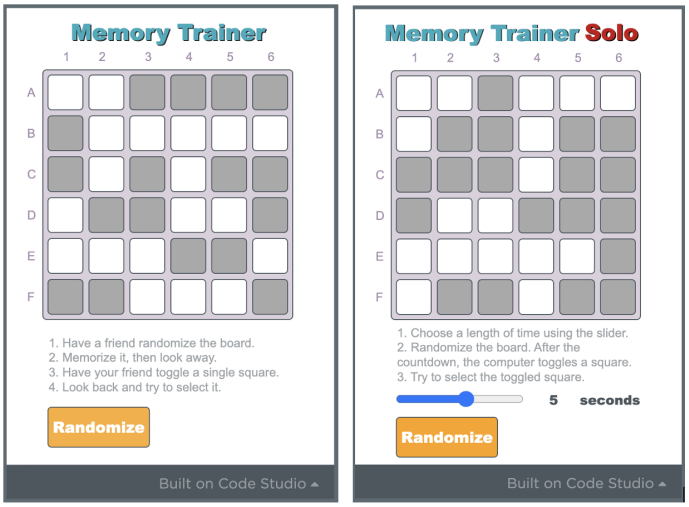


Figure 8: Memory Trainer playable versions online at bit.ly/cstamem

3.2 Twenty-One Card Magic

We can use a computer program to replicate what would normally be done using a deck of cards. Twenty-one cards are selected from a shuffled deck and dealt face-down into three stacks. The presenter asks the volunteer to select a stack and to look at those cards and choose one. The presenter then picks up all three stacks, *making sure to put the stack with the selected secret card in the middle* of the other two. Now, a second time, the pile is dealt into three stacks, left to right. Each stack is picked up and displayed to the volunteer, without the presenter seeing the card faces, and the volunteer this second time indicates which stack their card is in. The presenter again picks up all three stacks, putting the stack with the selected secret card in the middle of the other two. The process is repeated a third time, after which the user selected card is the 11th card in the deck, which the presenter can unveil with a flourish. Twenty-one Card Magic is described as a text-based assignment at bit.ly/cardtrick21.

4 RESOURCES

A treasure trove of CS-related tricks have been collected and described at cs4fn.org/magic. These include picking a card using mass hypnosis, using direction cards for a treasure hunt, pulling out cards to match a secret chosen card, and guessing how many cards a person has taken out of a card deck, and many mental math tricks. The site csunplugged.org/en/at-home connects CS concepts to mind-reading magic, guessing a number, binary search, and activities that have to do with information theory.

5 CONCLUSIONS

The *Number Boxes (Fig. 4)* and *Secret Tile* (*Fig. 7)* tricks were used in some of the outreach presentations [16] that reached several thousand students. Students generally expressed great fascination with these tricks and typically wanted them repeated. In surveys of over 200 students, 79% rated the “magic tricks” portion of these presentations as “Good” or “Very Good” (as opposed to “Poor” or “Fair”). Anecdotal evidence from the authors using magic tricks as programming assignments suggests students are motivated by this type of problem. They talk about them years later.

Computer programs that incorporate magic are both educational and fun, and liven up classroom discussions and assignments.

ACKNOWLEDGEMENTS

This work is supported in part by Break Through Tech (BTT) Chicago (chicago.breakthroughtech.org).

REFERENCES

[1] A. Clark, Profiles of the Future: An Inquiry into the Limits of the Possible. Orion Publishing, 1962. Reprinted 2000.

[2] P. Curzon and P. McOwan, "Engaging with Computer Science Through Magic Shows," in *ACM SIGCSE Bulletin*, vol. 40 no. 3, pp. 179–183, 2010. Retrieved from https://dl.acm.org/doi/10.1145/1597849.1384320

[3] J. Steinmeyer, *Hiding the Elephant*. Da Capo Press, 2004.

[4] Wilson J. González-Espada, Jennifer Birriel, and Ignacio Birriel. “Discrepant Events: A Challenge to Students' Intuition,” in The Physics Teacher 48, p. 508, 2010. Retrieved from https://doi.org/10.1119/1.3502499

[5] D. Reed, Assignment write-up for Magic Letter programming assignment. Accessed 5/4/2022. Retrieved from sites.google.com/view/bttcs111sp22/assignments/program-3-magic-square.

[6] W. Rapaport, Binary Magic. Accessed 5/4/22. Retrieved from http://www.cse.buffalo.edu/~rapaport/111F04/binarymagic.html

[7] Binary Magic, Accessed 5/4/22. Retrieved from http://www.mathmaniacs.org/lessons/01-binary/Magic_Trick/

[8] Letters and numbers variation of Binary Magic number boxes, Accessed 5/4/22. Retrieved from http://www.cs4fn.org/mentalism/cardsonyourmind.php

[9] D. Reed. Difference and sum of reversed digits gives 1089. Accessed 5/4/22. Retrieved from https://sites.google.com/site/cs141spring2017/programs/prog-1-guess-num

[10] CSUnplugged, Flip over a card trick using parity. Accessed 5/4/22. Retrieved from https://www.csunplugged.org/en/topics/error-detection-and-correction/integrations/quick-card-flip-magic/

[11] R. Greenberg, “Educational magic tricks based on error-detection schemes,” in *Proceedings of the 22nd Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pp. 170–175. ACM SIGCSE, 2017. https://doi.org/10.1145/3059009.3059034

[12] R. Greenberg and D. Reed, “Using Magic in Computing Education and Outreach,” in *Frontiers in Education* (FIE), San Jose, CA, 2018. https://doi.org/10.1109/FIE.2018.8658626

[13] R. Greenberg, Online playable versions of using parity to identify a user-selected card, Accessed 5/4/22. Retrieved from https://rig.cs.luc.edu/~rig/errdetectmagic https://doi.org/10.1145/3059009.3059034

[14] S. McGee, R. Greenberg, D. Reed, and J. Duck, “Evaluation of the IMPACTS Computer Science Presentations,” *The Journal for Computing Teachers*, vol. 26, pp. 26–40, 2013. International Society for Technology in Education.