COLOR BLOCKS

Varun Reddy[1], Nikhil Gabbeta[2], Sagar Naidu[3], Arun Reddy[4]

[1] *Btech, Computer Science and Engineering, SRM University, Tamilnadu, India*
[2] *Btech, Computer Science and Engineering, SRM University, Tamilnadu, India*
[3] *Btech, Computer Science and Engineering, SRM University, Tamilnadu, India*
[4] *Btech, Computer Science and Engineering, SRM University, Tamilnadu, India*

ABSTRACT

Smart Puzzle app is a classic sliding puzzle game based on Fifteen Puzzle. It allows you to choose among three levels of complexity which are: 3x3 (8- puzzle), 4x4 (15-puzzle), and 5x5 (24-puzzle).In addition to standard square boards with numbers, user can choose a board with a picture painted on it. There are three stock pictures installed with the puzzle, and user can add his/her own pictures by pressing Menu button when the picture selection page is displayed. The picture user select will be split into equal tiles. Once you select the picture, the solved puzzle is displayed for 3 seconds. Then it is randomly shuffled, and user have to move the tiles to their initial locations. When user solve the puzzle, the app will display that picture again along with the number of moves you have made. To change the puzzle's settings during the game, press Menu button. Note that when user change any settings, the game starts over. When user switch apps or exit during the game, your puzzle is stored and resumed next time user run Smart puzzle.

Keyword:-*Smart Puzzle; Color Blocks;*

1.intoduction

1.1 Overview

The COLOR BLOCKS is a sliding puzzle that consists of a frame of numbered square tiles in random order with one tile missing. The puzzle also exists in other sizes, particularly the smaller 8-puzzle. If the size is 3×3 tiles, the puzzle is called the 8-puzzle or 9-puzzle, and if 4×4 tiles, the puzzle is called the 15-puzzle or 16-puzzle named, respectively, for the number of tiles and the number of spaces. The object of the puzzle is to place the tiles in order by making sliding moves that use the empty space. The color blocks is a classical problem for modelling algorithms involving heuristics. Commonly used heuristics for this problem include counting the number of misplaced tiles and finding the sum of the taxicab distances between each block and its position in the goal configuration. Note that both are admissible, i.e. they never overestimate the number of moves left, which ensures optimality for certain search algorithms.Johnson and Story (1879) used a parity argument to show that half of the starting positions for the n-puzzle are impossible to resolve, no matter how many moves are made. This is done by considering a function of the tile configuration that is invariant under any valid move, and then using this to partition the space of all possible labelled states into two equivalence classes of reachable and unreachable states. The invariant is the parity of the permutation of all 16 squares plus the parity of the taxicab distance(number of rows plus number of columns) of the empty square from the lower right corner. This is an invariant because each move changes both the parity of the permutation and the parity of the taxicab distance. In particular, if the empty square is in the lower right corner then the puzzle is solvable if and only if the permutation of the remaining pieces is even. They also showed that the converse holds on boards of size m×n provided m and n are both at least 2: all even permutations are solvable. This is straightforward but a little messy to prove by induction on m and n starting with m=n=2. Archer (1999) gave another proof, based on defining equivalence classes via a Hamiltonian path. Wilson (1974) studied the analogue of the 15 puzzle on arbitrary finite connected and non-separable graphs. (A graph is called separable if removing a vertex increases the number of components.) He showed that, except for polygons, and one exceptional graph on 7 vertices, it is possible to obtain all permutations unless the graph is bipartite, in which case exactly the even 2 permutations can be obtained. The exceptional graph is a regular hexagon with one diagonal and a vertex at the centre added; only 1/6 of its permutations can be obtained. For larger versions of the n-puzzle, finding a solution is easy, but the problem of finding the shortest solution is NP-hard. It is also NP-hard to approximate the fewest slides within an additive constant, but there is a polynomial-time constant-factor approximation. For the 15- puzzle,

lengths of optimal solutions range from 0 to 80 single-tile moves (there are 17 configurations requiring 80 moves) or 43 multi-tile moves;[5] the 8-puzzle always can be solved in no more than 31 single-tile moves or 24 multi-tile moves (integer sequence A087725). The multi-tile metric counts subsequent moves of the empty tile in the same direction as one. The number of possible positions of the 24-puzzle is 25!/2 ≈ 7.76×1024 which is too many to calculate God's number. In 2011, a lower bound of 152 single-tile moves had been established; current established upper bound is 208 single-tile moves or 109 multi-tile moves. The transformations of the fifteen puzzle form a groupoid (not a group, as not all moves can be composed); this groupoid acts on configurations.

1.2 Objective

Puzzles are great for helping young brain develop and grow. That‘s because the brain looks for patterns in our world — and puzzles are a true patterning activity. Patterning is also the foundation of reading, math and logic skills. So when your child does puzzles, they are doing one of the best brain building exercises for developing their reading, math and logic skills. They also develop problem solving skills and individual success and achievement. That is why they like to do puzzles over and over again! What do you do when you are putting a puzzle together? Look for shapes that fit together, matching the pieces by colour or by image. Your brain stays focused on one single activity — it is hard to think of anything else except the pieces before you and how they fit together. I think that is why puzzling is so calming and relaxing — you are doing an activity that renews and refreshes your brain! Simply stated, your brain gets happy when you do puzzles.

PROBLEM SOLVING SKILLS: Ever listen to children‘s self-talk while they put puzzles together? They develop problem solving strategies for fitting the pieces together and completing the puzzle. They observe and detect similarities and differences, analyse, and do trial and error. (―Let‘s see, the blue piece goes with that blue piece — and they are the sky! ― ―This piece has a straight edge — it must be part of the border.

CHILDREN EXPERIENCE A SENSE OF COMPLETION AND MASTERY: A. Many pieces make a whole (also math concept) B. Fragments come together to complete an image C. Things that appear broken are fixed D. Things are put together where they belong

EXPERIENCE COMPETENCY, SUCCESS, ACHIEVEMENT Puzzles are often an individual activity. Puzzling requires a high degree of concentration for a period of time. Children become totally absorbed — and the success is theirs alone! It is a wonderful way for children to gain confidence in their growing abilities. That is why children love to do the same puzzle over and over again!

DEVELOP SOCIAL SKILLS: Puzzles can also be a small group (or family) activity. This promotes shared strategies, observation, and cooperation and a sense of shared achievement.

DEVELOP LITERACY SKILLS Talking about the process and also the puzzle picture gives excellent opportunities for conversation between child and parent, child and child. It also promotes ―self-talk‖ which helps children develop their own language skills. Puzzles are an excellent way to introduce new words and concepts in conversation

HOW TO SELECT A PUZZLE? There is no such thing as a ―too easy‖ puzzle. Kids often enjoy doing puzzles that seem ―easy‖ to adults — but the puzzle is just right for them. They will want to do the same puzzles over and over again. The important thing with puzzles is experiencing success and mastery in a fun, non-frustrating way. All puzzles develop patterning and problem solving skills and will develop your child‘s self-confidence. Puzzles also can promote conversation and once completed, can be used too as ―story starters.‖ Your child will move to harder puzzles when they are ready. Some children move quickly through puzzles others will take their time. Children should be encouraged to do puzzles at whatever level is fun, non-frustrating, and where they enjoy repeated success. Puzzles should 4 be designed so that children can advance through various stages of puzzle difficulty easily. Puzzles are such an excellent activity; a child should be encouraged to do the puzzles that are at the right skill level for them.

2.SYSTEM ANALYSIS AND DESIGN

Puzzles are ideal for players of all ages since they can keep the mind sharp and can be played for any level of difficulty. There are many different types of games people like to play. Some of the most popular are word, number and jigsaw puzzles. A favourite type of games of this genre is the n- puzzle. When you do these at home, you often have to take over the entire dining room table in order to have room to assemble the puzzle, and there is always the problem of lost pieces, but now there is an easier way to do color blocks puzzles from the

comfort of your own home without sacrificing your dinner table or searching high and low. Just come to this website and you will be able to search among a variety of n- puzzles for all ages and levels of difficulty. You can choose the dimensions and number of pieces as well as the design. This allows you to have full customization of your color blocks, all for free. You can browse through the categories which are comprised of such as art, architecture, and landscapes. Each category has several pictures in it for your gaming enjoyment. The application allows you to pick the number of pieces in the width and length. For those who wish to have a challenge, you can try the most difficult level of a 12 by 12 piece puzzle. The easiest size, perfect for children, is the 3 by 3piece color block puzzle. Much fun!

2.1 Existing System

Fifteen Version9.3 Fifteen - a puzzle, which is a 15 square tiles and bears the numbers from 1 to 15 All knuckles are enclosed in square of 4x4 (side of the square is four times longer than that of the knuckles). Thus when placing the tiles in a square is one empty space the size of a domino, which can be used to move the tiles. Goal of the game - rated accommodation properties, placing them in ascending from left to right and top to bottom, starting with the knuckles with the number 1 in the upper left corner and ending with an empty space in the lower right corner. You can select the background color of the game and knuckles. You can capture screen (screenshot) during the game and after the game. Play with your friends, relatives, colleagues, acquaintances, etc. Stir knuckles places and invite others to collect them. The classic game for those who want to pass the time and to train brains. Popular puzzle which Noah Chapman invented in 1878. It is necessary to move the chips with the numbers on the field, and arrange them in ascending order from left to right 6 8 puzzle is a smaller version of the 15 puzzle, just move the tiles horizontally and vertically and place them in the correct positions. If the shuffling of the pieces has finished, you can start with the solving of the sliding puzzle. Tap the tile closest to the empty space to slide it into the empty position. You can restart the slide puzzle at any time.

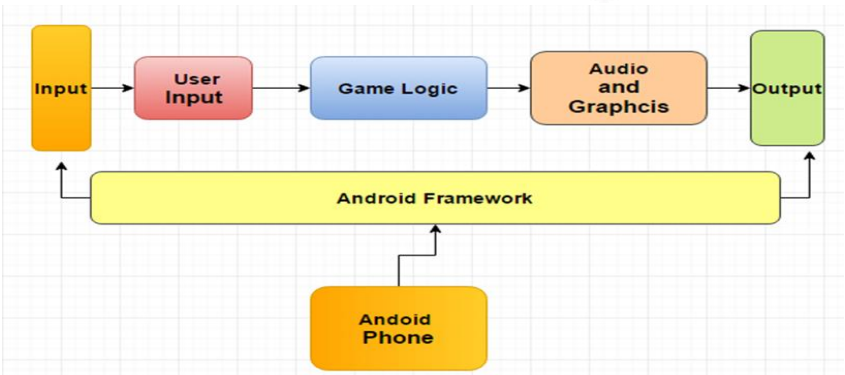
2.2 Issues in existing systems

1. If the user wants to move the tile or a block they have to slide it to the position
2. Only limited number of selections on puzzle exists
3. The puzzle doesn‘t have any pictures

3.System design

The project is proposed to design a game capable of supporting numbers, photos unlike existing system which supports only one format. the stop watch is removed from the proposed system which allows user to play freely as he/she likes. Our game‘s reshuffle works on totally random algorithms which are picked by system itself. The game consists of 3 difficulty levels i.e., easy , medium and hard which can be selected by the user itself. The easy mode consist of 8 tiles (3*3 puzzle), the medium mode consists of 15 tiles(4*4 puzzle),the hard mode consists of 24 tiles (5*5 puzzle) . In addition to standard square boards with numbers, you can choose a board with a picture painted on it. There are three stock pictures installed with the puzzle, and you can add your own pictures by pressing Menu button when the picture selection page is displayed. The picture you select will be split into equal tiles.

3.1 **System Architecture**



3.1.1 Input

In computer science, the general meaning of input is to provide or give something to the computer, in other words the state/act of a computer, component of a computer or relevant device being accepting something from the user, from a device or from a piece of software either automatically or manually is called input. We categorize computer devices as input devices because we use these devices to send instructions to the computer, we are sending our "Input" to the computer, some common examples of computer input devices are:

• Mouse • Keyboard • Touchscreen • Microphone • Webcam

We may also call some inner parts of the computer as input components to the other components, like the power-on button of a computer is an input component for the processor or the power supply, because it takes user input and sends it to other components for further processing. In many computer languages the keyword "input" is used as a special keyword or function, such as in Visual Basic or Python, the word "input" is used to get text input from the user. It is defined as what is put in, taken in, or operated on by any process or system, It is a place where, or information enters a system. In the game, the input is about taking commands about selecting game, choosing difficulty and making moves of tiles. This provides the system with a referrals to work with and to start a process given to system by user

3.1.2 User input User Input is one of the most important aspects of programming concepts. Every program should have some sort of user interaction, from getting a character's name for a game to asking for a password to log into a database. This article will teach the basics of user input in the BASIC Programming Language. Please note that the following code may vary from compiler to compiler. User input can be used for almost anything in your game. Game developers are always trying to find new and innovative ways to get the player involved in the game. A list of the more common uses for input are listed as follows

• Player movement • Camera movement • Menu selection • Actions • Attacking • Puzzles

The uses for input are infinite and finding a new use for input could create a brand new game mechanic.

3.1.3 Game logic

The (n -1) puzzle is traditionally represented as a 4 * 4 board with tiles numbered from 1 to 15 arranged in numerical order from top left to bottom right of the board (called the goal board). However, in this report, it is represented as an n * n board with 2 tiles numbered from 0 to (n -2) as each row and column is indexed from 0 (Figure 1.1). The remaining tile (bottom right) remains blank as it is used to move the other tiles around the board (called the space). The puzzle is created by re-arranging the tiles on the board (called the start board) and attempting to solve it by recreating the goal board. This is achieved by moving the tiles back into their goal board positions by moving the space around the board

The n=4 version of the puzzle, the Fifteen Puzzle was invented by Sam Loyd in the late 1870‘s (also called the Boss Puzzle) [Gardner 1961]. It was marketed as a tray with 15 numbered tiles and became very popular with the general public due to the difficulty of generating a solution. However, the puzzle was sold in such a way so that no solution could be derived from the start configuration, no matter what technique was applied. The best that could be achieved would be the tiles arranged in order in the tray, with only the tiles 13 and 14 inverted .

A configuration of the (n -1) puzzle board can be represented as a permutation of the 222 set {0,1,...,n -2,n -1}, where (n -1) is used to represent the space (called the start state). As expressed by [Storey 1879], the start state can only be solved if it is an even permutation of the goal state (the permutation to represent the goal board). An even permutation is a configuration such that it can be converted to the goal state by an even number of transpositions or interchanges. In the puzzle solution, all interchanges involve the space only – the solution path is the path the space takes from the start state to the goal state. Odd permutations cannot be solved as it is impossible to convert them to the goal state by any number of interchanges – an attempt to solve an odd board results in achieving a board similar to Sam Loyd‘s in Figure 1.2, with the 22 n -3 and n -2 tiles inverted. An algorithm to check if a board configuration is an even permutation will now be given.

Even Permutations

As expressed earlier, the initial configuration of the board (start state) can only be solved if it is an even permutation of the goal state. An even permutation can be determined by placing the goal state underneath the

start state and calculating the cyclic interchange of numbers between the two states. This idea is now presented in more detail.

Determining a state's evenness

There are two cases to consider when determining whether a start state is even or not: 21/ The number n -1 (space) is the final element in the start state i.e. already in the same position as it is in the goal state. 2 2/ The number n -1 is not the final element in the start state.

Space in its default position

In the first case, the space (n -1) in the start state is in the same location as its default location in the goal state. Figure 1.3 shows an example of such a board state. Adopting a functional programming approach [Bird/Wadler 1988], an algorithm to determine the cyclic interchange of numbers between the start and goal states can be defined as follows:

Even :: Board Board Int Int Int Int

Even startstate goalstate startnum numscheckedsofar numincycle totalinterchanges

| (numscheckedsofar == n^2) = return totalinterchanges

| (startnum has already been checked) =

Even startstate goalstate (numscheckedsofar+1) (numscheckedsofar+1) 1 totalinterchanges

| otherwise = check number in startnum's startstate location in goalstate (called goalnum), if (goalnum == numscheckedsofar) then

Even startstate goalstate (numscheckedsofar+1) (numscheckedsofar+1) 1 (totalinterchanges+numincycle-1)

else Even startstate goalstate goalnum numscheckedsofar (numincycle+1)

totalinterchanges

startnum represents the current number in a cycle being checked, numscheckedsofar represents the current number in startstate being checked, numincycle keeps a track of how many numbers have appeared in a cycle, while totalinterchanges is the total number of cyclic interchanges between the two states.

To demonstrate how the function works, Figure 1.4 shows the example board of Figure 1.3 (startstate) placed above the goal state (goalstate) of an n=3 board.

406517328 012345678

By following the steps defined above, the startstate of Figure 1.3 is checked for evenness. At the outset of the function, startnum = 0, totalinterchanges = 0, and numincycle = 1. Firstly, startnum is found in the startstate and the number below it in the goalstate (goalnum), 1, is noted. startnum becomes 1 in the startstate and the number below it in the goalstate, 4 is noted. This process is repeated for the number 4, but since the number below it in the goalstate is a 0, which is the first number in the cycle (numscheckedsofar), the cycle has been completed. Hence, this cycleconsists of the following numbers: 0,1,4,0 - the number of total cyclic interchanges so far (totalinterchanges) is (numincycle - 1) = 2.

The next number to be found is 1, but since it has already appeared in a cycle, there is no need to check it. 2 is then checked, and the following cycle is detected: 2,7,5,3,6,2. This process is repeated for the remaining numbers 3 to 8, the number denoting the space. As a result, the following cycles are detected

(0 1 4) (2 7 5 3 6) (8)

Hence, the first cycle contains 3 numbers i.e. 2 interchanges, the second cycle contains 5 numbers i.e. 4 interchanges and the third cycle contains 1 number i.e. no interchanges. By adding the total number of interchanges, it is found that there are a total of 6 interchanges in the start state. Since 6 is an even number, it can be deduced that the start state [4, 0, 6, 5, 1, 7, 3, 2, 8] is even and therefore a path from this state to the goal state [0, 1, 2, 3, 4, 5, 6, 7, 8] can be determined. By applying this method to all board states of the numbers 0,

1,.., n -1, it can be shown that there are (n2 −1)! even permutations when n2 -1 is in the same position in the 2 start and goal state.

Space not in its default position

The first case takes into consideration only those start states with the space in its default location (same position as in the goal state). However, when it is not in this position, its evenness depends on the Manhattan distance the space is away from its default location.

As defined in [Parberry 1997], the Manhattan distance of a number is the minimum amount of moves needed to move that number into its position in the goal state. By adding the number of moves required to move the space (n -1) into its default location to the number of interchanges inthe start state, it can be determined if this state is odd or even. Once again, the start state is placed underneath the goal state. However as seen above, this time n -1 (8) is not in its default location.

014867523 012345678

By following the algorithm presented in 1.2.1.1 for when n -1 is in its default location, the following cycles can be detected :

(0) (1) (2 7 5 6 4) (3 8)

The first and second cycles contain 1 number i.e. no interchanges, the third cycle contains 5 numbers i.e. 4 interchanges and the fourth cycle contains 2 numbers i.e. 1 interchange. By adding the total number of interchanges, in this case there is a total number of 5 interchanges in this start state. Since 5 is an odd number, it would seem that the permutation [0, 1, 4, 8, 6, 7, 5, 2, 3] is odd and therefore cannot be solved. However, since n -1 (8) is not in its default location, it has to be ascertained how many moves are needed to move it into its goal location.

By using the grid in Figure 1.9, which represents the Manhattan distance of n -1 from its position in the start state to its default location, this number of moves can be

If the start state [0, 1, 4, 8, 6, 7, 5, 2, 3] is represented in 3 * 3 grid form, then from the Manhattan distance table it can be seen that it takes 3 moves to move the space (n -1) to its default location (represented by 0 in the table). If this number is added to the number of interchanges in this permutation, this gives a total of 8 interchanges, which is even. Hence this is an even permutation.

A fast lookup table is used to determine the Manhattan distance table for a board of dimension n (Equation 1.1):

Equation 1.1:

22 [[((n-1) , (n-1) -1,..,(n-1)],

22 [(n-1) -1, (n-1) -1,..,(n-2)] , ...

[(n-1),(n-2),..,0]]

2 By following this logic, it can be shown that there are n !even permutations of the numbers (0, 1,.., n -1), which is the number of valid start states in the puzzle from which a path to the goal state can be obtained.

4.Conclusion

Through generating this Sudoku solver and generator I feel I have improved my programming ability. This was perhaps the largest program in terms of time invested and lines of code written that I have created. The code is not of the highest quality, and it is severely lacking in documentation, but some of the problems posed by the project were a good challenge to solve. Writing the solver has demonstrated the advantages of ‗s mart' algorithms over naïve algorithms evidenced in the measurements above. Finally, I have experienced participating in what could be called a small research project (useful for future work). Project Status The logic solver portion of the program is sufficient for solving many n- puzzles. However, the implementation could likely be improved to execute faster. Furthermore, there are many logic solving techniques that have not been implemented. The generator, on the other hand, is currently rather poorly implemented. It works in that it successfully generates grids of a given number of hints, but the variability of the time taken to generate is a

significant weakness. Additionally, the generator currently only offers two levels of puzzle difficulty (_easy‘ and _hard‘) a scale of difficulties would be perhaps s more useful.

Future Enhancement

Anyone who is watching the progress of technology must be looking forward to when the leap is made to combining computer gaming technology with board gaming. All it takes is to start using large touch-screen computers as the actual board. Imagine gaming lounges with huge iPad-type devices running all manner of games you can imagine. No need for separate pieces or decks of cards, and no chance of your game being lost if it gets knocked over! Board games gave birth to computer games, and now I think the final outcome of all this is a fusing of the two forms to give endless gaming possibilities. Those who are concerned that the coming technology revolution is going to see them with a whole lot of leisure time should 40 perhaps take some comfort in the fact that there are going to be more ways to enjoyably spend your leisure time than at any other time in history Of course, these new platforms are going to need new players and new designers, and with the scope of new technology available, there should be no reason that everyone can‘t start taking part in the gaming revolution. Since gamers also generate huge amounts of data whenever they play, there will be more than enough to keep the statisticians and game theorists busy for many generations to come.

5.Reference

[1] D. Moore, C. Shannon, D.J. Brown, G.M. Voelker, and S. Savage, ―Inferring Internet Denial-of-Service Activity,‖ ACM Trans. Computer Systems, vol. 24, no. 2, pp. 115-139, May 2006.

[2] A. Hussain, J. Heidemann, and C. Papadopoulos, ―A Framework for Classifying Denial of Service Attacks,‖ Proc. ACM SIGCOMM ‘03, pp. 99-110, 2003.

[3] A.R. Sharafat and M.S. Fallah, ―A Framework for the Analysis of Denial of Service Attacks,‖ The Computer J., vol. 47, no. 2, pp. 179-192, Mar. 2004.

[4] C.L. Schuba, I.V. Krsul, M.G. Kuhn, E.H. Spafford, A. Sundaram, and D. Zamboni, ―Analysis of a Denial of Service Attack on TCP,‖ Proc. 18th IEEE Symp. Security and Privacy, pp. 208-223, 1997