WEB-BASED SIMULATION OF AN ENGINEERING INSTRUCTIONAL LABORATORY

Constantinos Iliopoulos
University of the West of Scotland
United Kingdom

Anastasios Oikonomidis
Technological Education Institute of Piraeus
Greece

Abstract

The emergence of several kinds of electronic media and devices that connect to the Internet has created the opportunity for a wide variety of educators to provide different types of educational materials to their students/learners. It is still a considerable challenge to set up comprehensive, online engineering instructional laboratories to teach students online about the use of scientific instruments or processes linked with scientific experiments. Efforts for developing simulation applications that are meant to be delivered through websites have faced the inherent limitations of technologies like Adobe Flash and/or the requirement for the use of high cost development tools. The development of an online simulation system to facilitate learning of modern electronics is showcased and critically discussed. A case is made for the integration of Web 2.0 social computing elements and gamification principles in the developed online simulation system to enhance it further as a learning environment.

Online Simulation Systems in Engineering Laboratories

Engineering laboratories exist to offer students the opportunity to experience how theoretical concepts apply in the real world often by combining several processes commonly known as experiments. Through these experiments students often try to verify that what they have learnt in theory in terms of mathematical formulae, etc., can be applied in real life. In this way, students acquire practical skills like how to operate instruments and how to use devices and their accessories appropriately and safely, as well as experience how several substances change properties under different conditions (e.g., temperature, pressure), and many others. A key goal of every educational institution for engineers is to equip graduate engineers with the necessary skills both in theory and practice. Practice often takes place through labs equipped adequately to serve the educational needs of every learner. This means that there must be enough instruments, materials, working stations and staff in order for students to perform experiments, lab tests, practical exercises and so on effectively, and to develop practical skills. One small, physical instrument per twenty students

is likely not enough for all of them, particularly if it is really necessary to learn how it works.
This necessity forces institutions to spend sizeable budgets in an effort to offer their students education of high standards. Since the cost is often substantial even for high profile universities, efforts have been undertaken for its reduction through the use of simulation software. This kind of software is produced for both educational and research reasons and offers its users the opportunity to conduct experiments in their computers. Prominent examples of this category of software include MatLab® and Simulink® of MathWorks Corporation, covering a wide range of fields from mathematics to control systems and signal processing, LabVIEW® of National Instruments Corporation, covers fields like control systems and electronic circuits design and many more. These are the result of many years of research, with large groups of people who have worked hard for their initial development and still working for their maintenance and evolution. Unfortunately, such software products command high prices despite the fact that they can be acquired off the shelf. Using simulation software offered by the open source community solves the issue of cost. One major player in this field is Scilab of Scilab Enterprises offering versions for education suitable for teaching mathematics and engineering sciences at no cost.

All products mentioned above (as well as many others that also exist in the field of scientific simulations) are installed on individual computers, which means everyone involved (educators and learners) is asked to install a copy on their own computers. Educators teach learners about experiments and how they are conducted via software and learners work on them in order to produce several projects. Although such software often produces excellent results in the form of graphs, plots or log files there is one substantial drawback: students watch experimental procedures happening in computer monitors and they use graphical symbols to represent elements and instruments that look different from the real ones. A plain symbol of an oscilloscope usually found in simulation systems is nothing like a real oscilloscope where a number of knobs, potentiometers and buttons are required for its proper use. In the real world, only those students who will become researchers may be asked to use only simulation software and not the real thing. Those who will work on real world applications need hands-on experience and today's simulation software cannot provide it to them. Besides this drawback, it is not possible for educators to know how much time each student/learner spends on every project when working at home. Engineers are usually asked to work under pressure and tight deadlines, and engineering students need to learn how to adopt a culture of working effectively under a variety of time constraints. By using traditional, offline simulation software, students who perform experiments at minimum time may end up getting the same credit as other students in the same class who did the same job in double time. Traditional, offline simulation software does not seem to offer a consistent way of differentiating between the two types of student/learner in the previous example.

So, our proposal is to move from offline to online simulation software. In such environments both learners and educators will only need to have a web

browser and Internet connection. By adopting web technologies like JavaScript, PHP and HTML5 it is possible to skip the need to install plug ins to learners' browsers, and if the online simulation system is cross browser they can use the very same browser they are using for other online activities such as accessing their favorite social media websites. Of course, the use of such an online system can be charged the same way it is done for an offline one. Online systems are much more likely to be implemented by open source groups, academics or even other students who can share their work for free as it has been happening widely over the Internet in the last decade. Contributors can offer their parts to bigger projects and in this way it is possible to combine different modules implemented by different people. For example, one piece of simulation software for electronics can be produced by two groups of developers. One group could design and implement the various needed instruments and the second group could develop the electronic elements or components. Both instruments and elements could be re-used in other projects saving in this way a lot of effort. Parameters like the time of system use by each student can be easily added and passed to educators as feedback. Simply, learners/ students will log on through their unique user name and passwords so each time it will be recorded who is using the system, how much time is spent on specific learning activities and so on.

A key question with online simulation systems is how to provide comprehensive and convincing hands-on experiences to learners. In the following sections we will attempt to demonstrate how these needs could be addressed by using vector graphics, real instruments and elements/components as guides in order to produce their digital replicas.

JavaScript Web 2.0 Tools

In the early days of the Web, developers and users usually called everything that could be found online as "Web sites" or "Web pages". That was normal because most of the content was static, like plain text, images and maybe some video. During the late '90s when e-shops started to bloom, many people realized that the Web could be much more than static. While the e-shop business was getting more and more mature and social media was introduced in the mid '00s, the use of the Web itself was changing. This evolution is called Web 2.0. Several developments happened as part of this new Web 2.0 paradigm. For instance, the client side programming language JavaScript acquired a significant part in this new Web. Developers don't need any more to write code about everything like events or style change from scratch. Several JavaScript libraries appear online that do the job well, and programmers only have to load them through the HTML pages they create. Nowadays, the Web is inundated with such libraries and before anyone starts building anything online they'd better check first if a particular tool exists that already does what they want to accomplish. Fortunately, many of these libraries are free of charge, often available through open source communities, independent groups and individuals, even large corporations like Google and FaceBook. These libraries are categorized in different types such as Document Object Model (DOM) manipulation, Graphical User Interface (GUI), Visualization and more (List of JavaScript libraries, 2013). The large variety of fields that these JavaScript

libraries apply to, forced major Web players like Adobe's Flash to be left behind by many developers. Especially in the field of vector graphics on the Web, where a few years ago Flash was the unquestionably dominant force, many alternatives now have emerged. With the introduction of HTML5 and its feature of Canvas where vector graphics are embedded within the markup code, Flash (which is actually a third party plug-in for browsers) faces a serious competitor. The problem is that Flash is not a World Wide Web Consortium (W3C) recommendation and most likely it will never become one. The lack of browser native support for Flash, the growth of smart phones and tablets where there is even more limited and in some cases no Flash support at all has turned developers to lighter and most important cross-browser and platform solutions.

One of these solutions is the Raphaël Library (Baranovskiy, n.d. b) and has been developed by Dmitry Baranovskiy (Baranovskiy, n.d. a) a Sydney-Australia-based Web developer who is employed by Adobe. In the following section we are going to demonstrate some of the key principles for developing a rich graphic virtual world, comprising engineering instruments and elements, using Web 2.0 tools like Raphaël.

Building an Online Simulation System

The idea of building an online simulation system offering to students a level of experience that is close to the real thing begins with considering how the various parts of the system are depicted on screen. A simple square box tagged as "Oscilloscope" is far detached from the real oscilloscope device found in real laboratories. Expensive pieces of offline simulation software come with tagged plain boxes through which results of simulation processes are presented. But any engineer trained to draw mechanical or electrical drafts can create a digital replica having a very close resemblance with the real instrument, and at no cost. All that is required is time, proper guidance and a simple flat bed scanner as those found in any office nowadays. The first step involves placing real instruments on a scanner glass and creating high-resolution bitmap images of them at 1:1 scale. Of course, there are many instruments bigger than available scanners, but they can be scanned partially and be combined later in a single image, by using one of many image processing tools available online. In the following picture the result of a real oscilloscope scanned partially and combined later is depicted.

Figure 1. Bitmap image of a real oscilloscope

This image is used as a prototype for the creation of a vector graphics image that will look as close to it as possible. There are some commercial tools for

this task like Adobe Illustrator or Corel Draw. Fortunately, there is also an open source tool named Inkscape, which can do the same job. In either tool the scanned image will need to be imported in a new blank file. Before starting the drawing process, it should be determined which parts of the instrument will be movable at the final stage. In the above oscilloscope users should be able to rotate knobs determining VOLTS per DIVISION on the vertical screen axis and SECONDS per DIVISION for the horizontal axis. Also the POSITION potentiometers could be rotated so as to move traces on screen up, down or left, right. Anyone trying to build such a system will need to decide from the beginning which parts will be movable or not. The reason for this is because each such part should be assigned a unique identification, which will be used later in programming tasks. Through this identification, user interaction events can be attached to different parts. Inkscape comes with a full set of drawing tools creating Bezier paths around any object. After paths have been created by following the lines of the prototype image, they can be filled with the same colors as they were in the prototype by picking color samples for each part. The following picture exhibits the result of the above process.

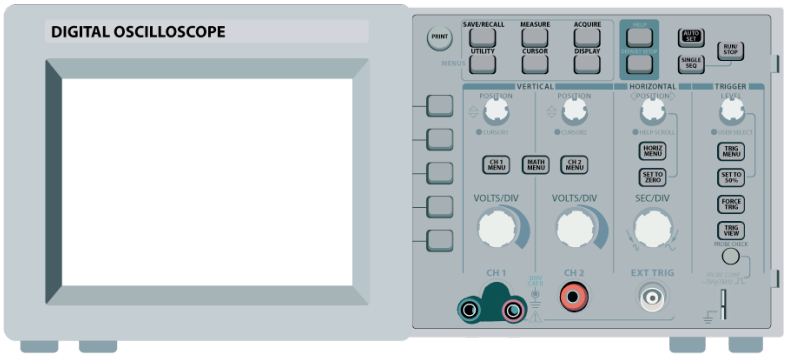

Figure 2. Vector Graphics image of the real oscilloscope.

By comparing Figures 1 and 2 we can easily conclude that resemblance between the real oscilloscope and its replica is very high.

The next stage involves the creation of a Scalable Vector Graphics (SVG) type file. SVG is "an XML-based vector image format for two-dimensional graphics that has support for interactivity and animation. The SVG specification is an open standard developed by the World Wide Web Consortium (W3C) since 1999 (Scalable Vector Graphics, 2013). The reason for doing this is for creating JavaScript code that will not only reproduce our replica in Web browsers, but it will also allow us to attach interactivity attributes to it, through the use of the Raphaël library. For the needs of our project a piece of PHP code was created to convert SVG to Raphaël code. For example, a plain circle in SVG is declared as follows: *<circle cx="100" cy="50" r="40" stroke="black" stroke-width="2" fill="red"/>* where *cx* and *cy* hold the coordinates of circle center, *r* the circle radius and *stroke*, *stroke-width* and *fill* the circle's attributes. The same circle in Raphaël could be described as: *var c = paper.circle(100, 50, 40).attr({"stroke": "black", "stroke-width": 2, "fill": "red"})* where paper is the main Raphaël object. Although interactivity could be applied directly on SVG code through its DOM, it is preferred to use

Raphaël because it offers cross browser support (not all browsers offer native SVG support) and most importantly it also offers much more simplicity. If only SVG is used and two oscilloscopes are needed to be on the same screen, then they should be both designed with the same level of detail. With Raphaël only one would need to be created through a JavaScript constructor and reproduced as many times as browser memory allows. Events for handling interactivity can be embedded within the constructor itself and therefore serve every oscilloscope object in the same way. This approach helps us to keep the code minimal and consistent.

After completing every part's behavior, such as events requiring rotation, change of color, repositioning, and so on, the next step involves implementing an algorithm that will make the simulation system not just look like the real one but work the same way, too.

Modified Nodal Analysis Algorithm and Gauss-Jordan Elimination

According to Eric Cheever (n.d.) professor of electrical engineering at Swarthmore College USA,

> Though the node voltage method and loop current method are the most widely taught, another powerful method is modified nodal analysis (MNA). MNA often results in larger systems of equations than the other methods, but is **easier to implement algorithmically on a computer, which is a substantial advantage for automated solution**. To use modified nodal analysis you write one equation for each node not attached to a voltage source (as in standard nodal analysis), and you augment these equations with an equation for each voltage source. (Cheever, n.d.)

By combining digital vector graphics replicas of electrical measuring instruments and elements / components with ***Modified Nodal Analysis*** (MNA) it is possible to implement a system simulating the operation of electrical and electronics circuits in a way which is very close to reality. MNA applies to circuits with passive (resistors), reactive (capacitors, inductors) elements and operational amplifiers. In a circuit with n nodes (excluding grounds) and m independent voltage sources MNA results in a matrix equation of the form Ax = z, where the A matrix is (n+m)x(n+m) in size, and consists only of known quantities, the x matrix is an (n+m)x1 vector that holds the unknown quantities (node voltages and the currents through the independent voltage sources) and the z matrix is an (n+m)x1 vector that holds only known quantities. To solve the circuit, a matrix manipulation of the form $x=A^{-1}z$ is needed. So the goal is to assign attributes to our graphics in order the system to distinguish them as resistors, capacitors, voltage sources, etc and construct the three matrixes. This can be done through JavaScript constructors. For example, a constructor resistor could have the following form:

```
function REScon(r, oc, ocv, cc, ccv){
    this.Rv = r - Math.floor(Math.random() * r * 0.05); // value with 5% tolerance
```

```
    this.Open = oc; // resistor node one graphic element
    this.OpenV = ocv; // resistor node one voltage
    this.Common = cc; // resistor node two graphic element
    this.CommonV = ccv; // resistor node two voltage
    this.type = 'res'; // type of element as resistor
    this.I = 0; // value of flowing current
  }
```

Let's assume that a graphic is created via Raphaël consisting of the resistor symbol and two circles on each "resistor" side. Users of this system should be able to "connect" this "resistor" with other elements. This is achieved by clicking with a mouse on each circle. An event puts one end of a graphic line to the circle center and by clicking on another circle the other end gets to that as well. This line serves as a connection wire. Several such lines exist as Raphaël objects allowing users to make various connections. Each circle serves as a connection node and is assigned to an "electrical" element via the following constructor:

```
function CONcon(o, kind)     // Variable, element id
        {       this.o = o; //circle object
                this.kind = kind; //kind of elements, as resistor, capacitor etc
                this.type = 'con'; //type of object as connector
                this.cx = 0; //X circle center
                this.cy = 0; //Y circle center
}
```

- First a Raphaël circle object is created: Circle1 = paper.circle(37.17, 328.148, 19.232).attr({"fill": "#000000", "stroke": "none", "stroke-width": 0, "cursor": "pointer", "opacity": 0});
- Then we assign it to a connector: Circle1Con = new CONcon(Circle1, ");
- We do the same for second circle and we create a resistor object: Resistor1 = new REScon(10000, Circle1Con, 0, Circle2Con, 0);
- At the end the attribute "kind" is entered for both circles: Circle1Con.kind = Resistor1; Circle2Con.kind = Resistor2;

Through this procedure each virtual connector is assigned to an electric element and its value. By clicking on each of them, these values and kinds are stored to an array. After each click, the system checks the connections array and tries to assign values to MNA matrices accordingly. When users have made correct connections, the A matrix is invertible. In the opposite case, connections are not correct, meaning that the "circuit" does not work due to open node(s). For passive elements and DC voltages this procedure is needed to be executed only once per connection. But when active elements (like capacitors) are used with voltages changing through time, it is required to execute matrix A inversion many times in order to get as many results as possible within a specific time range. The combination of time intervals and results assigned as attributes to a Raphaël path object returns the XY representation of node voltages through time. This is the exact same behavior as that of a real oscilloscope.

While values are assigned fast to matrices, the A matrix inversion is possible to last long for large circuits. This is a significant factor directly affecting system speed. A slow simulation procedure will force each user to abandon the system no matter how impressive are its graphics. The Gauss-Jordan Elimination algorithm (Adenegan & Aluko, 2012) offers the solution to this problem. The idea is to transform the original matrix to another one that is smaller and therefore easier (hence faster) to solve. To achieve that, the matrix should be transformed into diagonal form via elementary raw operations. If a zero is located in the diagonal rows, it must be switched until a non-zero is in the same place. If that is not possible, then the system has either infinite or no solutions. Finally, the division of a diagonal element and the right hand element in each row by the diagonal element in that makes each diagonal element equal to one (Adenegan & Aluko, 2012).

In the end, the combination of MNA and Gauss Jordan elimination algorithms within the graphics results in a system that is graphic rich, fast and accurate.

The Case of the Automatic Control Systems Lab

An online simulation system like the one described above has been implemented for the Automatic Control Systems (ACS) laboratory of the department of Automation, of the Technological Education Department of Piraeus Greece. It is accessible through the Web site http://auto-hsae.teipir.gr and serves attending learners/students and educators. Fundamental concepts of the control systems theory are taught through the lab courses and for that purpose there are six working stations. Each of them consists of an oscilloscope, a function generator, a digital multi-meter, a set of resistors, capacitors, potentiometers and operational amplifiers known as “Analog Computer” and a digital computer. Students are asked to perform a number of connections on the Analog Computer board and check results on the oscilloscope or multi-meter. The need to have an online simulation system was triggered by the increase of learner numbers while due to space and cost limitations it was not possible to create more working stations enough for all of them. So, three or four students had to work on a station that was built to serve only two of them. In addition, students were asking persistently for extra time to work on experiments. Some previous efforts to solve both problems by using simulation software such as SPICE (Quarles, Pederson, Newton, Sangiovanni-Vincentelli, & Wayne, n.d.) did not help them much. Students argued they didn’t just want to learn how circuits work but they also needed to learn to operate and work with the instruments themselves and that is not possible through SPICE. By using all of the tools and methods already mentioned above an online simulation system was created. The new system is identical to the real one and it is depicted in Figure 3.

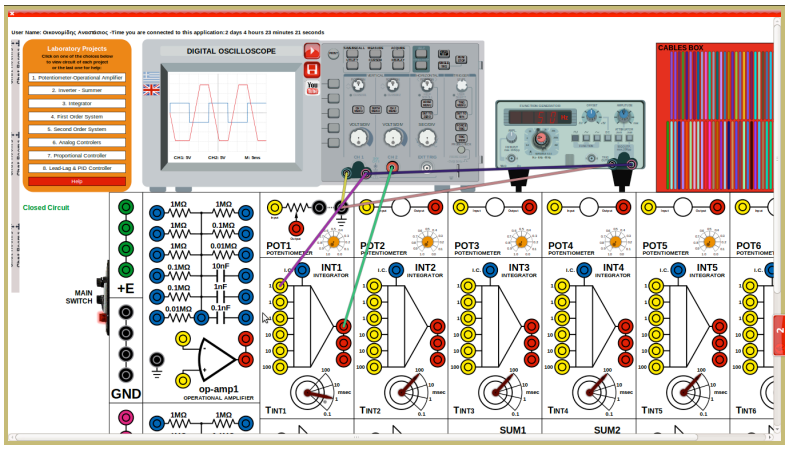

Figure 3. Online simulation system.

Development and implementation of it has gone through two previous versions and the current one allows students to work anytime it suits them. The most recent version of the system also enables students to talk to each other through an embedded real-time chat application. Preliminary feedback indicates that learner/student experience and satisfaction have increased significantly. Furthermore, the lab can now serve not only local but also distant learning students, too.

Next Steps: Towards Web 2.0 & Gamification

In this paper we have presented up to now issues concerning the use of simulations in education and more specifically in engineering education. Furthermore, we've presented important technological underpinnings of the simulation system that has been developed to help teach learners modern electronics. Finally, the ACS laboratory case focused on some of the issues linked with the deployment of the above system as well as key points in respect of learner feedback already received.

Reflecting on the development and actual use of the previously presented simulation system as it currently stands, coupled with the encouraging feedback we have received from actual learners, who have used the system, has motivated us to turn our attention towards enhancing it further, particularly from a learning standpoint. Creating a more integrated learning platform where different styles of learning are catered for, including social and peer learning makes academic sense and it is appealing to us, too. Also, creating a multifaceted platform where learning can be both fun and a challenge is also worth considering. Therefore, a review, analysis and evaluation of important developments in the fields of Web 2.0, gamification and online learning are ongoing.

Web2.0 is not solely concerned with new developments in fields such as JavaScript, AJAX, JSON, HTML5, PHP, etc., although they are very important and could open up new development possibilities for us. Web 2.0 also offers opportunities to create more comprehensive and integrated learning environments where learners can interact socially online and also learn from each oth-

er. Challenging learners in novel and exciting ways with an aim to enhance their learning creates new development opportunities.

Key aims for considering Web2.0 social computing enhancements and moving towards games-based learning and gamification are to enhance the motivation, engagement and experience of learners. We want to encourage students/learners to explore more comprehensively the previously introduced web-based simulated system and as a result enhance their learning.

Gamification is a relatively new concept, particularly when used within pedagogical, learning-oriented contexts. ***Gamification*** is defined as the use of game design elements and game mechanics in non-game contexts (Dominguez et al., 2013). Outside academia gamification has already been attempted in very diverse settings. For instance, in a business context Nike has used gamification principles on their website and has also created a host of additional applications, collectively named Nike+, to increase engagement and loyalty of web-site visitors and past Nike customers to the Nike brand and products.

Figure 4. Nike and gamification.

In gamified learning environments effort rather than total mastery of concepts, techniques, experiments etc. are often rewarded. Linking learning challenges to rewards and acknowledgement of achievements seem to be important underpinnings of modern, gamified environments. Appearing high on leaderboards, or being able to accrue virtual goods are additional incentives sometimes found implemented in gamified environments. Behaving in an altruistic manner and helping other learners accelerate their learning could also be considered as another development path suited to gamified and socially mature learning environments. Learners in gamified environments can also learn quickly to see failure as an opportunity to learn from, instead of becoming disillusioned, anxious or overwhelmed.

Gamified learning environments could also encourage desirable group behaviours and collective actions. These are qualities that need to be fostered and enhanced in academia in order to better prepare students/learners for successful future careers.

The above initial remarks signify that developments in the fields of Web2.0 and gamification are of relevance when it comes to creating more comprehensive and better integrated learning environments; therefore, they can be used to influence the future evolution of the above online simulation system.

Concluding Remarks

The development of the online simulation system presented earlier has been a considerable and multifaceted challenge. Advances and innovations in the fields of Web2.0 and gamification provide rich opportunities to develop more forward looking simulated environments and learning platforms.

Evolving learning environments, including the online simulation system presented in this paper, can make use of developments and lessons learned in the fields of Web 2.0 social computing and gamification to create more comprehensive and better integrated learning environments.

References

Adenegan, K. E. and Aluko T. M. (2012). Gauss And Gauss-Jordan Elimination Methods Forsolving System Of Linear Equations: Comparisons Andapplications. Retrieved from Journal of Science and Science Education, Ondo Vol. 3(1), pp. 97 – 105: http://josseo.org/_ebooks/Adenegan%20and%20Aluko%20p97-105.pdf

Baranovskiy, D. (n.d.a). Dmitry Baranovskiy's web log. Retrieved March 17, 2013, from http://dmitry.baranovskiy.com/

Baranovskiy, D. (n.d.b). Raphaël—JavaScript library. Retrieved March 17, 2013, from http://www.raphaeljs.com/

Cheever, E. (n.d.). Modified nodal analysis. Retrieved from http://www.swarthmore.edu/NatSci/echeeve1/Ref/mna/MNA2.html

Dominguez, A. E. et al. (2013). Gamifying learning experiences: Practical implications and outcomes. *Computers & Education*, *63,* 380-392.

List of JavaScript libraries. (n.d.). Retrieved March 17, 2013, from Wikipedia: http://en.wikipedia.org/wiki/List_of_JavaScript_libraries

Quarles, T., Pederson, D., Newton, R., Sangiovanni-Vincentelli, A., & Wayne, C. (n.d.). *The spice page*. Retrieved from Berkeley Wireless Research Center: http://bwrcs.eecs.berkeley.edu/Classes/IcBook/SPICE/

Scalable Vector Graphics. (n.d.). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Scalable_Vector_Graphics

Author Details

Constantinos Iliopoulos
Costas.Iliopoulos@uws.ac.uk

Anastasios Oikonomidis
anoikon@teipir.gr