Simplifying Algorithm Learning Using Serious Games

Sahar Shabanah
George Mason University
4400 University Dr.
Fairfax, VA 22033, USA
sshaban1@gmu.edu

Dr. Jim X. Chen
George Mason University
4400 University Dr.
Fairfax, VA 22033,USA
jchen@gmu.edu

ABSTRACT

Algorithm Visualization using Serious Games (AVuSG) is an algorithm learning and visualization approach that uses serious computer games to teach algorithms. It visualizes an algorithm to be learned in four forms: a text, a flowchart, a game demo, and a game. Moreover, (AVuSG) method integrates learning theories and models in addition to motivation theory to introduce three learning models that simplify algorithm learning.

Categories and Subject Descriptors

L.5.1 [**SIMULATION/GAMES**]: Game Based Learning; I.6.8 [**Gaming**]: Simulation and Modeling

Keywords

Serious games, algorithm visualization, algorithm learning

General Terms

Learning Theories, Game Engine, Algorithm Visualization

1. INTRODUCTION

Data structures and algorithms are important foundation topics in computer science education. Students deal with algorithms in many computer science courses. For instance, in computer graphics, students learn objects rendering algorithms, in networking, they study algorithms that solve networks traffic congestion, and in database, they learn algorithms that search or sort data. Accordingly, teaching algorithms is a common activity that takes place in many computer science classes. However, algorithms are often hard to understand because they usually model complicated concepts, refer to abstract mathematical notions, describe complex dynamic changes in data structures, or solve relatively difficult problems. Consequently, teaching algorithms is a challenging task that faces instructors and requires a lot of explaining and illustrating. Therefore, teaching aids other than chalkboard and viewgraph are needed to help students understand algorithms better [1]. The human ability to realize graphic representations faster than textual representations led to the idea of using graphical artifacts to describe the behavior of algorithms to learners that have been identified as algorithm visualization [9].

WCCCE'09, May 1-2, 2009, Burnaby, BC, Canada.

Motivation. Educational Computer Games are computer games that involve learning of certain knowledge [31]. Despite the frustration with educational computer games in the past, they reemerged recently as Serious Games. Serious games are computer games that have been developed for serious purposes other than entertainment such as training, advertising, simulation, and education. Since 2001, there have been several researches related to serious games such as the *Games To Teach* [19] and *The Making Games* [25] projects. The following features of computer games qualify them to be used for education in general and for algorithm learning in particular:

- **Computer games are popular.** Computer games have been broadly played all over the world by adolescents and young adults. In particular, the number of hours the standard college students spend on reading is half the time that they spend on playing computer games [26]. Therefore, the best method for teaching today students is to use computer games.

- **Computer games are interactive.** Computer games fully interact with players and encourage them to think and act. Thus, the use of computer games will improve algorithm learning since there is an increasing correlation between algorithm learning and the level of students' engagement [13].

- **Computer games are competitive.** A computer game player spends many hours in learning and playing the game. The motivation of the player to spend all that time with the computer game aroused from the game itself (Intrinsic Motivation) and not from an external incentive. Intrinsic Motivation improves learning as research shows [21].

- **Computer games simplify assessment.** To win a computer game, the player must understand its rules very well. Therefore, students understanding of the algorithms can be assessed using the winning/losing criteria of computer games that simulate the behavior of those algorithms.

- **Computer games utilize entertainment.** The use of computer games converts the unpleasant and tedious operation of algorithm learning into an enjoyable and interesting experience.

This paper introduces a new algorithm visualization method, namely Algorithm Visualization using Serious Games (AVuSG) that uses learning theories and serious computer games to simplify algorithm learning. The main part of this paper is divided into three sections. Section 2 describes related work. Section 3 explains the conceptual framework of (AVuSG) method, section 4 describes the (AVuSG) systems design, section 5 illustrates the algorithm game design, and section 6 describes three algorithm games prototypes.

2. RELATED WORK

SOS (1981) video was the first recognized algorithm visualization system. It uses animation, color, and sound to explain three sorting algorithms: insertion, exchange, and selection sorts [2]. However, BALSA (1984) was the first real-time, interactive algorithm visualization system, it allows the user to start, stop, or even run an algorithm backward [5]. Algorithm Explorer (2007) is an algorithm visualization system that uses three-dimensional objects to represent common data structures and animations to visualize algorithms. Its user interface allows the user to control how the visualization played and displayed on the screen [7]. SOS allows learners to view the visualization passively while BALSA and Algorithm Explorer provide the user with some kind of control over the visualization viewing process. Viewing is the most form of engagement that existing algorithm visualization systems provide for their users, other less supported engagement forms are responding, changing, constructing, and presenting [23]. MatrixPro (2004) is a visualization system that supports responding by asking viewers to answer questions related to the presented visualization. It uses a server called Trakla2 (2003) to automatically generate algorithm exercises and assess students returned answers [20]. JHAVE (2005) adds a responding support for a variety of algorithm visualization systems by providing stop-and-think questions [22]. Some algorithm visualization systems allow their viewers to change the visualization data or some other features. For example, GeoWin (2002) is a visualization system that allows the user to manipulate a set of actual geometric objects through the interactive interface [3]. Other system is CATAI (2002), which enables the user to invoke some methods on the running algorithm. In terms of method invocations, it is possible directly to access the content of the data structures or to execute a piece of code encapsulated in an ordinary method call [11]. ANIMAL (2002) supports constructing by enabling users to visualize algorithms through the composition of low-level drawings and the animation operations, but not on the abstract data types represented in the animation [27]. ALVIS (2004) is an interactive environment that enables students to quickly construct rough, unpolished (low fidelity) visualizations, and interactively present those visualizations to an audience. Its successor is ALVIS-Live! (2005) that reevaluates an algorithm line on every edit to update its visualization dynamically [18]. A project called Algorithm Studio (2004) supports presenting by allowing students to present their visualizations to their instructors in one-to-one sessions then to the entire studio. Finally, at the end of the semester, the students must present their solutions for an extra set of design problems to a jury of instructors [16].

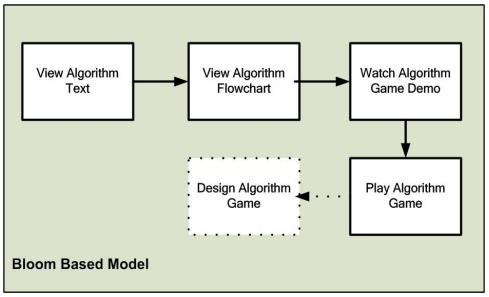


Figure 1: Bloom Based Model

Some researchers found that there is no significant difference in educational outcomes between students who use visualizations and those who do not [6] [17]. Moreover, in a recent effort to build a wiki for existing algorithm visualization systems, Shaffer *et al.* searched and analyzed hundreds of visualization systems and found that "most existing algorithm visualization systems are of low quality, and the content coverage is skewed heavily toward easier topics [29]."However, researchers remain positive about visualizations in general as Shaffer *et al.* states "while many good algorithm visualization systems are available, the need for more and higher quality visualization systems continues. There are many topics for which no satisfactory visualization systems are available. Yet, there seems to be less activity in terms of creating new visualization systems now than at any time within the past ten years [29]."The focus on graphics and sound instead of teaching aspects in the design of many algorithm visualization systems is responsible for their failing to be effective in teaching algorithms [30]. Other reason is the lack of features that encourage students' engagement with the displayed visualization [28]. Nevertheless, researchers identified the level of students' engagement as the most effective factor in the success of any algorithm visualization system since there is an increasing correlation between learning and the level of students' involvement with the visualization [13].

Algorithm Visualization using Serious Games (AVuSG) addresses the shortness in current algorithm visualization techniques by integrating learning theories and models in algorithm learning. Moreover, by visualizing algorithms as computer games, (AVuSG) introduces "playing"as a new engagement form that maximally engages students in the learning process.

3. (AVUSG) CONCEPTUAL FRAMEWORK

(AVuSG) framework consists of three parts: the Algorithm Representation Forms, the Learning Processes, and the Learning Models.

3.1 Algorithm Representation Forms

(AVuSG) produces four forms of representations or visualizations for the algorithm to be learned:

1. **The Algorithm Text:** a description of the algorithm steps that shows its basic idea and how it works.
2. **The Algorithm Flowchart:** a graph depicting the static visualization of the algorithm functionality.

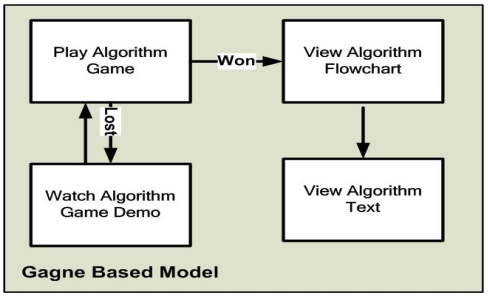


Figure 2: Gagne Based Model

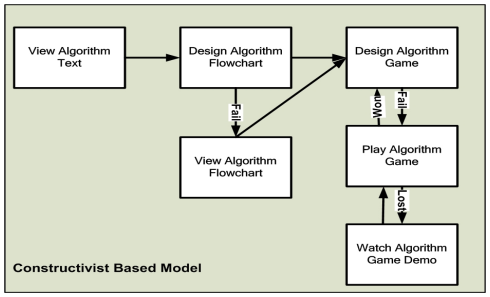


Figure 3: Constructivist Model

3. **The Algorithm Game Demonstration:** a dynamic visualization of the algorithm operations using the self-running demo of the algorithm game.
4. **The Algorithm Game:** an educational computer game with a game-play that simulates the behavior of the algorithm and graphics depict the features of its data structure.

3.2 Learning Processes

(AVuSG) defines three learning processes for learners to engage with any produced algorithm representation form:

1. **The Viewing Process:** in this process, the learner views the algorithm text, flowchart, and game demo.
2. **The Playing Process:** in this process, the learner plays the algorithm game.
3. **The Designing Process:** in this process, the learner develops the algorithm text, flowchart, game, and its demo.

3.3 Learning Models

(AVuSG) introduces three learning models that can be adopted either by students to learn the algorithm or by the instructors to teach the algorithm depending on the learning objectives that they want to achieve.

3.3.1 The Bloom Based Model

By applying this model (Figure 1), the learning objectives of Bloom Taxonomy (A.2.1) can be achieved as follows:

1. The learning starts in the viewing process, where learners watch the algorithm text to build their knowledge
2. Then, learners view the algorithm flowchart and game demo to comprehend how the algorithm works.
3. Next, in the playing process, learners apply their knowledge of the algorithm to play the algorithm game.
4. The game playing is divided into several game moves; each move simulates one algorithm step. During game playing, learners first analyze the algorithm into its steps to win each game move then synthesize all moves to win the whole game.
5. Learners can easily evaluate the speed, complexity, and efficiency of an algorithm by playing its related game. For example, if the game is hard to win, this means the algorithm is complex. Also, if the game data is not large this means the algorithm is suitable for small size data and so on.
6. As an optional and for more learning outcome. In the design process, learners analyze the algorithm and its played game to synthesize a new algorithm game design. Then, learners evaluate their designs for the new algorithm game.

3.3.2 The Gagne Based Model

The steps of this model (Figure 2) simulates the events of Gagne model of instruction (A.2.2) as follows:

1. The learning starts by playing the algorithm game that was created by the instructor, to gain the learners attention and activate their motivation. If lost the game, learners are provided with guidance through algorithm game demo. The playing process builds and stimulates the required prerequisite information related to the algorithm.
2. After winning the game, the learners are presented with the algorithm text and flowchart in the viewing process to learn the algorithm.
3. Eliciting and assessing the performance of learners can be achieved using the winning/losing criteria of the algorithm game.
4. Promoting retention and transfer can be acquired in the playing process.

3.3.3 The Constructivist Model

A description of this model (Figure 3) is as follows:

1. Learners start learning by viewing the algorithm text to have an idea about how the algorithm works.
2. Then, learners go to the designing process to design the flowchart of the algorithm.
3. If succeeded in building the flowchart, learners can start to design and develop their own algorithm games.
4. However, if learners fail any of these processes, they always can watch the algorithm flowchart, game demo, and play the algorithm game that has been built by their instructors.

The model deploys the constructivist theory (A.1.3) to affect learning in three ways:

1. Curriculum: for the same algorithm, each student builds his own algorithm game and flowchart using his prior knowledge.

Figure 4: Serious-AV Main

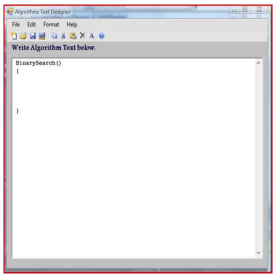


Figure 5: Algorithm Text Designer

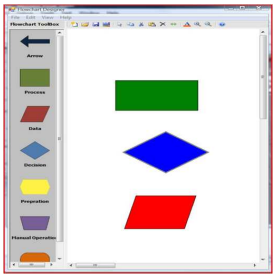


Figure 6: Algorithm Flowchart Designer

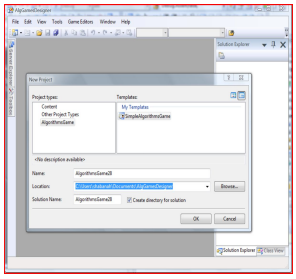


Figure 7: Algorithm Game Designer

2. Instruction: students analyze, interpret, and predict information about the algorithm to design its flowchart and game.

3. Assessment: students learn the algorithm by playing its game and judge their own progress through the winning/losing criteria of the game. Therefore, the assessment becomes part of the learning process.

4. (AVUSG) SOFTWARE SYSTEMS

Serious Algorithm Game Visualizer (Serious-AV) (Figure 4) is a visualization system that has been designed and developed to demonstrate (AVuSG) framework by providing several viewers and designers.

4.1 (Serious-AV) Viewers

(Serious-AV) supports the viewing and the playing learning processes by providing three viewers to enable the learner to view the algorithm text, flowchart, game demo, and game.

4.1.1 The Algorithm Text Viewer

A Windows Form that shows the algorithm text to the learner. It provides the learner with many options to manipulate the algorithm text such as copy, cut, paste, and print features.

4.1.2 The Algorithm Flowchart Viewer

A Windows Form that presents the algorithm flowchart to the learner. It provides the learner with many options to manipulate the algorithm flowchart such as copy, cut, paste, and print features.

4.1.3 The Algorithm Game Viewer

An XNA Window that displays the algorithm game to the learner to play the previously built game or watch its self-running demo.

4.2 (Serious-AV) Designers

(Serious-AV) supports the designing process by providing three types of designers for designing each form of the algorithm forms.

4.2.1 The Algorithm Text Designer

A Windows Form Application that simplifies the creation of the algorithm text by providing: a Text Editor to write the algorithm text, a File menu to save, open, and export the algorithm text to the Algorithm Text Viewer, and a Format menu to format the font; an Editing menu to cut, copy, paste, and delete the text (Figure 5).

4.2.2 The Algorithm Flowchart Designer

A Windows Form Application that simplifies the creation of the algorithm flowchart by providing: a Graphics Editor with a flowchart toolbox that has a set of drag-and-drop flowchart shapes, a File Menu to save, open, and export the flowchart to the Algorithm Flowchart Viewer, and an Editing Menu to copy, paste, move, resize, and delete a flowchart shape (Figure 6).

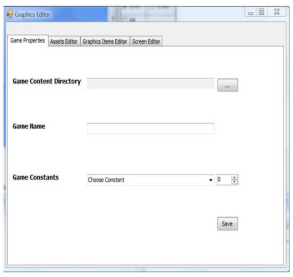
4.2.3 The Algorithm Game Designer

A Visual Studio Shell (Isolated Mode) that simplifies the development of an algorithm game and its demo by providing the following components (Figure 7):

- Code Editor: the main editor to develop the algorithm game using XNA framework and C# language.
- Script Editor: an editor to write game-specific script code that features IntelliSense-like programmer assistance.
- Developer Graphics Editors: four editors that support the creation of different components of an algorithm game using a flexible and user-friendly graphical user interface. The Properties (Figure 8) and Assets Editors (Figure 9) are both Windows Form Applications to enter the game properties such as Number of Lives and to load the game assets such as fonts, textures, and models. The Graphics Items and Game Screens Editors are both XNA Applications to support the creation of game graphics items and the game screens.
- The Algorithm Game Template: a blueprint for the creation of a new algorithm game.
- Serious Algorithm Visualization Game Engine (SAVGEngine): an XNA library that provides the new algorithm game by many useful components such as Play and Base game classes to handle the timing, loading, and rendering operations; basic game operations modules such as graphics, sound, input, physics, script, and game screens managers; and a Repository that has ready-to-use algorithm game components to be plugged-in into the new game.

5. ALGORITHM GAME DESIGN

The algorithm game features 2D and 3D representations of common data structures, a game-play that simulates the visualized algorithm operations, and sound effects that reinforce the game events. Any algorithm that has specific



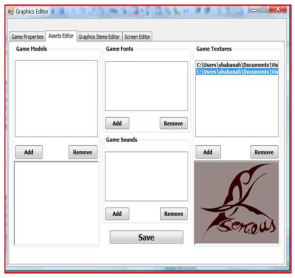


Figure 8: Properties Edit. Figure 9: Assets Editor

steps and a data structure can be visualized as an algorithm game. To create an algorithm game, the designer can either design a totally new game or modify the game-play of an existing game to simulate the algorithm steps. For example, the Binary Search Game (6.1) is created by modifying a known game called Pong to simulate the binary search algorithm. To simplify the development of algorithm games, for both the instructors and the students, they have been developed as XNA games. Since the Microsoft XNA Framework is a set of managed libraries based on the Microsoft .NET Framework that are designed for simplifying game creation for students and hobbyists. Moreover, the Algorithm Game Designer 4.2 has been designed to help in the design and the development of the algorithm games with as less code as possible.

5.1 Algorithm Game Motivation

Motivation theory (A.3) defines several guidelines that can be used to design algorithm games with internal motivation. First, the algorithm game should challenge the player by setting clear goals with appropriate difficulty levels and giving encouraging feedback. The information in the game should be complex and unknown to increase the player curiosity and imagination. The game must give the most control to the player by providing many customizing options. Moreover, it should encourage competition, collaboration and the recognition of peers.

5.2 Algorithm Game Design Elements

The following design elements [24] have been used to describe each algorithm game prototype:

- The Game Idea describes the game main goal and topic.
- The Game Start describes the game start up screen components.
- The Game Levels describes how the difficulty increases, how a level ends. Each completed level must achieve a learning sub-goal.
- The Game Milestone Events are points of the game at which the player rewarded or penalized.
- End of the Game explains what happens when the player loses, wins, or gets a high score.
- Game User Interface:
  - The Game Input is the player's contact with the game such as keyboard, mouse, and Xbox gamepad.

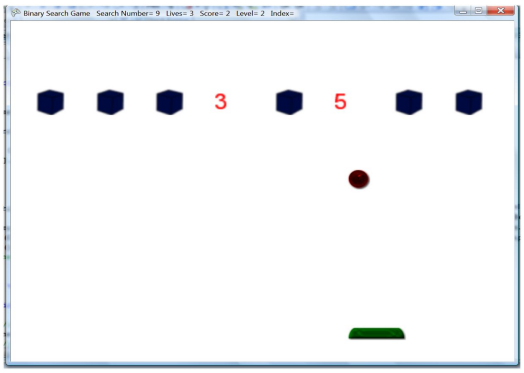


Figure 10: Binary Search- Play Screen

  - The Game Graphics are everything that contributes to the visual appearance of the game. The algorithm game graphics must depict the characteristics of its data structure, for example a block can be used to visualize one element of a data structure, while a set of blocks used to visualize an array.
  - The Game Sounds are either musical sounds that play at game goal events or sound effects that play at other game events.
  - The Game Screen is a collection of visual and audio components that describe the state of the game at any one time during the game life cycle. The basic game screens of every algorithm game are Title, Main Menu, Play, Won, and Lost screens.
- The Game Play explains how the game will be played and simulates the functionality of the algorithm.

6. ALGORITHM GAME PROTOTYPES

To explain how an algorithm game can be constructed for a given algorithm, three algorithm games prototypes are described.

6.1 Binary Search Game Prototype

The binary search algorithm finds the index of a specific value in a sequential list of sorted elements (array) as follows: it selects the middle element (median) of a sorted list and compares it with the (target value),

- if (median) > (target value), the middle element (index - 1) becomes the new upper bound of the list,
- else if (median) < (target value), the middle element (index + 1) becomes the new lower bound,
- else if (median) = (target value) then return the index of the middle element.

Then, it pursues this strategy iteratively for the new list bounded by the middle element; it reduces the search span by a factor of two each time, and soon finds the target value or else determines that it is not in the list.

6.1.1 Game Design Elements

• The main idea of the game is hitting an array of blocks with a ball using a paddle. The game is a mod of the Pong game.

• The game starts by displaying one game level that includes a group of blocks with their values hidden, a ball, a paddle, the Level Number, the Player Lives, and the Player Score.

• The game has several levels, at each new level the number of blocks is increased to make the game more challenging.

• The game ends when the player either loses all his lives or completes all the game levels successfully.

• The game milestone events are start of new level and a lost live.

• The game user interface includes game graphics items (array of cards, each card has a value to represent one element of the array, a ball, and a paddle), game sounds (HitBall, LostLive, Won, and Lost sounds), and game screens (Title, Main Menu, Play (Figure 10), Won, and Lost).

• The game play simulates the algorithm steps as follows:

1. The Player hits one block of the array with the ball using a paddle.
2. If the block is in the middle, the player scores one point.
   (a) If (search number> middle block value), the player plays on the right section of the array.
   (b) If (search number< middle block value), the player plays on the left section of the array.
   (c) If (search number== middle block value), the level ends.
3. If it is the last level and (Player Lives > zero), the player wins the game.
4. Else the level number is increased and the player starts new level.
5. If the block is not in the middle, the player loses one live.
6. If (Player Lives==0), the player loses the game.
7. Else the player repeats the same level.

6.2 Selection Sort Game Prototype

The Selection Sort algorithm sorts an array of numbers as follows: it finds the minimum value in the list, swaps it with the value in the first position, and repeats for remainder of the list (excluding the swapped elements at the beginning).

6.2.1 Game Design Elements

• The main idea of the game is sorting a group of dominoes in a fixed time according to the algorithm rules.

• The game starts by displaying one game level that includes a group of dominoes with their values shown, the Repeating Limit, the Level Number, the Player Score, and the Level Time.

• The game has several levels, at each new level the number of dominoes is increased, and the time is decreased to make the game more challenging.

• The game ends when either all the game levels are completed successfully or the Repeating Limit equals zero.

• The game milestone event is the start of new level.

• The game user interface includes graphics items (group of dominoes, each one has a value), game sounds (TimeEnds, Won, and Lost sounds), and game screens (Title, Main Menu, Play, Won, and Lost).

• The game play simulates the algorithm steps as follows:

1. The player chooses the smallest dominoes value and inserts it in its correct sorting place on the left.
2. If the player inserts the selected domino in incorrect place, the Player Score is decreased by one.
3. If the Level Time ends without sorting all the dominoes, the player loses the level, then:
   (a) If (Repeating Limit > zero), the player repeats the same level and the Repeating Limit is decreased by one.
   (b) Else the player loses the game.
4. If completes the level in the specified time, the player goes to the next level and the Player Score and the Level Number are increased.
5. If completes all levels on time, the player wins the game and the Player Score is displayed.

6.3 Insertion Sort Game Prototype

The Insertion Sort algorithm sorts an array of numbers as follows: it removes an element from the input data, inserts it into the correct position in the already-sorted list, until no input elements remain, and repeats for remainder of the list (excluding the elements in already-sorted list).

6.3.1 Game Design Elements

• The main idea of the game is sorting a pile of cards in a fixed time.

• The game starts by displaying one game level that includes a pile of unsorted, covered cards, the Player Lives, the Game Timer, the Repeating Limit, the Level Number, the Player Score, and the Level Time.

• The game has several levels, at each new level the number of cards is increased to make the game more challenging.

• The game ends when either all the game levels are completed successfully or the Player Lives equals zero.

• The Game milestone events are start of new level and a lost live.

• The game user interface includes game graphics items (array of cards, each card has a value), game sounds (LostLive, Won, and Lost sounds), and game screens (Title, Main Menu, Play, Won, and Lost).

• The game play simulates the algorithm steps as follows:

1. The player chooses one card at a time to be the key.
2. The player uncovers the chosen card to see its value.
3. The Player must compare the card with all the cards on the lift to insert it in its sorted place.
4. If inserts the card in incorrect place, the player loses one point.
5. If the Level Time ends without sorting all the cards, the player loses the level, then:

(a) If (Player Lives > zero), the player loses one live and repeats the same level.

(b) Else the player loses the game.

6. If completes the level in the specified time, the player goes to the next level and the Player Score and the Level Number are increased.

7. If completes all levels on time, the player wins the game and the Player Score is displayed.

7. CONCLUSIONS AND FUTURE WORK

(AVuSG) is an algorithm learning and visualization method that uses serious computer games to visualize algorithms. It benefits from the players desire to win, love to compete, and entertaining resulted from playing games to motivate students learning algorithms. Moreover, it facilitates the students' assessment using the winning-losing criteria of computer games without the need for external questions. In addition, the approach integrates learning theories with game design to introduce three educational models, the instructors can deploy in their classes to teach students algorithms.

(Serious-AV) system implemented the (AVuSG) framework with a goal to be used by both instructors and students. The instructor uses (Serious-AV) designers to design a text, a flowchart, and a game for the algorithm under study. The students use (Serious-AV) viewers to view their instructor designs. Depending on the deployed learning model, students may also create their algorithm text, flowchart, and game designs. However, game design and development are not easy tasks, so more attention is given to the development of the Algorithm Game Designer. It has several components and editors to automate and simplify the game development. The implementation of most of these components have been completed except for the Script, Screens, and Graphics Items editors. The Script editor will allow designers to design their games using a script language such as Iron-Python or Lua, where the Screens and Graphics Items editors will allow for the graphical design of several game components. Currently, the (SAVGEngine) game engine supports the creation of 2D algorithm games, but it will be extended to support 3D games. In addition, the algorithm game repository will be updated with more game components that support the design of different algorithm games. Serious-AV has one implemented algorithm game, the Binary Search game, but more games will be added later.

When the proposed system is fully implemented, a user study for the (AVuSG) approach can be conducted. The students can be divided into four groups, one of the three learning models will be applied on each group, and the fourth group will use the traditional instruction methods.

8. REFERENCES

[1] R. Baecker. Sorting out sorting: A case study of software visualization for teaching computer science. In *Software Visualization: Programming as a Multimedia Experience*, pages 369–381. The MIT Press, 1998.

[2] R. Baecker and D. Sherman. Sorting out sorting. 30 minute colour sound film, Dynamic Graphics Project, University of Toronto, 1981. Excerpted and reprinted in SIGGRAPH Video Review 7,1983.

[3] M. Bäsken and S. Näher. Geowin - a generic tool for interactive visualization of geometric algorithms. In *Revised Lectures on Software Visualization, International Seminar*, pages 88–100. Springer-Verlag, 2002.

[4] S. B. Bloom. *Taxonomy of educational objectives.* Pearson Education, 1984.

[5] M. H. Brown and R. Sedgewick. A system for algorithm animation. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 177–186. ACM Press, 1984.

[6] M. D. Byrne, R. Catrambone, and J. T. Stasko. Do algorithm animations aid learning? Technical Report GIT-GVU-96-18, 1996.

[7] E. Carson, I. Parberry, and B. Jensen. Algorithm explorer: visualizing algorithms in a 3d multimedia environment. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education,Covington*, pages 155–159. ACM Press, 2007.

[8] K. Crawford. Vygotskian approaches to human development in the information era. *Educational Studies in Mathematics*, 31:43–62, 1996.

[9] P. D. Eades and K. Zhang, editors. *Software Visualization*, volume 7. World Scientific, 1996.

[10] R. S. Feldman. *Understanding Psychology.* McGraw-Hill, 1996.

[11] G. I. G. Cattaneo and U. Ferraro-Petrillo. Catai: Concurrent algorithms and data types animation over the internet. *Visual Languages and Computing*, 13(4):391–419, August 2002.

[12] R. Gagne, L. Briggsm, and W. Wager. *Principles of Instructional Design.* New York: Holt, Rinehart & Winston, 3rd edition, 1988.

[13] S. Grissom, M. F. McNally, and T. Naps. Algorithm visualization in cs education: comparing levels of student engagement. In *SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization, San Diego*, pages 87–94. ACM Press, 2003.

[14] A. Hejdenberg. The psychology behind games. Gamasutra– Website, April 26, 2005. Accessed 10/10/2008, http://www.gamasutra.com/features/20050426/hejdenberg_01.shtml.

[15] W. Huitt and J. Humme. *Educational Psychology.* College of Education, 1999.

[16] C. Hundhausen. The "algorithms studio" project: using sketch-based visualization technology to construct and discuss visual representations of algorithms. In *HCC '02: Proceedings of the IEEE 2002 Symposia on Human Centric Computing Languages and Environments, Arlington*, pages 99–100. IEEE Computer Society, 2002.

[17] C. Hundhausen, S. Douglas, and J. Stasko. A meta-study of algorithm visualization effectiveness. *Visual Languages and Computing*, 13(3):259–290, 2002.

[18] C. D. Hundhausen and J. L. Brown. What you see is what you code: A radically dynamic algorithm visualization development model for novice learners. In *VLHCC '05: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric*

Computing, pages 163–170. IEEE Computer Society, 2005.

[19] H. Jenkins. The games to teach project. Comparative Media Studies-MIT –Website, 2001. Accessed 03/10/2008, http://www.educationarcade.org/gtt/proto.html.

[20] A. Korhonen. *Visual Algorithm Simulation.* Doctoral dissertation (tech rep. no. tko-a40/03), Helsinki University of Technology, 2003.

[21] T. W. Malone. What makes things fun to learn? heuristics for designing instructional computer games. In *SIGSMALL '80: Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems, Palo Alto*, pages 162–169. ACM Press, 1980.

[22] T. L. Naps. Jhave: Supporting algorithm visualization. *IEEE Computer Graphics and Applications*, 25(5):49–55, 2005.

[23] T. L. Naps, G. Rossling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. A. Velazquez-Iturbide. Exploring the role of visualization and engagement in computer science education. In *ITiCSE-WGR '02: Working group reports from ITiCSE on Innovation and technology in computer science education, Aarhus*, pages 131–152. ACM Press, 2002.

[24] M. Packard. A crash course in game design and production. Lord Generic Productions– Website, 1996-2001. Accessed 10/26/2008.

[25] C. Pelletier. The making of games project. London Knowledge Lab– Website. Accessed 03/17/2008, http://www.lkl.ac.uk/research/pelletier.html.

[26] J. M. Randel, B. A. Morris, C. D. Wetzel, and B. V. Whitehill. The effectiveness of games for educational purposes: a review of recent research. *Simul. Gaming*, 23(3):261–276, 1992.

[27] G. Rossling and B. Freisleben. Animal: A system for supporting multiple roles in algorithm animation. *Visual Languages and Computing*, 13(3):341–354, 2002.

[28] G. Rossling and T. L. Naps. A test-bed for pedagogical requirements in algorithm visualizations. In *ITiCSE '02: Proceedings of the 7th annual conference on Innovation and technology in computer science education, Aarhus*, pages 96–100. ACM Press, 2002.

[29] C. A. Shaffer, M. Cooper, and S. H. Edwards. Algorithm visualization: a report on the state of the field. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education, Covington*, pages 150–154. ACM Press, 2007.

[30] L. Stern, H. Sondergaard, and L. Naish. A strategy for managing content complexity in algorithm animation. In *ITiCSE '99: Proceedings of the 4th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education, Cracow*, pages 127–130. ACM Press, 1999.

[31] M. J. Wolf. *The Medium of the Video Game.* University of Texas Press, 1st edition, 2002.

APPENDIX

A. THEORIES AND MODELS

A.1 Learning Theories

Learning theories provide a conceptual framework for interpreting the examples of learning.

A.1.1 Behaviorism

The learner responses to environment stimulation in ways that increase or decrease the likelihood of the same response in the future [15].

A.1.2 Cognitivism

Learning is a complex process that utilizes problem-solving, insightful thinking, and repetition of a stimulus-response chain [10].

A.1.3 Constructivism

Constructivism states that human beings actively construct knowledge for themselves through active interaction with the environment and previous experiences. Vygotsky's social development theory focuses on the connections between people and the sociocultural context [8]. Constructivism influences learning in three ways:

- Curriculum: using curricula customized to the students prior knowledge.
- Instruction: encouraging students to analyze, interpret, and predict information.
- Assessment: assessment becomes part of the learning process so that students judging their own progress.

A.2 Models of learning

Models of learning attempt to examine and organize all the elements that contribute to learning in a systematic way that can easily be applied to learning situations.

A.2.1 Bloom's Model

Bloom defined a hierarchy of six objectives for any learning process: 1) Knowledge: remembering previously learned material; 2) Comprehension: grasping the meaning of the material; 3) Application: using learned material in new situations; 4) Analysis: breaking down material into its component; 5) Synthesis: combining parts to form a new whole; 6) Evaluation: judging the value of material [4].

A.2.2 Gagne's Model

Gagne, Briggs, and Wager have derived nine events of instruction that can be applied during a learning process: 1) gaining attention, 2) activating motivation, 3) stimulating recall of prerequisite learning, 4) presenting stimulus material, 5) providing learning guidance, 6) eliciting the performance, 7) providing feedback, 8) assessing the learner's performance, 9) promoting retention and transfer [12].

A.3 Motivation Theory

Motivation theory is concerned with the factors that stimulate or inhibit the desire to engage in a behavior. Malone and Lepper [21] distinguish between two types of motivation: Extrinsic that is supported by factors external to the activity and Intrinsic that arises directly from doing the activity. Some of the factors that enhance the motivation of the learner are individual since they operate even when a learner is working alone: challenge, curiosity, control, and fantasy. Other interpersonal factors play a role only when someone else interacts with the learner: competition, cooperation, and recognition [14].