

# An Example of Virtual Environment and Web-based Application in Learning



Mohamed Hamada

*Language Processing Lab, The University of Aizu, Japan.*

**Abstract**—Due to its importance as a model for several software and hardware applications, automata theory is a core topic in computer science and engineering education. But because of its abstract mathematical nature automata is used to be taught by traditional lecture-driven style. Virtual reality and web-based learning can provide automata learners with a unique experience such as interactive learning, simulation opportunities, visualizations of abstract concepts, etc. In this paper a web-based virtual environment for automata theory, as an example, is introduced in addition to an evaluation of its use in context.

**Index Terms**—Virtual reality, web-based learning, automata theory.

## I. INTRODUCTION

Virtual reality (VR) offers learners unique experiences that are consistent with successful instructional strategies such as hands-on learning, simulations, abstract topics visualization, etc. The virtual reality learning environment contains a multimedia information context that offers unique interactivity and can be adapted for individual learning styles. The following are some of the benefits of using virtual reality in learning.

- VR supports experiential learning where learners use more brain sensory during the learning process.
- VR supports active learning where learners are engage in the learning process more actively.
- VR supports collaborative learning where learners can communicate and share experience with each other in a virtual environment that simulates a classroom.
- VR allows learners to gain more control on their learning process.
- VR allows Teachers to act as facilitators not as knowledge transmitters. This means knowledge must be actively constructed by learners, not passively transmitted by teachers.

On the other hand web-based and active learning provides a powerful mechanism to enhance depth of learning, increase material retention, and get learners involved with the material instead of passively participate in the learning process. Web-based learning is currently a hot research and development area. The benefits of web-based learning are clear

at hand: learners can overcome the constraints of time, distance, and boundaries.

Learning science research indicates that engineering students tend to have active and sensing learning preferences, and engineering related educators are recognizing the need for more active and collaborative learning pedagogy [22]. So far several learning models have been developed (e.g. [4, 9, 13, 17]) for the realization of the learning preferences of science and engineering learners. Among these models, Felder-Silverman [4] is simpler and easier to implement through a web-based quiz system, as in Felder-Soloman [21]. The model classifies engineering learners into four axes: active vs. reflective, sensing vs. intuitive, visual vs. verbal, and sequential vs. global. Active learners gain information through a learning-by-doing style, while reflective learners gain information by thinking about it. Sensing learners tend to learn facts through their senses, while intuitive learners prefer discovering possibilities and relationships. Visual learners prefer images, diagrams, tables, movies, and demos, while verbal learners prefer written and spoken words. Sequential learners gain understanding from details and logical sequential steps, while global learners tend to learn a whole concept in large jumps.

In Rosati [20] a study of this model was carried out to classify the learning style axes of engineering learners. The study showed that engineering learners tend to have strong active, sensing, visual, and sequential learning preferences. The concepts in the theory of computation course have important use in designing and analyzing computational models of several hardware and software applications. These concepts are abstract in nature and hence used to be taught by a traditional lecture-driven style, which is suitable for learners with reflective preferences. Since computer engineering learners tend to have strong active preferences, a lecture-driven teaching style is less motivating for them.

In this paper, a research using web-based active and collaborative learning in the automata theory and related fields is presented. The twofold contribution of this work is a novel use of existing technology to improve learning and a longitudinal quasi-experimental evaluation of its use in context. As a first contribution, we introduce an integrated virtual environment that is designed to meet the active learning preferences of computer engineering learners. For the second contribution: several classroom experiments are carried out. The analysis of the experiments' outcomes and the students feed back show that our integrated virtual environment is useful as a learning tool.

Our integrated virtual environment (IVE) is designed to meet the active learning preferences for computer engineering

learners. IVE can be used as a supporting tool for active and collaborative learning not only for the automata theory course, but also for several other courses such as theory of computation, formal languages, discrete mathematics, computational models, principles of programming languages, compiler design and other related courses. Such courses cover a variety of topics including finite state machines (automata), pushdown automata, and Turing machines, in addition to grammars and languages. We cover such important topics in our integrated virtual environment. The covered topics will be written in Java as applets and then integrated into a single virtual environment using the Java2D technology of Sun Microsystems [10]. This implies that our virtual environment is portable, machine independent and web-based enabled, which makes it a useful tool as an interactive and online collaborative learning virtual environment.

The environment integrates several different materials to support the learners' preferred style. It includes a movie-like welcome component, an animated hyper-text introduction for the basic concepts, a finite state machine simulator with several operations, a set of visual examples for learners' motivation, and a Turing machine simulator.

To show the effectiveness of our integrated virtual environment as a model of interactive online collaborative learning tool, several classroom experiments were carried out. The preliminary results of these experiments showed that using our environment not only improved the learners' performance but also improved their motivation to actively participate in the learning process of the related subjects and seek more knowledge on their own.

The paper is organized as follows. Following the introduction, Section II discusses topics of automata theory. Section III introduces our integrated virtual environment and its components including finite state machines, visual machine examples, and Turing machines. The performance evaluation of the environment will be presented in Section IV. Some related work will be discussed in Section V. Finally, we conclude the paper and discuss the results and possible future extensions in Section VI.

## II. AUTOMATA THEORY

In this section we will briefly introduce the basic concepts of automata theory such as finite state machines, regular expressions, Turing machines and their properties.

Automata theory intersects with many other disciplines such as mathematics, computer science, engineering, linguistics, physics and biology as illustrated in Fig. 1.

### 2.1. Finite state machines

Finite state machines or automata represent mathematical models for several software and hardware applications. Informally, a finite state machine is a machine with a finite number of states and a control unit that can change the current machine state to a new state in response to an external effect (input). It has limited memory capabilities which make it a suitable model for applications that require no information about previous actions.

Finite state machines can, generally, be classified into two main categories: *transducers*, which are finite state machines with output, and *acceptors* (or *recognizers*), which are finite state machines without output but has a set of final states. Examples of transducers are Mealy machines which introduced by G. H. Mealy in 1955 [23] and (their equivalent) Moore machines which introduced by E. F. Moore in 1956 [24].

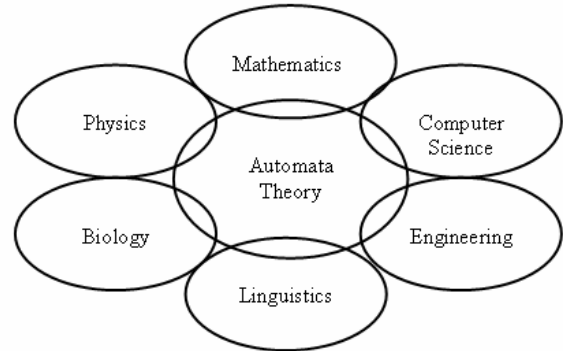


Fig. 1. Relationship between automata theory and other disciplines.

Depending on the way the machine controller responds to the input, finite state machines (acceptors) are classified into deterministic (DFA): if the controller can change from one state to another (one) state, in other words the controller can not be in more than one state at any one time. Nondeterministic (NFA): if it changes from one state to several states, that is, the controller can be in several states at once, and nondeterministic with empty move ( $\lambda$ -NFA): if (in addition to NFA) it can also change states in response to empty (no) input. In fact non-determinism in automata does not increase the expressive power of the automaton, but there can be substantial efficiency in describing an application using a nondeterministic automaton.

Formally, a finite state machine  $A$  is defined as a 5-tuple  $A = (Q, \Sigma, \delta, q_0, F)$ , where  $Q$  is a finite set of states that represent the machine states,  $\Sigma$  is the set of possible inputs to the machine,  $\delta$  represents the finite state machine controller,  $q_0 \in Q$  is the initial (starting) state of the machine, and  $F \subseteq Q$  is the set of possible final (accepting) states of the machine. Depending on how the machine controller  $\delta$  works, machines are classified into DFA, NFA, or  $\lambda$ -NFA.

- In a DFA,  $\delta$  is defined by  $\delta: Q \times \Sigma \rightarrow Q$ .
- In NFA,  $\delta$  is defined by  $\delta: Q \times \Sigma \rightarrow 2^Q$  ( $2^Q$  is the power set of  $Q$ ).
- In a  $\lambda$ -NFA,  $\delta$  is defined by  $\delta: Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$ .

A sequence of inputs is said to be *accepted* (*recognized*) by the finite state machine if the machine controller, starting from the initial state, scans all the inputs and stops at one of the final states. The class of languages that can be accepted by the finite state machine is called *regular* languages. In a DFA, two states  $p$  and  $q$  are said to be *equivalent* if: for all input strings  $w$ ,  $\delta(p, w)$  is a final state if and only if  $\delta(q, w)$  is a final state. Otherwise, the states are said to be *distinguishable*. If two states are equivalent, we can remove either of them without affecting the functionality of the underlying finite state machine. Generally,

for a set of equivalent states, we can replace them by any member of the set without affecting the function of the underlying finite state machine. This leads to the concept of finite state machine *minimization*. A finite state machine is said to be *minimum* if all of its states are distinguishable. Minimization of finite state machines is important in minimizing the cost of the application that the machine is modeling.

The three models of finite state machines DFA, NFA, and  $\lambda$ -NFA are equivalent. In other words, given any type of the machine, we can transform it to the other. By definition, we can see that  $DFA \subseteq NFA \subseteq \lambda\text{-NFA}$ , but we can transform  $\lambda\text{-NFA}$  to NFA and NFA to DFA (see Fig. 2).

## 2.2. Regular expressions

Regular expressions (RE) are important notation that are not automaton-like, but play an important role in the study of automata and their applications. Regular expressions can define exactly the same class of languages that the various forms of finite automata can describe. However, regular expressions offer something that automata do not: a declarative way to express the strings we want to accept.

Just as finite automata are used to recognize patterns of strings, regular expressions are used to generate patterns of strings. A regular expression is an algebraic formula whose value is a pattern consisting of a set of strings, called the language of the expression. Operands in a regular expression can be any of the following:

- Characters from the alphabet over which the regular expression is defined.
- Variables whose values are any pattern defined by a regular expression.
- $\lambda$  which denotes the empty string containing no characters.
- Null which denotes the empty set of strings.

Operators used in regular expressions include:

- Union: if  $R_1$  and  $R_2$  are regular expressions, then  $R_1 | R_2$  (also written as  $R_1 \cup R_2$  or  $R_1 + R_2$ ) is also a regular expression.
- Concatenation: if  $R_1$  and  $R_2$  are regular expressions, then  $R_1 R_2$  (also written as  $R_1.R_2$ ) is also a regular expression.
- Kleene closure: If  $R_1$  is a regular expression, then  $R_1^*$  (the Kleene closure of  $R_1$ ) is also a regular expression.

Equivalence between regular expressions and finite automata is shown in Fig. 2.

## 2.3. Turing machines

Turing machines (TM) are the most powerful finite state machines. They can simulate exactly what a digital computer can do. Informally, TM consists of a finite set of states and a controller that can read or write symbols on an infinite length tape. Unlike other finite automata, a Turing machine controller can move in both directions on the tape. The machine starts with an initial state, a finite number of input symbols written on the tape (all other infinite number of tape cells are blank), and the controller is set to the first input symbol from the left. According to the current state and the current scanned symbol

on the input tape, the controller takes the next move. It can overwrite the current scanned symbol or leave it untouched, change the current state, then move to either the left or the right, and so on. If no more moves are possible, then the machine *halts*. In some cases, the machine may run forever and never halt.

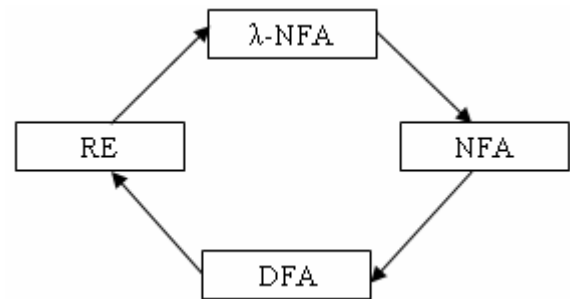


Fig. 2. Transformation between finite state machines and regular expressions.

Formally, a Turing machine  $M$  is defined as a 7-tuple  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ , where  $Q$  is a finite set of states that represent the machine states,  $\Sigma$  is the set of possible inputs to the machine,  $\Gamma$  is the complete set of tape symbols ( $\Sigma \subseteq \Gamma$ ),  $\delta$  is the transition function that represents the Turing machine controller,  $q_0 \in Q$  is the initial (starting) state of the machine,  $B \in \Gamma \setminus \Sigma$  is the blank symbol, and  $F \subseteq Q$  is the set of possible final (accepting) states of the machine. The Turing machines can be used as a language recognizer. In this case, if the machine halts with the controller in a final state from  $F$ , then we say that the initial input is *accepted* by the machine. The class of languages accepted by Turing machines is called *recursively enumerable* languages. But more interestingly, Turing machines can be used to compute functions, exactly the same way that modern digital computers can do. In this case, the function arguments are represented as sequences of 1's separated by 0's, and are written on the machine's tape. The function definition is represented as a set of rules suitable for the machine's transition function. Then, the machine works on that input. If it halts, the output symbols left on the tape represent the value of the function application on the arguments.

In this section we clearly see that the concepts of automata theory have abstract and mathematical nature and hence used to be taught by traditional lecture-driven style which is less motivating for engineering learners. In the next section we will introduce our web-based virtual environment which can be used as useful supporting tools for teaching and learning the abstract automata concepts in a more active way.

## III. WEB-BASED VIRTUAL ENVIRONMENT

We introduce an integrated virtual environment (IVE) to facilitate the active learning of automata abstract concepts and to meet the active learning preferences for computer engineering learners. IVE can be used as a supporting tool for web-based, active learning not only for automata theory course, but also for several other courses such as theory of computation, discrete mathematics, computational models, principles of programming languages, compiler design and

other related courses. Such courses cover a variety of topics including finite state machines (automata), pushdown automata, and Turing machines, in addition to grammars and languages. We cover such important topics in our integrated virtual environment. The web-based tools that cover these topics is written in Java as applets and then integrated into a single environment using the Java2D technology. This implies that our virtual environment is portable, machine independent and web-based enabled, which makes it a useful tool as an interactive online learning environment.

Our virtual environment contains several components which have been integrated into a single unit to make all topics easily accessible for learners. The components include the following: an animated (movie-like) welcome component, a hyper text introduction to the automata theory topics, a finite state machine simulator, a Turing machine simulator, a self assessment exercises, a chatting component for supporting online collaborative learning, and other components showing the visual examples of finite state machines. The welcome and introduction components use plain and animated text, which are suitable for learners with sequential learning preferences. The simulators and visual examples of components are best suited for learners with active and sensing learning preferences which most computer engineering learners prefer.

In designing our IVE learning tools we considered the active construction learning model [25,26] that has some basic design principles including the following.

- a. Teachers act as facilitators not as knowledge transmitters. This means knowledge must be actively constructed by learners, not passively transmitted by teachers.
- b. Assessment procedures should be embedded in the learning process and should consider learners' individual orientations.

In the sequel of this section, we will describe some of the components of our integrated virtual environment.

### 3.1 IVE Components

The integrated virtual environment and its components are shown in Fig. 3.

In Fig. 3 the numbers from 1 to 12 are described as follows.

- Number 1 points to the *Welcome* component which has a movie-like introduction to the topics.
- Number 2 points to the *Introduction* component which has the PowerPoint slides of the theory of computation topics.
- Number 3 points to the *Finite State Machine* (FSA) simulator (which is the current active IVE component in Fig. 3).
- Number 4 points to the *Turing Machine* (TM) simulator (which is the current active IVE component in Fig. 4).
- Numbers 5 to 7 point to the three of the visual examples which are currently integrated with IVE: *Tennis Game*, *Video Player*, and *Rice Cooker*, respectively (the first is the current active IVE component in Fig. 7).
- Number 8 points to the *Self-assessment* component.

- Number 9 points to an empty slot which can be used as a component in the future extensions of IE.
- Number 10 points to the Java2D editing options and memory monitoring.
- Number 11 points to the finite state machine (the current active component) editor, and number 12 points to the operations that can be performed on the edited machine.

The first two components of the environment introduce the principle ideas of the theory of computation. One component presents a short movie-like introduction that welcomes the learners to the topic. The other one is a rich hyper-text and power point slides that serves as introduction to the basic concepts. Learners can navigate through the component and learn about the basic concepts. The animated text is combined with an optional audio narration, which is convenient for learners who have difficulties reading text. It is also presented bilingually; in English and Japanese.

### 3.2 Finite State Machine Simulator

The finite state machine simulator is integrated as a basic component of the environment. It allows learners to draw an automaton visually and apply several operations to it. The possible operations include: NFA to DFA transformation,  $\lambda$ -NFA to NFA transformation, DFA to regular expression, and regular expression to  $\lambda$ -NFA. In addition to these transformations, learners can minimize the given automaton, check the acceptance/rejection of an input to the automaton, zoom-in and out, and auto layout the automaton. The simulator interface is shown in Fig. 3.

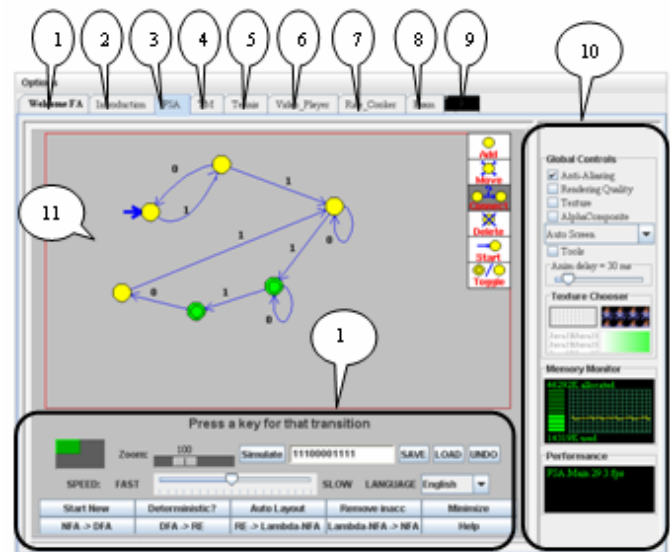


Fig. 3. The main interface of IVE (showing the finite state machine simulator as the current active component).

### 3.3 Turing Machine Simulator

The Turing machine simulator is integrated into the environment as well. Learners can write their machine in the input window, and then write the input of the machine on the (infinite) tape. After that, they can start to operate the machine on the input and observe how it works. For example,

to add two positive integers  $m$  and  $n$ , the function  $add(m, n) = m+n$ , is represented by the Turing machine rules shown in Fig. 5. A rule in the form  $a b c >$  means that if the current state is  $a$  and the current input tape symbol is  $b$ , then the controller changes the current state to  $c$  and moves one step to the right (right is represented by  $>$  and left by  $<$ ). A rule in the form  $a b c d$  means that if the current state is  $a$  and the current input tape symbol is  $b$ , then the controller changes the current state to  $c$  and the current input tape symbol to  $d$ . If the learner wants to add, for example, 2 and 3, i.e. compute the function  $add(2,3)=2+3$ , then he/she must write 2 as 11 and 3 as 111 separated by 0 on the input tape, which means the input string will be 110111. Running the machine on this input by clicking the run button will result in the machine halting with the output string 11111 written on the tape, which means 5. Learners can see the machine running on the input symbol in a step-by-step manner which can help the learner to see how the Turing machine acts on input and how it can compute functions. All operations of the Turing machines can be simulated by this simulator. The component interface showing the Turing machine simulator is shown in Fig. 4. While the Turing machine operates on its input, a number of short comments also appear on the editor to give the learners more information about the theory of Turing machines. The Turing machine simulator also includes sound effects to make learning more fun and interesting to learners.

In Fig. 4 the numbers from 1 to 7 are described as follows.

- Number 1 points to the machine's current state.
- Number 2 points to the machine's head, which points to an square on the tape.
- Number 3 points to the machine's infinite tape. The user can scroll the tape to the right or to the left by clicking on the right or left boundaries respectively.
- Number 4 points to the machine's editor. Users can edit for writing or modifying their Turing machines.
- Number 5 points to the comments from the simulator. These comments can assist the user during the simulation process.
- Number 6 points to the simulation operation buttons. Users can start running the machine, halt the running machine, simulate an input, etc., by clicking the appropriate button.
- Number 7 points to the current active TM simulator.

### 3.4 Visual examples

In our integrated virtual environment, a set of visual finite state machines examples are introduced with the aim of motivating learners in courses that include such topics. These selected examples represent useful daily life machines, games, and a puzzle. We have created six examples: an elevator, a vending machine, a man, a wolf and a goat puzzle, a video player, a rice cooker, and a tennis game. In this section, we will describe the last three examples which we have already integrated into our virtual environment.

#### 3.4.1 Tennis Game

Tennis is one of the popular games worldwide. It is interesting to show to learners that a tennis game could be modeled by automata (see Fig. 6). It enables them to learn

about automata while enjoy playing the game online. A typical tennis game can be represented by three finite automata: one for the points, one for the games, and one for the sets.



Fig. 4. The Turing machine simulator interface as the current component in IVE.

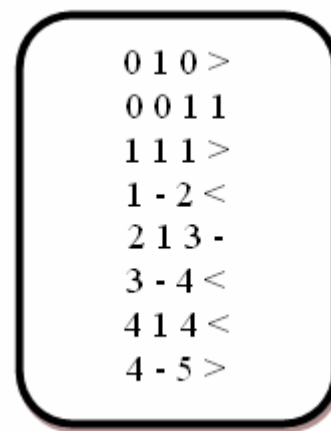


Fig. 5. A Turing machine example to add two positive integers.

Our tennis game simulator considers two players, A and B, who can be selected to play. It also allows auto play where the players play randomly. The simulator displays the score as well as the underlying three automata. The first automaton is related directly to the points; when a player wins a point, a new state is created. The second automaton is related to the game; when a player wins a game, a new state is created. The third automaton is related to sets; when a player wins a set, a new state is created. Fig. 7 shows a snapshot of the tennis game simulator interface. The game is completed when a final state of the automaton, representing the game set, is created. The winner is the player who reaches this final state first. The simulator also integrates animation of the two virtual players and various sound effects to make the learning more fun and interesting.

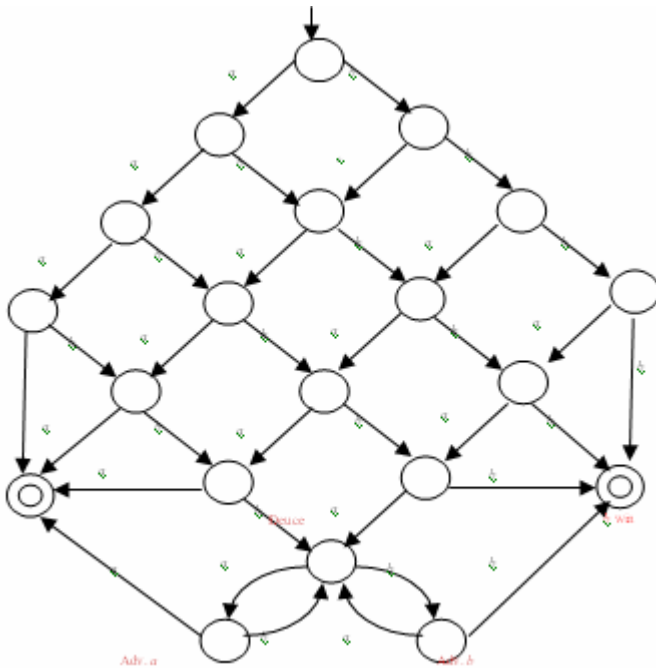


Fig. 6. An automaton representing tennis game.

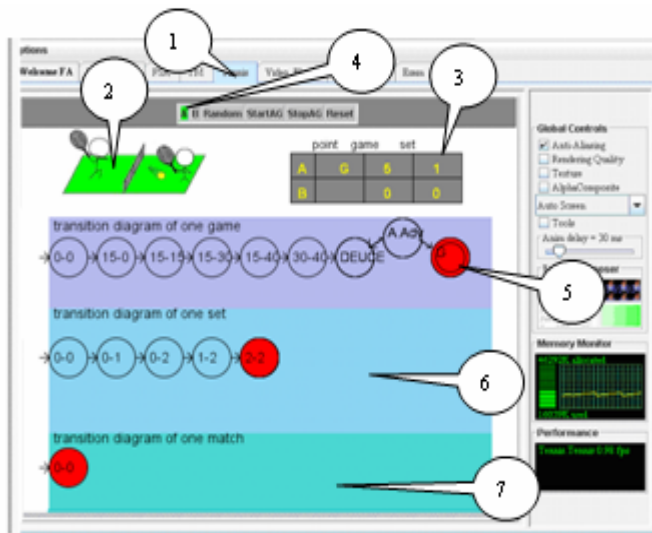


Fig. 7. The Tennis game simulator interface within the IVE.

In Fig. 7 the numbers from 1 to 7 are described as follows.

- Number 1 points to current active tennis game automata simulator simulator.
- Number 2 points to the virtual two players: A and B.
- Number 3 points to the score board simulator.
- Number 4 points to the current active player.
- Number 5 points to the automaton simulator that models the current game.
- Number 6 points to the automaton simulator that models the current set.
- Number 7 points to the automaton simulator that models the current match.

In the video player and rice cooker examples, the underlying automata are given first, and then when an operation takes

place, the corresponding state is highlighted. In the tennis game automata, however, we considered a different approach: the automata are created state by state in response to the corresponding operation. This is done intentionally to give the students a different taste of machine modeling by automata and to make it more interesting.

#### IV. EVALUATION OF PERFORMANCE

We carried out two experiments in order to evaluate the effectiveness of our integrated virtual environment tools on the learning process of engineering students. The first experiment evaluates the improvement in the students' motivation. The second experiment evaluates the effectiveness of using the tools on the students' performance. The purpose of introducing the visual automata examples is to enhance the students' motivation. To measure the effectiveness of these visual examples, we performed two experiments in the automata and formal languages course. The first one was for students who already completed the course; the sample population included 52 students who studied the topics in different classrooms. The following question was asked: "If the course was an elective course, would you choose to study it? And, do you recommend other students to study it?" Five options were given for responses: (a) don't know, (b) no, (c) maybe no, (d) maybe yes, and (e) yes. The responses were as follows: 3 answered a, 3 answered b, 6 answered c, 27 answered d, and 13 answered e. Then, we demonstrated our visual examples to the students and repeated the same question again. Their responses (after seeing the examples) were: 1 for a, 3 for b, 2 for c, 29 for d and 17 for e. Comparing the results from "Before" and "After" exposure to the examples, there was a slight improvement in motivation. For choices a, b, and c, if the number of responses decreased, it indicates a positive response, which is what occurred. While for the other choices d and e, the increasing number of responses indicates positive response, which also occurred.

We note that there was only a small improvement in the students' motivation, which is natural in this case because the students had already completed the course. In the next experiment we noted a better improvement in the motivation of students who were new to the course. In the second experiment, a total of 69 students were included, and they were all new to the course. The same steps, as with the pervious experiment, were repeated with a slight modification in the question. The question was "If the course was an elective one would you chose to study it?" As before, students were allowed to choose from among the five responses: a, b, c, d, and e. Their responses (before seeing the examples) were as follows: 22 answered a, 6 answered b, 10 answered c, 23 answered d, and 8 answered e. Next, we demonstrated our visual examples to the students and presented the same question to them again. Their responses (after seeing the examples) were as follows: 9 answered a, 4 answered b, 8 answered c, 34 answered d, and 14 answered e. Comparing the results "Before" and "After" exposure to the examples, we can see a better improvement in their motivation. As with the previous experiment, for choices a, b, and c, if the number of responses decreased it meant a positive response, which is what occurred. While for the other choices d and e, an increasing number of responses meant a positive response,

which also occurred.

We note that the motivation in the case of junior students (second experiment) was better than that of the senior students (first experiment). This result might be explained by the fact that the juniors had not studied the course before.

A preliminary study shows that the integrated virtual environment can improve the learning process of computer engineering students who study automata theory course and related courses. Last semester, the students were divided into four groups, each group containing 20 students. A set of 40 randomly selected exercises was distributed among the groups, 10 for each group. Each group members could collaborate inside their group but not with any other group members. No group could see the exercises of other group. Two groups were asked to answer their assigned exercises using the integrated virtual environment and the other two groups without using it. An equal time period was provided to all the groups. The result showed a better performance for the two groups using the IE. Then, the experiment was repeated by redistributing the exercises among the four groups. Again, the two groups with the IVE showed better performance.

## V. COMPARISON WITH SIMILAR WORK

There are a number of similar tools which have been developed (e.g. [1, 2, 3, 8, 16, 19]) to enhance the learning of automata theory topics. Most of them suffer from one or more flaws that make them less effective as a learning tool, particularly for less advanced students. For example, JFLAP [19] is a comprehensive automata tool but it requires skilled learners who already know the basics of automata to make full use of its rich operations. The automata tools in [16] are a powerful tool, but do not provide a convenient mechanism for displaying and visually simulating the finite state machines. The ASSIST automata tools in [8] are difficult to setup and use. The tools in [1] lack visual clarity and dynamic capability. Almost all have been designed as tools for advanced learners. These tools work on the assumption that the learners have already grasped the fundamental concepts. They are also dependent on advanced mathematical and idiosyncratic user interactions. On the contrary, our tools are designed as an easy-to-use, easy-to-learn, stand-alone, and all-in-one integrated virtual environment.

## VI. CONCLUSION

Applications of technology can provide course content with multimedia systems, active learning opportunities and instructional technology to facilitate education in the area of computer engineering to a broad range of learners. Such interactive course materials have already been introduced for several topics in computer engineering courses; see for example [5, 6, 7, 14, 15, 18].

In this paper, we followed the same path and introduced a set of visual tools to support interactive (e)-learning for automata theory concepts. It can also be used in other courses such as computational models, formal languages, language processing, theory of computation, compiler design, discrete mathematics, and other similar courses.

Through the results of our experiments, we also showed that our visual tools can enhance learners' motivation and performance. In addition an opinion poll showed a positive feedback on the environment tools from the students. In future work, we plan to enhance our visual tools by adding more features, more visual examples and games, and by performing more performance evaluation experiments.

## REFERENCES

- [1] H. Bergstrom. Applications, Minimization, and Visualization of Finite State Machines, *Master Thesis. Stockholm University*, 1998, related website at: <http://www.dsv.su.se/~henrikbe/petcl/>.
- [2] J. Bovet. Visual Automata Simulator, a Tool for Simulating Automata and Turing Machines, University of San Francisco, Available at: <http://www.cs.usfca.edu/~jbovet/vas.html>, 2004.
- [3] N. Christin, DFApplet, A Deterministic Finite Automata Simulator, Available at: <http://www.sims.berkeley.edu/~christin/dfa/>, 1998.
- [4] R. Felder and L. Silverman. Learning and Teaching Styles in Engineering Education, *Engineering Education*, vol. 78, no. 7, pp. 674-681, 1988.
- [5] S. Hadjerrouit. Learner-centered Web-based Instruction in Software Engineering, *IEEE Transactions on Education*, vol. 48, no. 1, pp. 99-104, 2005.
- [6] M. Hamada. Web-based Tools for Active Learning in Information Theory, to appear in the *ACM SIGCSE*, vol. 38, 2007.
- [7] M. Hamada. Visual Tools and Examples to Support Active E-learning and Motivation with Performance Evaluation, *Lecture Notes in Computer Science*, vol. 3942, pp. 147-155, 2006.
- [8] E. Head. ASSIST: A Simple Simulator for State Transitions, Master Thesis. State University of New York at Binghamton, 1998, related website at: <http://www.cs.binghamton.edu/~software/>.
- [9] N. Herrmann. The Creative Brain, Lake Lure, NC: Brain Books, 1990.
- [10] Java2D of Sun Microsystems [www.sun.com](http://www.sun.com)
- [11] Java Team. Buena Vista University, [http://sunsite.utk.edu/winners\\_circle/education/EDUHM01H/applet.html](http://sunsite.utk.edu/winners_circle/education/EDUHM01H/applet.html).
- [12] J. Keller. Development and Use of the ARCS Model of Motivational Design, *Journal of Instructional Development*, vol. 10, no. 3, pp. 2-10, 1987.
- [13] D. Kolb. Experiential Learning: Experience as the Source of Learning and Development, Englewood Cliffs, NJ: Prentice-Hall, 1984.
- [14] S. Li and R. Challoo. Restructuring an Electric Machinery Course with Integrative Approach and Computer-assisted Teach Methodology, *IEEE Transactions on Education*, vol. 49, no. 1, pp. 16-28, 2006.
- [15] J. Masters and T. Madhyastha, Educational Applets for Active Learning in Properties of Electronic Materials, *IEEE Transactions on Education*, vol. 48, no. 1, 2005.
- [16] M. Mohri, F. Pereria and M. Riley. AT&T FSM Library, Software tools, 2003, Available at: <http://www.research.att.com/sw/tools/fsm/>.
- [17] I. Myers. Gifts Differing, Palo Alto, CA: Consulting Psychologists Press, 1980.
- [18] R. Nelson and A. Shariful Islam. Mes-A Web-based Design Tool for Microwave Engineering, *IEEE Transactions on Education*, vol. 49, no. 1, pp. 67-73, 2006.
- [19] S. Rodger. Visual and Interactive Tools, Website of Automata Theory tools at Duke University, 2006, <http://www.cs.duke.edu/~rodger/tools/>.
- [20] P. Rosati. The Learning Preferences of Engineering Students from Two Perspectives, proceeding of *Frontiers in Education*, Tempe, AZ, pp. 29-32, 1998.
- [21] B. Soloman and R. Felder, Index of Learning Style Questionnaire, <http://www.engr.ncsu.edu/learningstyle/ilsweb.html>.
- [22] Transforming Undergraduate Education in Science, Mathematics, Engineering and Technology, in *Committee on Undergraduate Science Education*, Center for Science, Mathematics, and Engineering Education. National Research Council ed. Washington, DC: National Academy Press, 1999.
- [23] G. H. MEALY. A Method for Synthesizing Sequential Circuits, *Bell System Technical Journal*, vol. 34, no. 5, pp. 1045-1079, 1955.
- [24] E. F. MOORE. Gedanken Experiments on Sequential Machines, in *Automata Studies*, pp. 129-153, Princeton University Press, Princeton, New Jersey, 1956.

- [25] S. Hadjerrouit. Toward a Constructivist Approach to E-learning in Software Engineering, proceeding *E-Learn-World Conf: E-Learning Corporate, Government, Healthcare, Higher Education, Phoenix, AZ*, pp. 507-514, 2003.
- [26] G. Wilson, Ed., *Constructivist Learning Environments: Case Studies in Instructional Design*, Englewood Cliffs, NJ: Educational Technology, 1998.



**Mohamed Hamada** (PhD) is currently an Asst. Prof. at the software department, the University of Aizu, Japan. His research interests are: advanced learning technologies, natural inspired computations and programming languages. He is an IEEE and ACM member.