5-2011

# An introductory educational board game for use in early computer science education

Tyler Moore
*University of Arkansas, Fayetteville*

# An Introductory Educational Board Game for Use in Early Computer Science Education

An Honors Thesis submitted in partial fulfillment of the requirements of Honors Studies in Computer Science

By

Tyler Moore

2011
Computer Science
College of Engineering
**The University of Arkansas**

## Table of Contents

## List of Illustrations

## Abstract

Early computer science education should be necessary in high school curricula, but often it becomes inextricably linked to the act of programming instead of the study of the principles of computation. In order to divest computer science from programming a new teaching medium is needed, and early research into games as teaching tools shows some positive results when used properly. In order to find a better way to teach early computer science concepts I have designed and implemented a board game which illustrates and defines a few necessary computer science terms and mechanics. I had reasonable success in the classroom, with mixed results from two completely different groups of students. The game seems effective, but my methods of teaching and lesson plan surrounding the game weakened the gains I recorded. I plan on reworking the base rules and developing new expansions which would increase the playability of the game and  simplify the methods for delivery of the computer science material covered during a game session.

## I Introduction

Outside of the classroom, games play an important social and cultural role in the lives of adolescents because games can effectively gain and hold the attention of their target audience for extended periods of time. As a user spends more time with the game he or she becomes more familiar with the structure and nature of the game, and this familiarity engenders a greater desire to explore the game. This feedback loop, if tapped effectively, can be used to present useful and educational material to the user in a positively biased way. Unfortunately, this loop is rarely tapped effectively and exploration into games as teaching tools seems to be very modest. Therefore, I endeavor to present a game designed for table-top play which can be used inside and outside the classroom to impart educational materials to audiences ranging from adolescent to young adult in a fun and engaging way.

It is patently clear that it has become necessary to involve young children in computer science education. Educators have already noticed a need to increase exposure to computers in general schooling as computing becomes more pervasive. Greater steps are necessary, though, to engender students with cross-disciplinary problem-solving and logical thinking skills, skills which can be developed through the study of the principles of computation. Unfortunately, much of computer science can be difficult to grasp without a significant background in math, logic, and language studies. In the case of the young student, simply introducing rules-based games which encourage one to refine a game-playing strategy can be an easy stepping stone into a discussion of computation. Marking successful exposure could be as easy as placing the child in a situation where the child presents himself or herself the question: "How would a computer think about this problem?" For the older audience that has at least some knowledge of math and language studies, using a game which explores integral computer science concepts while demonstrating them at their most basic level can ease the transition. Often, non-technical students shy away from introductory courses in computer science because they force the student to learn a skill which he or she feels is unnecessary, i.e. programming. This hypothesis has been examined to good result at Carnegie Mellon. Instead of using traditional introductory methods, researchers used Raptor, a flowchart simulation tool, to steer the course away from hard programming and towards the more ephemeral study of computation principles. [1] It is very interesting to note that Carnegie Mellon researchers noticed a marked increase in enrollment in this course due mainly to word-of-mouth advertising. [1] Non-traditional teaching methods clearly have a place in the early curricula.

By framing the study of computation in the form of a game, the student can aggregate a similar foundation skill set without the presence of a demotivator like the study of programming. Students will respond more positively to the game than the traditional coursework because the game is far more accessible a medium than C-based programming languages. Also, students are more likely to have prior experience with board games and digital games, especially non-technical students, than with Pascal or Java. Also, games use universal rewards as motivators and administer them on a short timescale which generates more interest for the material in the context of the game than if the context were a programming language. Because a game is more accessible, has wider appeal, and has a more gentle learning curve than traditional teaching methods in early computer science courses, I chose to design a board game which could be used in the classroom to introduce basic and intermediate computer science concepts to students interested in exploring principles of computation.

I have chosen to design a board game because board games are novel and accessible, requiring little set-up to deploy in a traditional classroom setting. A physical board game can be used in the classroom to generate initial interest in the material with little expenditure of class time and teaching effort. The novelty of the board game serves to draw in non-technical players as well as ingratiating players which would otherwise be demotivated by the presence of a computer.

The game draws heavily from the game-like computer science concept of a busy beaver developed by Tibor Rado. [5] Stated simply, a busy beaver is a Turing machine with n+1 states, where one state is a Halt state and n is a positive integer, that writes the greatest number of ones when compared to busy beavers with an equivalent number of states to its tape before halting. In the game I have designed each player operates a read/write head on a pseudo-infinite tape, that is a simple closed loop around the game board's edge, and endeavors to write the most number of ones to the tape by the end of the game. Each player is given a number of resources which he expends slowly during gameplay and which he may sometimes recover depending on the actions he takes. The first player to exhaust the arbitrarily selected number of his resources has entered his Halt state and ends the game. A player's turn consists of writing some symbol to the tape, either a zero or a one, then a transition consisting of moving the read/write head one, two, or three spaces, and then reading from the tape. Depending on what symbol is read, some action is performed by the player and then the next player's turn begins. At it's simplest level, the game plays like Parcheesi, perennially a favorite board game among children. If the player wishes, he may delve deeper into the nature of computability and the Turing machine by using a more complex rules set. The game touches on other computer science concepts like looping and algorithm design and implementation, and very lightly on conditional statements as well.

## II Previous Work

The use of games in education is a topic currently under exploration. Many understand the gains a child can receive in the classroom when the material is presented in an innovative fashion, particularly when it is structured as a competitive game. There are a few elements which have been found to promote student involvement when adopted: a well-defined system of rules, clear but challenging goals, the attachment of fantasy to the student activity, progressive difficulty levels, interaction with a high degree of student control, uncertain outcomes, and immediate and constructive feedback. [3] When these axioms were followed in a non-violent and non-gender-specific game designed to educate, students proved to be more attentive during the instruction time and scored better on tests designed to evaluate how well a student has learned the material. [2] Also, both boys and girls perform equally well in an educational environment focused on learning through games, even when the material covers concepts which are traditionally gender biased, such as math or computers. [2] This is an unexpected benefit. It is also interesting to note that students involved in education through games report more positive views on the learning experience than those which were exposed to the material in a more traditional manner. It seems the only pitfall is that students expect the same striking qualities from educational games as are exhibited in games played for fun on their own time, i.e. rich visual environments, well-developed plot, etc. I believe that the scope of the game can inhibit this final perceived flaw in educational games. If the game is designed to be used as an introduction to a course, a supplemental piece of the whole, then the stress to produce a rich, interactive experience can be lessened without diminishing the gains in motivation and gender equality which game environments provide.

Some attempts have been made to introduce course environments which negate the overhead that the C family of languages carries. Currently, early computer science courses tend to focus on language study paired with the study of basic data structures and algorithms. The student first learns a programming language, and then learns how to express fundamental ideas of computer science in that language. There are some who believe that this overhead is detrimental to the student and the school environment in terms of retention and motivation. Students outside of the traditional computer science programs often find the thought of programming distasteful, so they shy away from computer science introductory courses. Professors at Carnegie Mellon University have tried to eliminate this most distasteful part of the introductory course by focusing on the computer science and excising the language study. Carnegie Mellon's notable attempt to introduce computer science in a palatable manner for non-majors focused on the principles of computation through a program called Raptor. [1]

Raptor's objective seems to be solely to enhance the material, not to take class time away from the goal of the course just to learn the syntax and semantics of a full-featured programming language. The researchers note that little additional time was spent learning the tool after a comparatively brief tutorial. Another attempt at Carnegie Mellon includes the adoption of Alice in early computer science courses, a language which abstracts much of the obfuscating syntax and simply allows a user to semantically define a "program" using logical animations and on-screen tools. [4] I believe these alternate avenues of early computer science education succeed because the language of the course, the basic educational units, so to speak, is designed to enhance the material, not to detract from it or to confuse the student as it seems full-featured programming languages often do.

## III The Game

The design of the game focuses on a number of computer science concepts. Players become familiar with basic algorithm design as they are exposed to the game. Each player must focus on a strategy and implement that strategy during play in order to win. This decision-making process is tied closely to basic algorithm design and selection. Through playing the game, students will naturally begin to ask the question: is it more efficacious to execute one series of instructions or another? Players also encounter basic data structures like stacks and queues. More advanced students can look past the playing of the game and examine the concepts behind its build, concepts like the Turing machine and the existence of the Busy Beaver class of Turing machines. This examination culminates in the addressing of basic concepts of computability and some basic questions of computer science, like "What makes a computer a computer?"

These concepts are implemented purposefully throughout the game, some bluntly teaching students about computer science, and some obliquely introducing the concepts. Because the goal is very clearly defined, students have to use critical thinking skills to examine the current position of the board relative to the position of the board in the case of winning play. They must make the board more and more like their winning case using a very strict turn structure and order. This critical thinking and strategy very closely approximates the computer scientist's early struggle to solve a problem. Given a small set of tools and a particular set of known quantities, evaluate the initial position and design a path to an acceptable final position which will render the desired solution. During gameplay, players organize cards on the board using basic data structures like stacks and queues. Cards placed in a stack fashion are readily available to the next player that visits that stack, but cards placed in a queue fashion may not be available for some time. This physical arrangement can help early students more readily visualize these basic data structures when applications are called for in the future. Because each player controls the read/write head of a Turing machine, it is an easy step to show the student the function of a transition table, the function which they provide in the game. Each turn, the player must decide what transition to make and what symbol to write, essentially creating a transition table on-the-fly for their Turing machine. More advanced players could even write down a few transitions at the beginning of the game and simply follow their tables instead of making the decision as the game develops.

Here is a complete transcription of the rules set at the time of the public playtesting session:

**How to win:** Be the player with the most ones pawns on the board when the game ends.

**How to end the game:** Exhausting your ones pawns, or attempting to read a zero card from a space without zero cards gives each other player one more turn before the game ends.

**How to start the game:**

1. Deal a number of zero cards to each space based on the length of the game desired, reserving a minimum of 12 cards.
2. Deal three zero cards to each player from the reserve, face down.
3. Randomly determine which player is the starting player.
4. Each player chooses a starting vertex, beginning with the first player.
5. The first player begins his first turn and play proceeds clockwise.

**Diagram of a player's turn:**

- Phase 1, the Write phase: Write output to the tape
- Phase 2, the Transition phase: Transition on the tape
- Phase 3, the Read phase: Read input from the tape

**Writing:**

- If the player wishes to write a one, he must replace the pawn, if any, currently on the space with one of his own color.
- If the player wishes to write a zero, she must remove the pawn currently on the space, if any, and then that player executes the text of a zero card in her hand. Finally, the player places the chosen zero card from her hand onto the space. She will place it on top of or underneath the pile on the space depending on the text of the card.
- If the player wishes to write a blank, he places nothing on the space and may change his direction on the tape.

**Reading:**

- If the input read is a one, draw and execute a chance card.
- If the input read is a zero, draw the top zero card from the pile on your current space.

**Transitioning:**

- On a player's turn he may choose to move 1, 2, or 3 spaces in his current direction.

**Zero Cards:**

Zero cards, in general, employ innocuous individual effects which may or may not be helpful in a given situation. A couple of typical cards might look like:

- "Move 10 spaces in your current direction."
- "Read the top zero card from your current space."
- "Place a pawn of your color on your current space."

Zero cards may help you end the game more quickly by exhausting resources, or may help you extend the game by conserving your available resources. These effects are varied among the designed zero cards.

**Chance Cards:**

Chance cards, in general, fall into 1 of 5 categories: Global Positive, Global Negative, Individual Positive, Individual Negative, and Neutral. Chance cards affect a larger portion of the game and are generally stronger in effect than zero cards. Global Positive effects generally assist all players in a positive manner, by either giving them a choice of effects or by giving a universally good effect. A Global Positive chance card could look like the following:

- "All players may place a pawn of their color on their current space, or a space adjacent to their current space."
- "All players may write a zero card from their hand. If a player chooses not to, he may read a zero card from his current space instead."

Global Negative cards will generally foul players' plans by forcing the return of resources or by unexpectedly shifting the state of the board. A Global Negative chance card could look like the following.

- "All players must recover a pawn of their color from the board."
- "All players must place a zero card from their hand in stack fashion onto their current space. Any player with no zero cards in their hand must read a zero card from their current space."

Individual Positive cards will generally either foul an opposing player's plans or help the current player expend or regain resources. An Individual Positive card could look like the following:

- "Place a pawn of your color on your current space, or force an opponent to remove a pawn of their color from play."
- "Read the top zero card from the pile on your current space, then you may write a zero card to your current space (executing the text as usual)."

Individual Negative cards will generally help an opposing player or foul your current board position. An Individual Negative card could look like the following:

- "Select an opponent with the least number of pawns in play. That opponent may place a pawn on their current space."
- "Recover a pawn of your color from play. Select a zero card from your hand and write it to your current space."

Neutral cards will generally perform actions which are neither overtly positive or negative, but affect all players equally. A Neutral card could look like the following:

- "Move each Read/Write head 5 spaces in its current direction."
- "Starting with the active player, each player reads a zero card from their current space and then writes a zero card to their current space."

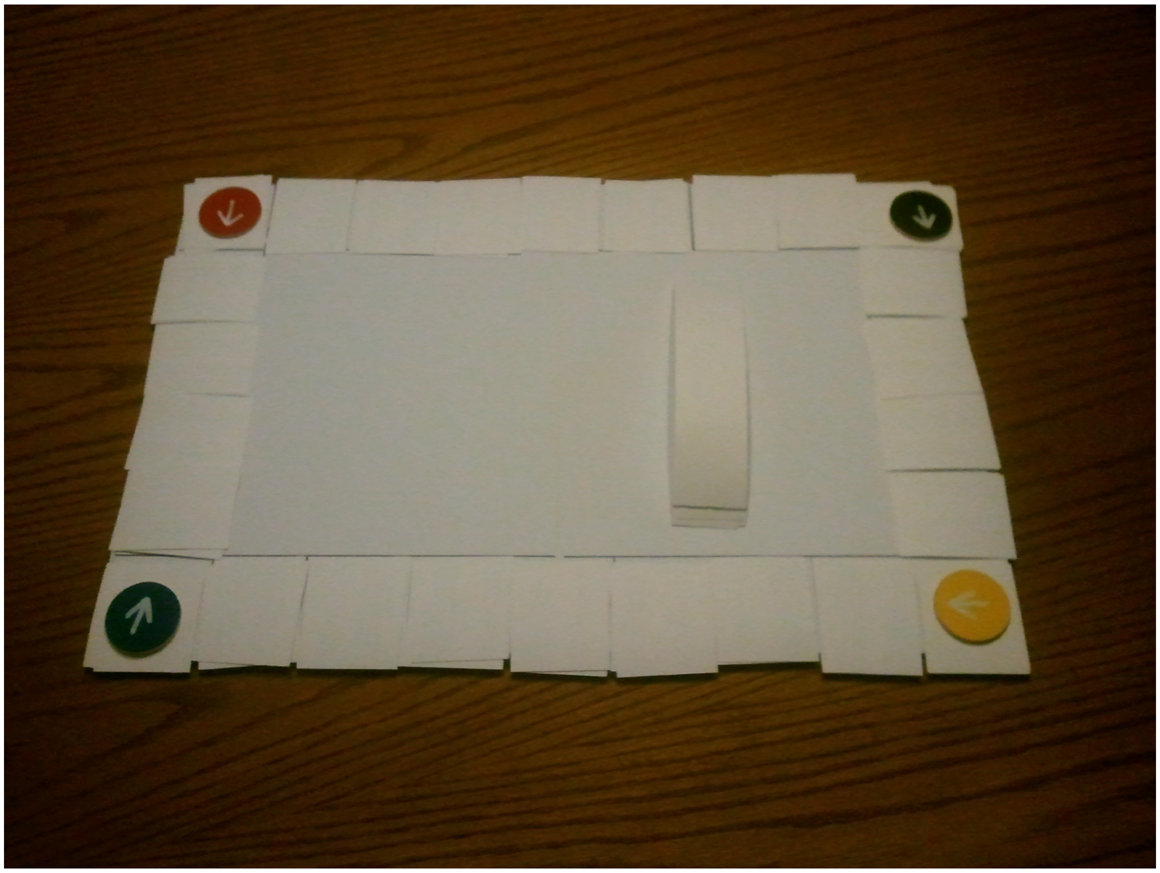


*Illustration 1: A game board after set-up*

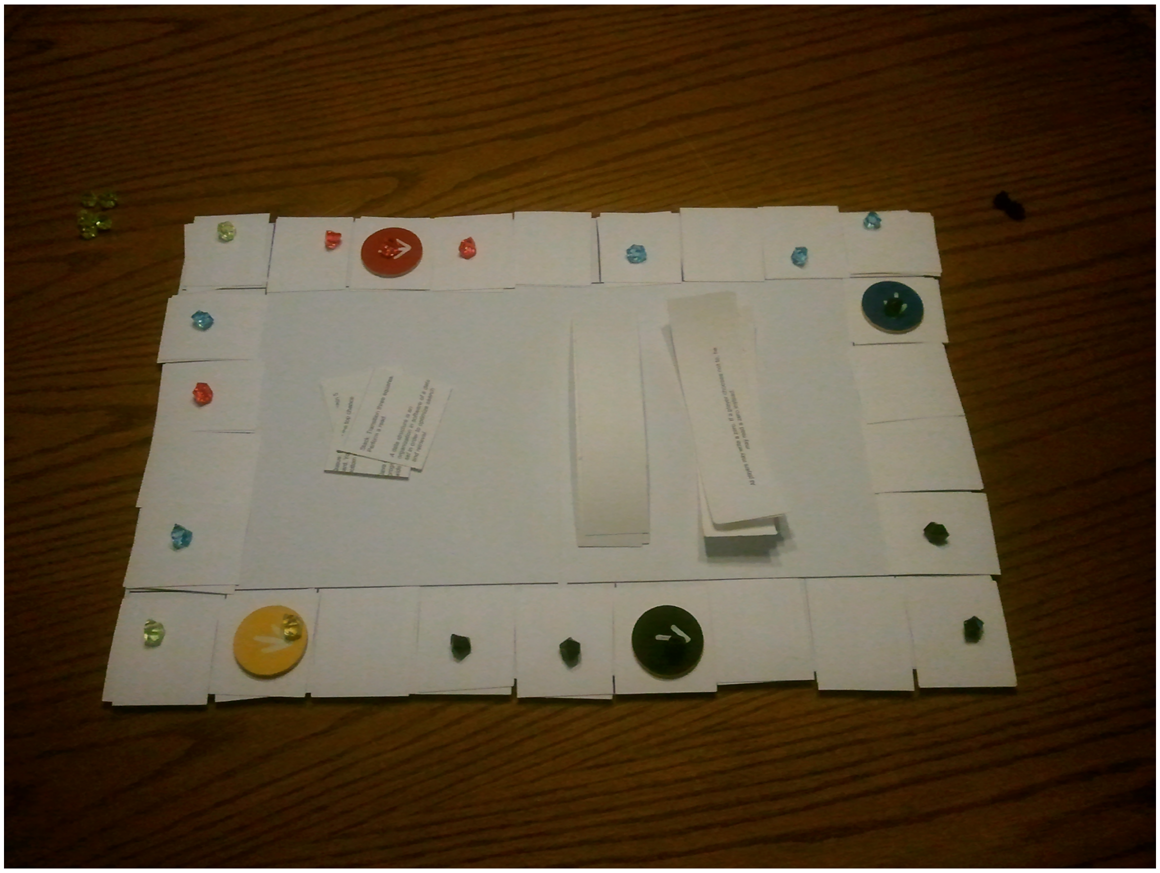
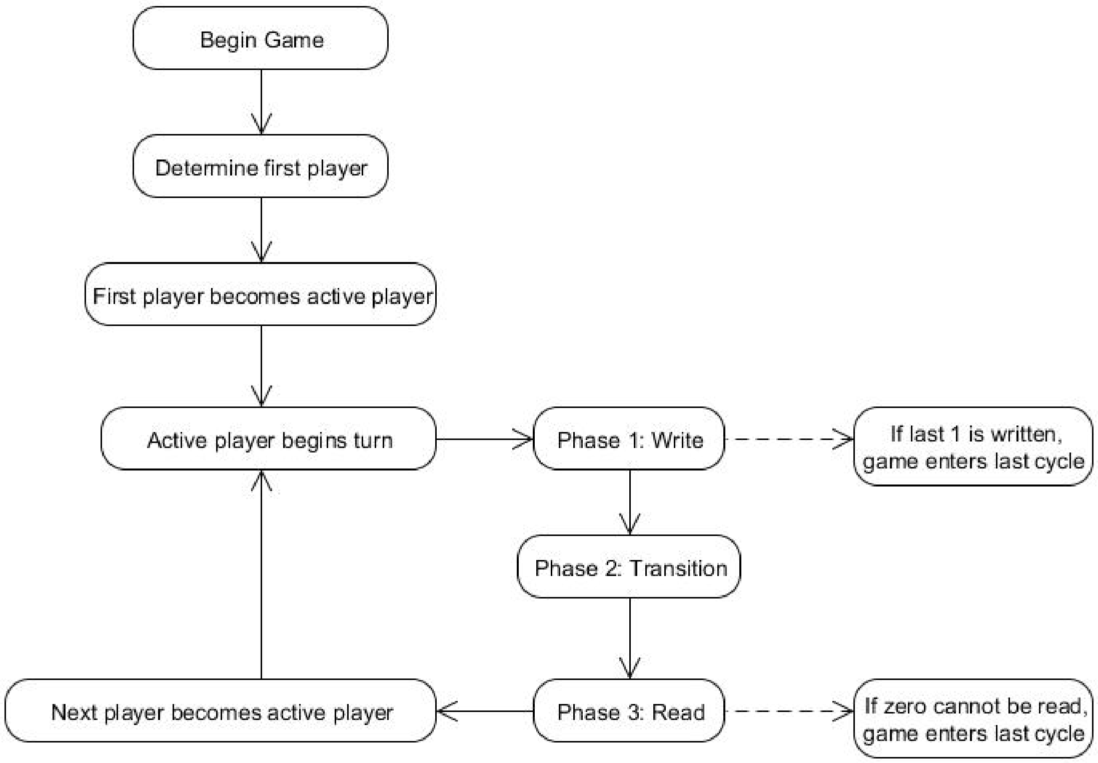


*Illustration 2: A game board during play*


Begin Game
Determine first player
First player becomes active player
Active player begins turn
Phase 1: Write
If last 1 is written, game enters last cycle
Phase 2: Transition
Phase 3: Read
If zero cannot be read, game enters last cycle
Next player becomes active player
*Illustration 3: A basic flow of gameplay*

## IV Discussion

The goal of any educational effort is to engage its target audience and to produce a real and measurable difference in that audience. Traditional education methods are effective, but succumb to known pitfalls again and again. Students aren't motivated to learn the material or the structure of the course of learning isn't intriguing and fails to capture the imagination of the student. These two problems stand out among a countless sea of others. Research into using games as teaching tools has revealed that games close the gap between the genders, even in hard science and math courses, courses traditionally dominated by males, and that games motivate students to consistently improve, tying them more closely to the material by sparking their interest and setting their imaginations alight.[3] Leveraging powerful tools like competition to grab students and motivate them to truly internalize the material is the key to improving education efforts across the board.

From playtesting various iterations of the rules with players that have notable experience with board games I can say that a game with 3 or 4 players typically lasts 30 minutes or less. A game would most often end with the attempted reading of a zero card, and slightly less often with the exhaustion of a resource. There seemed to be a very slight advantage in the favor of the first player because he or she is likely to have one more turn than the other players. This advantage is usually negligible, and is offset by the simplicity of the board game. Often, one extra turn won’t change the state of the game drastically.

## V Analysis

I was able to bring the game into two high school classrooms. Group A consisted of fourteen students, mostly upperclassmen. I presented the game, allowed them to play a game to completion, and gave the pre- and post-test in a total of forty minutes. Group B consisted of twenty students, mostly freshmen, and I had ninety minutes to present the game, allow them to play the game, and to finally assess their learning. In order to determine if any learning had occurred I devised a very simple pre- and post-test for the students to answer to the best of the ability. I asked a small number of questions to establish a baseline for their knowledge of the field, and then asked them to self-report how knowledgeable they were about computers and computer science. After seeing the students' responses, I believe that the content of the test definitely displays my neophyte status in conducting a knowledge survey. The seven questions I asked were:

- What is computer science?
- Define the term: Algorithm
- Give an example of an algorithm.
- Define the term: Data Structure
- Give an example of a data structure.
- On a scale from 1 to 10, how familiar are you with computers?
- On a scale from 1 to 10, how familiar are you with computer science?

Group A gave the following number of satisfactory answers to the questions 1 through 5 on the pre-test:

- 1 of 14 gave a satisfactory answer
- 1 of 14 gave a satisfactory answer
- 2 of 14 gave satisfactory answers
- 4 of 14 gave satisfactory answers
- 2 of 14 gave satisfactory answers

The average responses to questions 6 and 7 were 6.5 and 2.2 respectively.

Group B gave the following number of satisfactory answers to the questions 1 through 5 on the pre-test:

- 1 of 20 gave a satisfactory answer
- 1 of 20 gave a satisfactory answer
- 1 of 20 gave a satisfactory answer
- 1 of 20 gave a satisfactory answer
- 1 of 20 gave a satisfactory answer

The average responses to questions 6 and 7 were 5.7 and 1.2 respectively.

I think that Group B's inexperience contributed greatly to the disparity between the two playtest groups. It is interesting to note that although some students in both groups couldn't define what an algorithm was or what a data structure was, they could sometimes give a valid example of one, and vice versa. Also, no student in either group successfully answered all of the questions on the pre-test satisfactorily.

After giving the pre-test I began to teach the groups about how to play the game. Teaching went smoothly with Group A. Few questions were asked and few mistakes in gameplay were made. They seemed to pick the game up fairly quickly and appeared to be fairly well entertained by it. One of the four active games was dominated with students refusing to play the game by the rules, instead they ignored the reference sheet provided and only through gentle nudging by both me and their regular teacher were they convinced to restart and play the game as outlined in the rules. The three other games ran smoothly with few incidents. Teaching with Group B was more rocky. With 6 additional students I was teaching 5 active games instead of 4, and they were spread more uniformly across the room, making it difficult to engage with more than one or two games at a time. After the first few minutes of gameplay I began to circulate through the room to observe individual games more closely, correcting play mistakes and rules misinterpretations, something which was unnecessary with the first group for the most part. Group A tended to self-report play errors and be more forward in asking questions about proper gameplay. Group B tended to talk amongst themselves about personal goings-on if there was confusion about gameplay instead of gaining my attention and asking me the proper course of action. This forced me to be much more proactive in my teaching style, moving physically from active game to active game in order to maintain discipline and insure the students were staying on task. I believe the tendency of Group B to get sidetracked was caused by their younger age, their increased number, and their distribution in the classroom. I also believe that the lengthened time period to test the students and teach and play the game is a product of the same. It is important to note that Group A did not, for the most part, choose their own seating, while Group B did.

After the games were finished (or nearly finished in the case of one active game in Group B), I had the students complete the post-test consisting of the exact same 7 questions. Here are Group A's compiled results on the second test:

- 1 of 14 gave a satisfactory answer (No change)
- 5 of 14 gave a satisfactory answer (4 more satisfactory answers)
- 6 of 14 gave a satisfactory answer (4 more satisfactory answers)
- 5 of 14 gave a satisfactory answer (1 more satisfactory answer)
- 4 of 14 gave a satisfactory answer (2 more satisfactory answers)

The average responses to questions 6 and 7 were 5.7 and 2.6 respectively. This is a decrease of .8 units in question 6 and an increase of .2 units in question 7. Group B's compiled results on the second test:

- 1 of 20 gave a satisfactory answer (No change)
- 14 of 20 gave a satisfactory answer (13 more satisfactory answers)
- 15 of 20 gave a satisfactory answer (14 more satisfactory answers)
- 3 of 20 gave a satisfactory answer (2 more satisfactory answers)
- 5 of 20 gave a satisfactory answer (4 more satisfactory

answers)

The average responses to questions 6 and 7 were 5.6 and 1.5 respectively. This is a decrease of .1 units in question 6 and an increase of .3 units in question 7.

The majority of satisfactory answers were regurgitated from my briefing, a specific example being that I used a recipe as an example of an algorithm and 10 of the 14 satisfactory answers from Group B gave a recipe as their example of an algorithm. There were a number of students that were able to give a different example, citing the phases of the game as an algorithm, something which I think is very intuitive. It is interesting to note that the self-reported computers knowledge actually dropped in both cases, which could be an example of random reporting based on the students, but which could also be a direct result of learning during the game. The students learn a little about the game, find it slightly daunting, if a little entertaining, and it exposes a wealth of things that they may not have been aware of previously. This could negatively influence the self-reported numbers as shown, though it is purely speculation on my part. There was a slight increase in both cases of self-reported computer science knowledge, which, because of the meekness of the increase, is probably just for my benefit.

In addition to asking them to answer questions on the quiz, I also asked for written and verbal feedback about the game and the experience. The two most often received comments from these anonymous surveys were that the game was entertaining and that the rules were too dense. I think that a lot of the perceived density comes from the fact that I tried to include both the computer science concepts and the game rules in a single briefing at the beginning of my time with the students. This perceived density probably results from a conscious decision I made to establish and remain faithful to the domain of computer science in the game rules and player reference sheet language. Instead of calling the player-tokens pawns or pieces, I only refer to them as Read/Write heads. Similarly, the plastic markers were only referred to as ones and the cards were only referred to as zeroes. I believe that this context switch from their normal denotations of these words presented a huge and wholly unnecessary barrier to understanding. After discussing and playing the game with a colleague that has a minor in education I've drawn up a plan of action regarding how to teach the game and the science in the same small time period. The primary suggestion was to separate the two learning hurdles, loading the learning of the game into the beginning of the period, and loading the computer science concepts and language into a debriefing period after the game. The debriefing period is the key to experiential learning, it is the time when the student reflects on his experiences and internalizes the learned material. For the instructor, it is the time to draw connections between what the student has been presented and what the student was expected to learn. Implicit learning occurs during the playing of the game, explicit learning occurs during the debriefing. The critical bridging of the concepts learned during the game and then learned during the debriefing will reinforce for the student any learning that may have occurred. In this way, I hope to improve any playtesting sessions I run in the future. Also, in future visits to the classroom I would eliminate the first question of my pre- and post-test altogether. Even those students that gave a satisfactory answer, two out of a total of thirty four tested, were still quite unsure of themselves. I didn't address the question directly in either my briefing or during the play of the actual game itself, so it was a nonstarter and only served to confuse or dishearten most students. I would instead add questions regarding the student's exposure to other board games, asking when the student last played a board game, what that game was, and how often they play board games would have helped me analyze what presented the greatest barrier to learning more about computer science, the rules of the game, or the complexity of the presented material. In regards to the morale of the students, the general reaction among them during the learning process and while playing the game was positive. The students seemed, for the most part, attentive and excited to play the game. There were a small number of students that did not enjoy the exercise, neither the briefing nor the act of playing the game itself. In regards to the game itself, I noticed a number of issues with the current rules concerning mass-teaching and also complexity of play. The large number of piles and cards and tokens floating around the board confused some players, and slowed the games down quite a bit. It might take a few seconds for a player to straighten a pile after every turn, just to insure it didn't become confused with any other adjacent piles. This can be corrected easily without changing much of the core rules, but will require additional development time to correct balance issues before it could be playtested in a public setting again.

## V Conclusion

I believe the future of education lies in games. They motivate, educate, and ease traditional learning biases in a way that current educational methods have failed to prevent or reduce. I hope to use these fundamental attributes to explore the applications of games in the classroom and in the home to initiate learning. Even abstract games with pasted-on themes could be useful in teaching younger children about particular concepts. Essentially, my game is such.

## VI Future Work

I believe the next step is to revise the rules as follows for future playtesting sessions. Instead of dealing the zero cards to the edge of the board, there is a central pile from which zeroes are read and to which zeroes are written. This accomplishes a number of things. It reduces the number of components the players must keep track of, it ties the game closer to Tibor Rado’s Busy Beaver because the tape will begin the game blank, and it decreases set-up and tear-down time. Also, Instead of using the zero cards to track whether a space currently has a zero or a one written to it, there will be a small token which will mark the space with the appropriate symbol. A player’s one marker will be able to occupy the new token by resting in a central hole. If there is no token on the space, then it is blank. One side will be black with a white hole in the middle, this is the zero side. The other side will be red with small black writing at the top and bottom that reads “One”. The red side will have a white hole in the middle into which the player's ones marker is placed. This, too, accomplishes a number of things. It continues to reduce the set-up time, it simplifies the substance of the game, and it acts as a simple and elegant visual marker which can quickly be used by a player to appraise the current state of a particular space on the board. It also eliminates a particular prickly part of the rules concerning what is read when a player encounters a space. I intended there to be a precedence, first try to read a one, if there is no one, try to read a zero, if there is no zero, then end the game. During playtesting, this provided no educational value and served only to confuse the players when I attempted to explain it. Perhaps a modular expansion to the game can introduce conditionals like this.

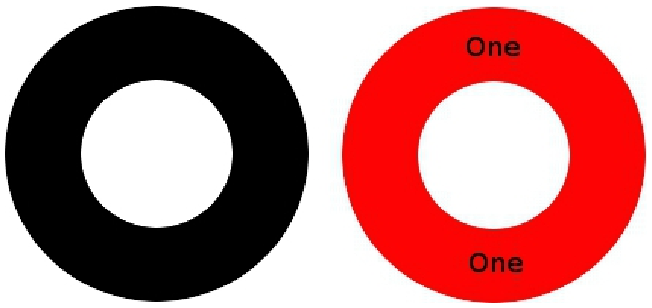


*Illustration 4: A mock-up of the Zero/One token.*

It also will change how the game will end, perhaps lengthening the game since an end condition will be entirely nullified. This could be positive, but more likely it will be negative. I have not yet come up with another end condition for the game to replace the exhaustion of zero cards condition.

In addition to these rules changes, I would implement those changes to the curriculum I listed above, principally the stripping of the reference sheet of computer science domain terms and the separation of the teaching of the game from the teaching of computer science through a briefing/debriefing model. The largest body of work before me is the construction of additional supplemental expansions to the base game that would increase the complexity of the game itself (without hindering it's educational value) and also introduce more complex computer science material (without hindering it's entertainment value.) Some features which I have considered implementing in such an expansion or expansions include the design of a set of states by the players (in order to more closely mimic a Turing Machine) before the game, and the inclusion of new cards which would allow players to edit their transition function during the game. Another feature which I do not touch on directly in the base game is the notion of looping and of basic conditionals. Although these  two things are inherent in the structure of the game, really in the structure of any game, they do not have good conceptual parallels during play and could be too difficult to address during a debriefing after the basic game with new players. It would be interesting to include a loop module which would allow players to manipulate the central data structure using a loop concept, or perhaps implement another supplementary board which would act as an optional “side-game” while the other players continue to interact with the main board. Simple conditionals can easily be slotted into the existing game through more complex zero cards and chance cards. These cards might read as simply as this: “If X, then Y, else Z.” Boolean logic, a topic of study necessary when addressing more complex conditional statements, would be a much more complex addition, though, and might be outside of the scope of even an expansion of the base game.

## References

[1] T. Cortina, “An introduction to computer science for non-majors using principles of computation,” *SIGCSE Bulletin*, pp. 218–222, May 2007.

[2] M. Kebritchi, A. Hirumi, and H. Bai. “The effects of modern mathematics computer games on mathematics achievement and class motivation,” *Computers and Education* vol. 55, pp. 427–443, 2010.

[3] M. Papastergiou, “Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation,” *Computers and Education* vol. 52, pp. 1–12, 2009.

[4] W. Dann, and S. Cooper, “Education Alice 3: Concrete to Abstract,” *Communications of the ACM,* vol. 52, no. 8, pp. 27-29, Aug. 2009.

[5] A. Perrone, and G. Ferraris. "Intelligent Versus Random Beavers—an Agent-Based Approach in Facing the Busy Beaver Problem," *Metroeconomica,* vol. 55, no. 2/3, pp. 332-344, May/Sep. 2004.