Serious games for building skills in computing and engineering

Joe Michael Allen, University of California, Riverside

Joe Michael Allen is a Ph.D. student in Computer Science at the University of California, Riverside. His research interests include educational games for building skills for college-level computer science and mathematics.

Prof. Frank Vahid, University of California, Riverside

Frank Vahid is a Professor of Computer Science and Engineering at the Univ. of California, Riverside. His research interests include embedded systems design, and engineering education. He is a co-founder of zyBooks.com.

Mr. Shayan Salehian, University of California, Riverside
Dr. Alex Daniel Edgcomb, Zybooks

Alex Edgcomb finished his PhD in computer science at UC Riverside in 2014. Alex works with zyBooks.com, a startup that develops interactive, web-native textbooks in STEM. Alex has also continued working as a research specialist at UC Riverside with his PhD advisor, studying the efficacy of web-native content for STEM education.

Serious Games for Building Skills in Introductory Algebra and Computer Science Courses

Joe Michael Allen, Frank Vahid*, Shayan Salehian, Alex Edgcomb*
Dept. of Computer Science and Engineering, University of California, Riverside
*Also with zyBooks

ABSTRACT

Various forms of online games are used in education. Most games are generic in nature, such as quiz games or adventure games, into which nearly any topic (like math, history, spelling, computer programming) can be integrated. We focus instead on games that each builds a specific skill through repetition. Repetition is known to improve learning by moving concepts from short-term to long-term memory, but excessive uninteresting repetition can lead to stress and fatigue, and eventually to students giving up. In contrast, many games involve repetition but in an interesting/enjoyable form, with the player striving to improve. Our approach custom designs each game for a specific skill deemed important to a topic, such as writing if-else statements in computer programming. The custom design is needed to make the skill be an essential part of the gameplay. This paper presents several games our team has developed so far, for topics in computing and math, and describes plans to build more. The games are free and currently at http://www.cs.ucr.edu/~vahid/seriousGames/. The games are web-based (HTML5) and require no software installation, being playable on any standard web browser.

I. Introduction

Learning some subjects involves developing skills. The prototypical example is the learning of multiplication tables; by developing the skill of multiplying single digit numbers in one's head almost instantly, one can shift focus to higher-level learning like multiplying multi-digit numbers, performing long division, solving algebraic equations, and more. In computing, a good programmer has developed various skills, such as writing if-else statements to detect numerical ranges. An experienced programmer can correctly write such a statement almost instantly without expending much thought, instead focusing mental efforts on more important program aspects.

Developing a skill requires repetition. Unfortunately, when learning, repetition is often exhausting. Unlike tasks that humans voluntarily repeat (like consuming certain foods), tasks like multiplying numbers or writing if-else statements are not something most humans naturally enjoy. As a result, a teacher cannot assign as much homework as desired, lest the teacher "burns out" the students leading to poor performance, class withdrawals, or distaste for a subject.

However, many computer games inherently involve repetition. The popular game of Tetris, for example, involves rotating various falling shapes to fit neatly into rows without gaps. As a person repeatedly plays the game, the person learns to quickly recognize certain patterns and to perform rotations almost without thinking, improving the player's score and causing the game to become faster and more challenging. The accomplishment of going further drives the player to want to play more. The repetition is fun, rather than exhausting, and the skill is developed.

Computer games have long been used in education, but not usually for building skills. Instead, the games present common learning activities in a generic game form. For example, the activity of taking a quiz may be converted into a generic game form wherein a question is posed and one has to "shoot" the right answer, or has to remember the answer on a set of turned-over cards (the "memory" game). As another example, students may virtually proceed from room to room, answering questions or solving problems in order to win prizes and gain access to other rooms. Social components may be added to such games as well.

Such games typically are independent of the actual subject matter. The same game form can be used for math, history, or computing, by plugging in different questions or problems. For college students, wrapping quizzes in a game form is not necessarily engaging (and some would say it is patronizing).

Our focus is instead to build skill through repetition. As a result, for a given skill, we seek to custom-design a game that inherently involves repeating that skill, ideally in a way that is engaging and not exhausting. Various tasks in computing, math, and engineering benefit from skill mastery, like creating logical expressions to detect specific number ranges, creating for loops that print particular items in an array, or determining the voltages at various points in an electrical circuit.

We present several games we have developed for such college-level learning. We expect to make the games available via the web for free (and some already are) to students and teachers who wish to master basic skills so as to enable focus on higher-level thought in math and computing.

II. Background

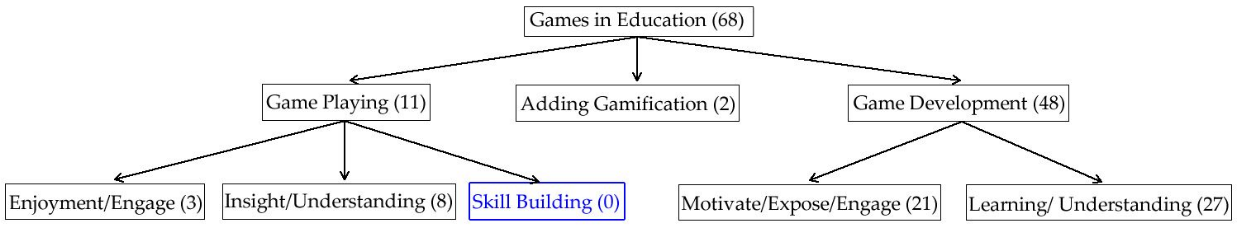
Gaming has become one of the most popular pastimes in world. In 2016, the Entertainment Software Association surveyed over 4,000 U.S. households and found that 63% of households have at least one person who plays video games for at least 3 hours a week [5]. This amount is an increase from the same study done in 2015 which found 42% of households had a gamer [7]. Furthermore, Granic found that 97% of American children and adolescents play games for at least one hour per day in the United States [9]. Since the first video games were released in the

1950's [10], games have improved on the visuals, storyline, and enjoyment. People spend hours playing popular commercial games like Call of Duty or Angry Birds. Educators recognized the impact games were having on American culture, so began incorporating games in education. This is where the field of serious games was born.

Games have been used in education from kindergarten through college for many decades now. The first paper we found in the SIGCSE conference (a leading conference on computer science education) related to using games for education was in 1997 [4]. The paper discussed a game used to teach network protocols to high school students by making them act as the protocol components. Games can be a powerful tool since they can change the way that concepts are taught in the classroom.

To understand the current state of games used in computer science education, we surveyed SIGCSE conference paper submissions over the past 20 years. Paper submissions needed to meet the following criteria for inclusion in our survey: 1) the title of the paper included either "game" or "games" and 2) the paper was not about a panel discussion taking place at the conference. In total, we found 68 papers to include in our survey. From the papers surveyed, we created a taxonomy to categorize the common ways that games were being used for education. We recognized three major categories 1) Game development, 2) Adding gamification to the classroom, and 3) Game playing. Figure 1 summarizes these findings.

Figure 1: Games-in-education taxonomy. Numbers indicate the SIGCSE papers in each category. 7 game papers did not fit into these categories. This paper introduces the Skill Building category.



A. Game Development

One way that games are incorporated in education is by having students develop games themselves. This method is primarily seen in computer science education. Game development aims to provide students with a deeper understanding and appreciation for programming via a motivating application. Game development is also used in many camps [12], workshops [13], or outreach programs [14][15] for younger students. Using game development in education can have many purposes. One purpose is to motivate or engage students. Cliburn and Miller [11] show this when they allowed their students to choose a final project (a game, a "choose your own adventure" story, or traditional project) for their introductory programming class. Additionally, game development can be incorporated for the purpose of helping students learn and understand various concepts. Leutenegger and Edgingto [16] created an introductory level programming class with a “games first approach” to teach fundamental computer science concepts.

B. Adding Gamification to the Classroom

A second way that games can be used in education is gamification. Gamification is the use of game design elements in non-game contexts [17]. Gamification can take the form of adding leaderboards in a classroom to promote healthy competition, awarding student achievements for completing class related assignments, or by adding “quests” to encourage group related assignments. Decker and Lawley [18] created the “Just Press Play” project for first-year college students to enhance their undergraduate experience. One portion of this project was creating an achievement system that awarded students for completing various tasks and activities. The focus was on creative and cultural activities, technical or skill-focused activities, and individual/social activities. Some achievements included making a faculty member laugh, visiting on-campus locations, and having a passing level of 90% for an intro programming course. Toth and Kayler [19] also introduced gamification by integrating a role-playing game into two computer science courses. Grades were converted into experience points, stories were added to projects to give the feeling as if the students were going on an adventure, and treasure was hidden at campus locations that would help nurture students’ education.

C. Game Playing

The third way games are used in the classroom is through game playing. Game playing is a simple way to use games in the classroom. There are several reasons why instructors may want to use game playing in their class: 1) Enjoyment/Engagement, 2) Insight/Understanding, and 3) Skill building.

In the traditional classroom, some students are unexcited, unengaged, and uninterested in the content being covered. Games have been used in an attempt to address these problems by adding excitement and enthusiasm into the classroom. Game playing may add interaction and a sense of fun. Soh [20] created a game-based technique that was integrated in their Multiagent systems class. The class was extended to include four game days where students would form groups and compete in role-playing games related to issues such as auction, task allocation, coalition formation, and negotiation. Based on student surveys, game days were enjoyable and the most helpful part of the class, helping them learn and understand the material.

Game playing can also be used to help teach students and provide insight and understanding into a topic. Researchers [21] designed a game to teach the computer science concept of minimum spanning trees (MST) to middle school students. The game simulated a graph having has edges and nodes. The player's goal was to find a flag using the smallest distance to traverse the graph (i.e., find an MST). Additionally, researchers [22] built a game to teach distributed systems to undergraduate students. The game was designed to teach students how to keep a system running despite being plagued with multiple issues. The researchers found that students were engaged and were able to learn quickly by interacting with a complex distributed system.

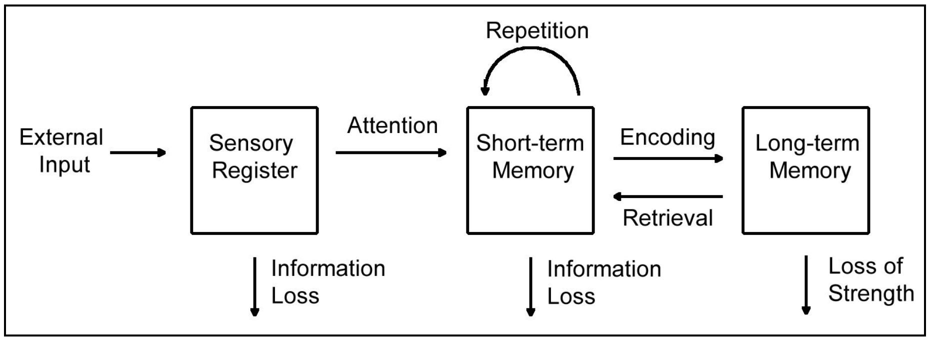
Lastly, games can be played to build skill. This is where we are focusing our efforts. Of all the SIGCSE papers we examined, none mentioned using games to build skill. We are not currently aware of others focusing on such work. Two key items are important when developing games to build skill: 1) the game should be inherent to the topic being taught and 2) repetition should be used to move the skill from short-term to long-term memory.

In contrast, many games today are basically quizzes. These games add color, design, and a point system, yet they are played the same way a quiz would be taken: select one possible answer and then see if it is correct or not. Kahoot! [23] is one such example of this. Any type of question can be substituted into this game; be it physics, biology, or English related. These games can be fun, but they are not necessarily teaching a skill through repetition. When people learn something, that something starts off in short-term (working) memory. Through repetition, that skill moves into long-term memory allowing mastery. The following section explains.

III. Philosophy

Games that build skill through repetition have the potential to be highly effective learning tools. External stimuli is immediately registered within the appropriate sensory dimension of the brain. From there, the information either fades or is moved into the short-term (working) memory. Information stored in short-term memory eventually disappears, or with practice and repetition can be moved into relatively permanent long-term memory. See Atkinson and Shiffrin's Memory Model [1].

Figure 2: Visual representation of Atkinson and Shiffrin's Memory Model [1][25].



Around the 1990's, Jonathan Sweller [2][3] built upon this work and contributed his own Cognitive Load Theory. Sweller emphasized that the brain's working memory is limited and only has a finite amount of space to process information. Sweller believed that learning can be broken down into two critical processes: schema acquisition and transferring learned information into automatic processing.

A schema is equivalent to a basic block of learning. The brain builds schemas to classify information. By creating more schemas, the brain will have more knowledge and ability to deal with similar problems. Once a schema is formed, the next step is to move that schema into automated processing or long-term memory. With the proper amount of practice, students should be able to move newly learned information (or new schemas) in working-memory to long-term memory. This process frees up space in working-memory for students to think at a higher level or focus on a completely new problem all together. Furthermore, information stored in short-term memory must be processed, taking time and effort. By storing information in long-term memory, the information can be accessed almost immediately and does not need to take time to be processed.

IV. Games

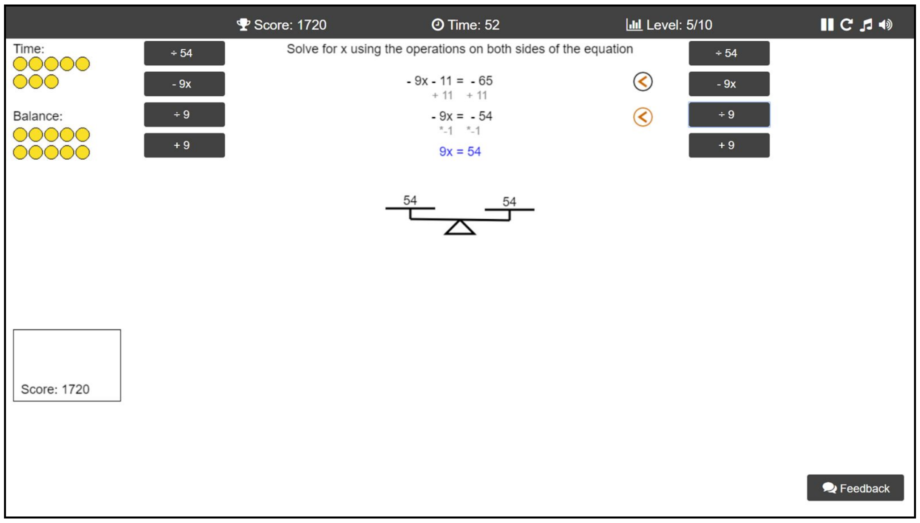
We are designing web-based games to build skill through repetition. Currently, we are focusing on creating games to teach introductory algebra and introductory computer science topics. The following section will introduce the games that we have created. Most of our games will be available to use for free at http://www.cs.ucr.edu/~vahid/seriousGames/

A. Solve for x (Algebra)

Many math students struggle with understanding the concept of solving for a single variable in an equation ("solve for x"). Students must master this skill to advance to higher-level concepts in math. We developed a game to help students master the skill of solving for x.

The solve-for-x game provides equations containing a single variable, which the player must solve for. Equations are randomly generated, and increase in difficulty as the player progresses. The screen's center has a see-saw-like scale to add visual representation of keeping the equation balanced, which teeters left or right depending on the equation's current state. The player has four possible operations they can apply to the equation. To emphasize the concept of keeping the equation balanced, the game requires the player to select the left and right operations before advancing. If the player tries to apply two different operations, they cannot advance and a help message appears. We do this to make sure that the player is comfortable making mistakes while still learning from errors. The player quickly learns that the see-saw stays balanced (and more points are awarded) by applying the same operation to both sides of the equation. Through repeated play, players learn which operations more quickly solve the equation (earning more points for more quickly-solving). As the levels increase, the player is given harder more complex equations, eventually involving fractions, higher coefficients, and multiple instances of the variable. When all the levels are completed, a final score is given and the player is encouraged to play again to beat their high score.

Figure 3: Screenshot of the “Solve for x” game.

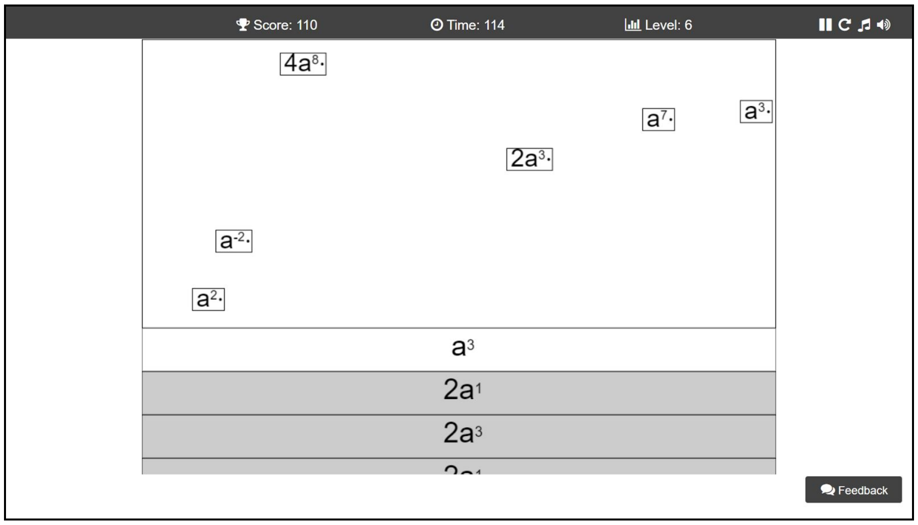


B. Exponent Multiplication (Algebra)

Students often confuse the idea of multiplying/dividing terms having exponents with adding/subtracting terms having exponents. This simple mistake which can be fixed if the proper training is given. We developed a game to help students build the skill of multiplying/dividing terms having exponents.

The “Exponent Multiplication” game teaches the concept of multiplying and dividing terms with exponents. Terms, comprised of variables and exponents, float around the screen and the player must form and match terms with their corresponding values in the blocks below. The player is able to drag terms together to multiply them or they can double click on terms to divide them apart. As the game progresses, the blocks at the bottom of the screen will rise and if they hit the top of the screen the game is over. When the player matches the term with the correct block, they will receive points and the block they used to score will disappear. As the player beats the levels, the terms get more difficult and complex. At later levels, terms will include negative exponents as well as positive and negative coefficients.

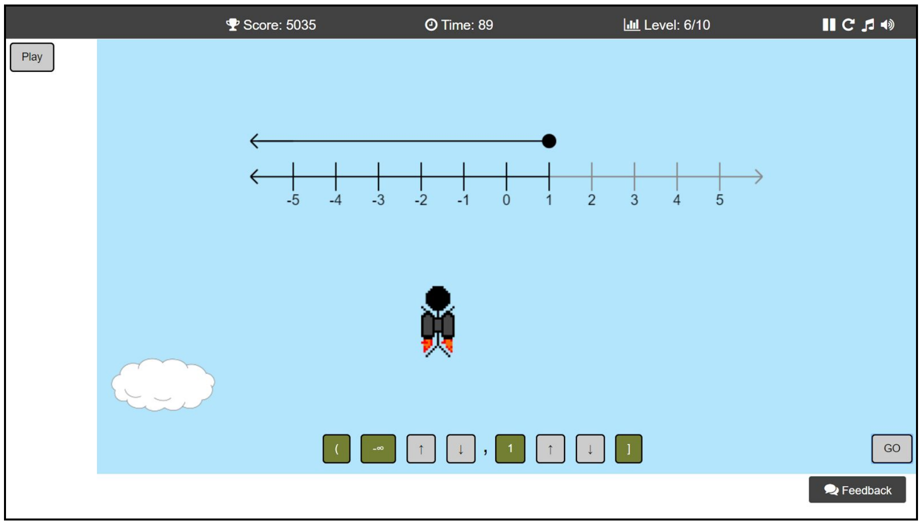
Figure 4: Screenshot of the “Exponent Multiplication” game.



C. Interval Notation (Algebra)

Interval notation is a concept that many students, even at the college level, have trouble understanding. Understanding intervals and their relationship with a number line is an important concept. We developed a game that helps students master interval notation and the number line.

Figure 5: Screenshot of the “Interval Notation” game.



In the “Interval Notation” game, the player takes on the role of a character with a jetpack who is trying to traverse obstacles and fly as high as they can go. The game begins by showing the player a number line with an indicated range. To win, the player must create the proper range on the number line using interval notation. When the player submits their answer, there are four possible outcomes; each with different animations to emphasize different lessons. The player can get the answer correct, provide a partial solution, provide an incorrect solution, or they can provide an invalid solution. At any point the player answers incorrectly, they will be given additional chances to answer the same problem after receiving helpful feedback from the game; further encouraging learning even when they are incorrect. In beginning levels, the solutions require a single interval answer while in later levels, the correct answer requires the player to specify two intervals.

D. Boolean Number Line (Computer Science -- Under Development)

Computer science students need to understand boolean logic. Boolean logic can get quite complicated, but we chose to focus on introductory concepts such as the functionality of the logical AND (&&) operator and the logical OR (||) operator. We combined these concepts with ranges depicted on the mathematical number line. We hope to teach the concepts of the logical AND operator, the logical OR operator, and how to interpret ranges on a number line.

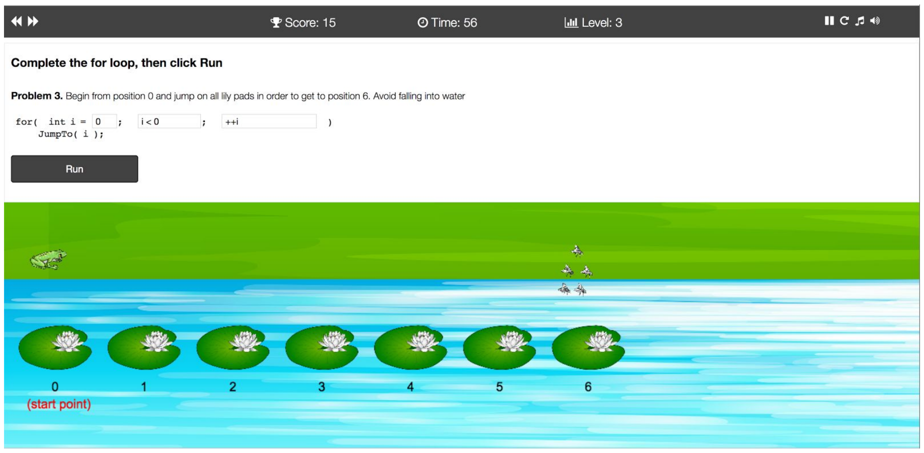
In the the “Boolean Number Line” game, aliens are taking over the earth and stopping them is up to the player. Alien ships are located at given intervals containing boolean logic, like (x > 2) && (x < 5). The player must build up an attack using a number line defense system. The player scores by shooting the ships out of the sky. This is done by the player providing a solution on the number line which matches the boolean range problem. As the levels increase, so does the difficulty of the problems.

E. For Loop (Computer Science)

Iteration (looping) in computer programming is a skill that programmers should be comfortable with. To help teach this concept, we have designed a game that aims to improve the player’s loop creation skill.

The game includes a frog, some lily pads with numbers, and some flies which hover over the last pad. The aim of the game is to move the frog from the beginning pad to the end pad so that the frog can eat the flies and the player can score points. The specific concepts which the game focuses on are loop setup conditions, loop increments, and the loop terminating condition. The player is required to determine these factors correctly to move the frog to the desired pad. The game becomes gradually harder as levels are completed. In later levels, the player must deal with both negative and positive step-sizes in addition to different ending. Finally, to measure how fast learners can solve each level, we reduce the number of flies on the last pad (lowering the level’s maximum score) as time elapses.

Figure 6: Screenshot of “For Loop” game.



Complete the for loop, then click Run

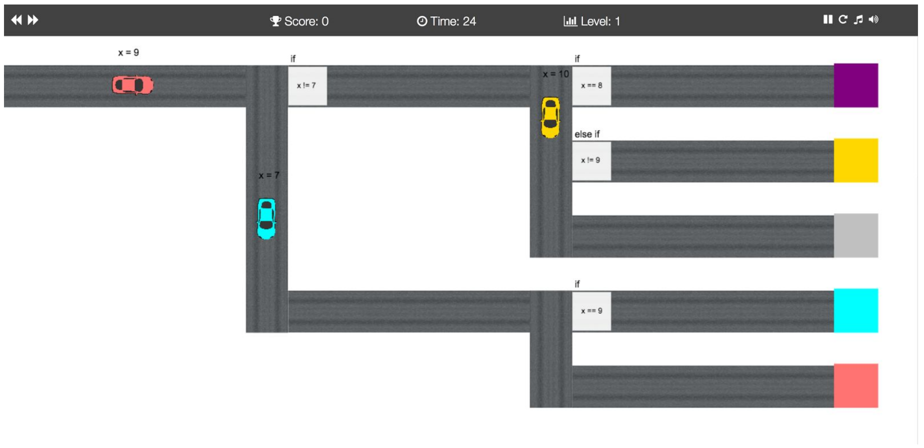
Problem 3. Begin from position 0 and jump on all lily pads in order to get to position 6. Avoid falling into water

```
for(  int i = 0  ;  i < 0  ;  ++i  )
    JumpTo( i );
```

F. If/Else Statements (Computer Science)

Another important concept when learning programming is the if-else statements. To improve the understanding of if-else, we designed a car game.

Figure 7: Screenshot of the “If/Else Statements” game.



This game consists of a number of colored cars trying to reach their destination; a house of the same color. Moreover, a value is assigned to each car which is used for navigation. Cars are connected to destinations via a number of roads. Each road has an expression at its start that determines which cars are allowed to enter the road. The player can click on a car to change the car's value, so that the car will reach its proper destination and score points. The game has different levels beginning with simple if-else conditions and advances to more complex nested if-else conditions. Players will eventually learn and become faster in processing these concepts as they are trying to direct more cars to their destination to earn more points.

G. Nested For Loop (Computer Science -- Under Development)

Nested for loops is another essential concept that students should understand if they want to be successful in solving more advanced programming problems. Traversing specific ranges in a matrix, or two-dimensional array, is an example of using a nested loop that we intend to teach.

H. Interval Counting (Computer Science -- Under Development)

Being able to count the number of integers in a range is an important skill for programmers to have. This skill is highly applicable when working with loops, data structures, etc. Programmers should be comfortable determining how many integers are within a range, but they are typically off by one. For example, how many integers are included in the range 0-10? The answer is 11, when typically a response may be 10. We hope to teach the concept of counting integers within a given range.

I. Skill / Fun

Each game is specifically designed for each skill. The solve-for-x skill involves keeping an equation balanced while applying operations to get x on one side; the game's points are earned directly from keeping such balance and applying the best operations. The exponent game involves combining terms to achieve particular results; the game directly involves selecting and dragging terms to combine them. The interval game involves knowing how intervals appear on a number line; the game involves flying through properly formed intervals on that line. The looping game involves creating loops with a specific iteration pattern; the game directly illustrates the pattern via the jumping frog. The if-else game involves understanding how a program executes if-else statements having particular expressions; the game directly shows such execution via the moving cars. Note that none of the games could readily be converted to teach a different skill; each game's play is inherent to each skill.

Each game's "fun" parallels features of other popular games, such as the exponent game having similar features as Asteroids (combining terms, rather than dividing them) and Tetris (keeping the bottom from rising to the top).

V. Future Work

We plan to make more games for introductory math and computer science college level courses (and possibly beyond). We plan to introduce these games into intro-level classes and study their effects on student learning and enjoyment.

VI. Conclusion

Over the past couple decades, games have been introduced into the classroom at an increasing rate. Games can be integrated into education through game development, adding gamification to a class, or by game play. We are focusing on creating web-based games for college-level introductory math and computer science courses. The novelty of our work twofold: 1) our games are inherently built around a single core topic and 2) the games are designed to build *skill through repetition*. By practicing and repeating a skill enough, the skill may be "mastered" and move into automatic, long-term memory. We introduced several games that we have created or are developing. We are excited to see how our games are used and observe how effective they are in and out of the classroom. The games are at http://www.cs.ucr.edu/~vahid/seriousGames/ and are free. The games are web-based (HTML5) and require no software installation, being playable on any standard web browser.

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation (grant #1542851), a U.S. Department of Education GAANN fellowship, and a Google faculty research award. We are grateful for their support.

REFERENCES

[1] Atkinson, R.C. and Shiffrin, R.M. "Chapter: Human memory: A proposed system and its control processes." The psychology of learning and motivation, Vol. 2, pp. 89–195. 1968.
[2] Chandler, P. and Sweller, J. "Cognitive Load Theory and the Format of Instruction." Cognition and Instruction, 8(4), 293-332. 1991.
[3] Sweller, J. "Cognitive Load Theory, Learning Difficulty, and Instructional Design." Learning and Instruction, Vol. 4, pp. 295-312. 1994.
[4] Shifroni, Eyal and David Ginat. "Simulation game for teaching communications protocols." SIGCSE '97 Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education. 1997.
[5] Essential Facts 2016. http://essentialfacts.theesa.com/Essential-Facts-2016.pdf. Accessed February 2016.
[6] http://www.polygon.com/2016/4/29/11539102/gaming-stats-2016-esa-essential-facts. Accessed February 2016.
[7] Essential Facts 2015. http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf. Accessed February 2016
[8] http://www.polygon.com/2015/4/14/8415611/gaming-stats-2015. Accessed February 2016.

[9] Granic, Isabela, Adam Lobel, and Rutger C. M. E. Engels. “The Benefits of Playing Video Games.” American Psychologist. January 2014
[10] Games Studies Collection.
http://www.stonybrook.edu/commcms/libspecial/videogames/tennis.html
[11] Cliburn, Daniel C. and Susan M. Miller. “Games, Stories, or Something More Traditional: The Types of Assignments College Students Prefer.” SIGCSE '08 Proceedings of the 39th SIGCSE technical symposium on Computer science education. 2008.
[12] Alan Cleary, Lucas Vandenbergh, John Peterson. “Reactive Game Engine Programming for STEM Outreach.” SIGCSE '15 Proceedings of the 46th ACM Technical Symposium on Computer Science Education. 2015
[13] Liu, Jiangjiang, Cheng-Hsien Lin, Joshua Wilson, David Hemmenway, Ethan Hasson, Zebulun Barnett, and Yingbo Xu. “Making games a "snap" with Stencyl: a summer computing workshop for K-12 teachers.” SIGCSE '14 Proceedings of the 45th ACM technical symposium on Computer science education. 2014.
[14] Veronica Cateté, Kathleen Wassell, Tiffany Barnes. “Use and development of entertainment technologies in after school STEM program.” SIGCSE '14 Proceedings of the 45th ACM technical symposium on Computer science education. 2014.
[15] Antti-Jussi Lakanen, Ville Isomöttönen, Vesa Lappalainen. “Life two years after a game programming course: longitudinal viewpoints on K-12 outreach.” SIGCSE '12 Proceedings of the 43rd ACM technical symposium on Computer Science Education. 2012.
[16] Scott Leutenegger and Jeffrey Edgington. “A Games First Approach to Teaching Introductory Programming.” SIGCSE '07 Proceedings of the 38th SIGCSE technical symposium on Computer science education. 2007.
[17] Sebastian Deterding, Dan Dixon, Rilla Khaled, Lennart Nacke. “From Game Design Elements to Gamefulness: Defining “Gamification.” MindTrek '11 Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments Pages 9-15. 2011.
[18] Adrienne Decker and Elizabeth Lane Lawley. “Life’s a Game and the Game of Life: How Making a Game Out of it Can Change Student Behavior.” SIGCSE '13 Proceeding of the 44th ACM technical symposium on Computer science education. Pages 233-238. 2013.
[19] David Toth and Mary Kayler. “Integrating Role-Playing Games into Computer Science Courses as a Pedagogical Tool.” SIGCSE '15 Proceedings of the 46th ACM Technical Symposium on Computer Science Education. Pages 386-391. 2015.
[20] Leen-Kiat Soh. “Using Game Days to Teach a Multiagent System Class.” SIGCSE '04 Proceedings of the 35th SIGCSE technical symposium on Computer science education. Pages 219-223. 2004
[21] Britton Horn, Christopher Clark, Oskar Strom, Hilery Chao, Amy J. Stahl, Casper Harteveld, Gillian Smith. “Design Insights into the Creation and Evaluation of a Computer

Science Educational Game." SIGCSE '16 Proceedings of the 47th ACM Technical Symposium on Computing Science Education. Pages 576-581. 2016
[22] Joel Wein, Kirill Kourtchikov, Yan Cheng, Ron Gutierez, Roman Khmelichek, Matthew Topol. "Virtualized Games for Teaching About Distributed Systems." SIGCSE '09 Proceedings of the 40th ACM technical symposium on Computer science education. Pages 246-250. 2009.
[23] https://getkahoot.com/
[24] http://www.cs.ucr.edu/~vahid/seriousGames/
[25] McLeod, S. A. (2007). Multi Store Model of Memory - Atkinson and Shiffrin, 1968. Retrieved from www.simplypsychology.org/multi-store.html