

## 양자화학 교육을 위한 속웨이브 시뮬레이션 및 시각화

이 창 재\*

선문대학교 자연과학대학 화학과  
(2008. 9. 11 접수)

## Shockwave Simulations and Visualization for Teaching Quantum Chemistry

Chang Jae Lee\*

Department of Chemistry, Sun Moon University, Asan 336-708, Korea  
(Received September 11, 2008)

**요 약.** 양자화학에는 추상적이거나 비직관적인 개념이 많이 존재하여 학생들이 이해하기에 어려움을 준다. 다행히, 전통적인 강의실에서의 강의를 보조할 멀티미디어 웹 응용프로그램을 제작할 강력한 도구들이 웹 기술의 발전으로 많이 생겼다. 이 논문에서는 이 문제를 다루는 한 방안으로서 속웨이브 기술을 이용하여 웹 브라우저 상에서 쌍방향 시뮬레이션과 시각화를 동시에 해결하는 방법을 제시한다. 여러 예를 통하여 어떻게 속웨이브 기술의 특성을 이용하여 웹에 기반을 둔 학습 자료를 용이하게 개발하는가를 보여준다.

**주제어:** 웹기반 교수법, 디렉터, 속웨이브, 시뮬레이션, 시각화

**ABSTRACT.** Quantum chemistry has many concepts that are either abstract or counterintuitive, giving students difficulties in understanding them. Fortunately, the advancement of web technologies provides us with hordes of powerful tools to produce rich multimedia web applications that supplement traditional classroom teaching. In this paper we present an approach to address this issue that combines interactive simulations and visualization on the web browser using Shockwave technology. With several examples, we show how to take advantage of the features of the Shockwave technology in developing web-based course material with ease.

**Keywords:** Web-based teaching, Director, Shockwave, Simulation, Visualization

### INTRODUCTION

The progress of web technologies opened up new horizons for education. The education scene is no longer in the confines of textbooks and classrooms, and much educational material has been developed and delivered over the web. Using the web has a distinct advantage over classroom-only instruction: the instructor can program or write scripts to make the web page dynamically respond to student inputs such as key strokes and mouse clicks. Especially, interactive data visualization and simulation of natural phenomena can be utilized for the utmost benefit to students of science.

Creating interactive, multimedia courseware involves assembling text, graphics, audio, video and animation, and programming or scripting. Traditionally, Java has been the premier language for such supplementary online courseware production.<sup>1</sup> Typical scenario of Java-based course material may be depicted as shown in Fig. 1(a). The course material may consist of either a long single web page that can be scrolled or a set of web pages hyperlinked together. In this scenario the bulk of text explanations is in HTML format and interactive simulations are executed using one or more Java applets.

However, having multiple applets can make the

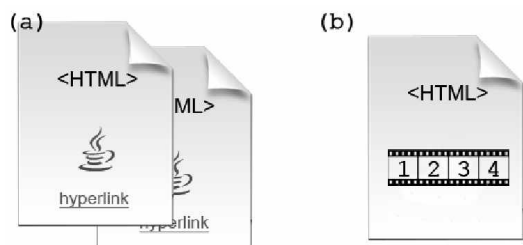


Fig. 1. Schematic diagrams of a course material with (a) Java applets embedded in web pages and (b) a Flash or Shockwave page with multiple frames.

web page load slowly, and navigating away from a web page and opening a new one that has applets also takes time. Consequently, it is often desirable to have a single web page and achieve what can be done with multiple applets and multiple pages. That can be done with some e-learning tools such as Flash, which is equipped with a built-in Timeline that allows the use of multiple pages--"frames" in Flash parlance (Fig. 1(b)). In addition, creating or controlling objects on the screen can be done visually, and more advanced controls can be done with Action Script, the Flash scripting language. Thus, Flash is a very powerful rapid application development (RAD) environment for courseware creation. With the missing of timeline in the integrated development environments (IDE's) coupled with limited visual programming, Java may not be the most efficient RAD environment for courseware creation.

Another authoring tool, relatively lesser-known but more capable than Flash, is Director Shockwave Studio (hereafter called Director for short).<sup>2</sup> Although Director started out as a tool mostly for creating multimedia pieces and games, nowadays a growing number of web sites for science education use it as a courseware development environment.<sup>3</sup> The Director IDE also features a timeline along with a multitude of intuitive drawing tools and a huge Library of widgets and pre-built codes for visual authoring, as shown in Fig. 2.

The visual development environment is quite powerful, so a completely new user to the tool can produce a high-impact educational material in a

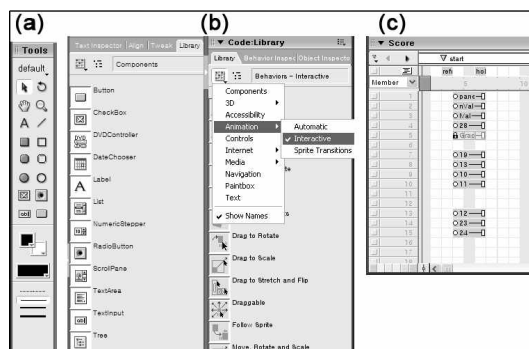


Fig. 2. Parts of Director's built-in graphic user interface (GUI). (a) Tools palette. (b) Library windows showing Flash Components and codes. (c) Timeline (Score). The numbers 1,2,3,... in (c) denote layers that objects on the screen (stage) occupy.

matter of hours without writing a single line of code.<sup>4</sup> For more versatility one can use the built-in scripting language, Lingo script,<sup>5</sup> which is a fully object-oriented language with not so a steep learning curve. Often scientific simulations require fast rendering of the results in real time. Director has a high-performance sprite rendering engine for both 2D and 3D graphics, as compared to Flash.<sup>2</sup> For 3D graphics Director can utilize either OpenGL or DirectX hardware-accelerated rendering. Flash (even the current CS4 version) lacks a true 3D engine. Another good thing about Director is that, while viewing a frame, other frames can be loaded in advance with commands such as preload and preloadNetThing, minimizing loading time. For these and other reasons, we have used Director as the main development tool for creating interactive simulations for educational purposes.

A work authored with Director is subsequently published for the web in a format called Shockwave. A Shockwave "movie" can be played with the Shockwave player-a web browser Plug-in similar to the Flash player. Currently, nearly 60% of web browsers have the Shockwave player installed, so the Shockwave format already has a large user base. If a browser does not have the player, it can be downloaded and installed readily.

We report in this paper how we use Director and Shockwave to do interactive simulations and visualization for teaching chemistry. Previous

works have been reported elsewhere.<sup>6,7</sup> Examples are chosen from topics in physical chemistry, especially on atomic and molecular structures but they can easily be extended to other subject areas. These topics are based on quantum theory, which is fraught with abstract concepts that are far removed from everyday experiences, causing students to be susceptible to many misconceptions.<sup>8-15</sup> In addition, quantum chemistry involves equations and mathematical expressions that are beyond the ordinary mathematical skills of many chemistry students. These factors contribute the students' loss of motivation to study quantum chemistry. McKagan et al. argue that visualization, interactivity, context, and effective use of computations are particularly effective for helping students understand the abstract and counterintuitive concepts of quantum mechanics.<sup>16</sup> There are several other reports about the effectiveness of the use of interactive visualization and simulations for teaching quantum mechanics.<sup>17-20</sup> As mentioned above, the Shockwave technology is an efficient RAD tool for creating interactive visualization and simulations. Thus, we believe that a report on such an implementation with Shockwave to ease student difficulties with quantum chemistry would be a valuable contribution to the literature.

In the next section, two-dimensional simulations and graphing with Lingo are discussed. Director comes with two approaches to drawing--the Imaging Lingo and the Vector Shape Lingo. The Imaging Lingo consists of commands and functions to manipulate each pixel of a bitmap. The Vector Shape Lingo commands manipulate vector shape objects with mathematical operations, so the results are resolution-independent as in Flash graphics. Either approach has its own strengths, and we will consider both of them. Then we discuss interactive three-dimensional simulations and visualization. Here, three examples are discussed: new surface plot technique for visualizing atomic orbitals, space quantization of angular momentum, and creation of a 3D viewer of molecules with external PDB (Protein Data Bank) files pulled from the web.<sup>21</sup>

## TWO-DIMENSIONAL GRAPHING WITH LINGO

### Hydrogen radial functions

As an example of graphing with Vector Shape Lingo, we consider the case of hydrogen wave functions. Here we are interested in plotting the radial part of the wave function,  $R_{n\ell}(r)$  and the corresponding radial distribution function,  $P(r) = r^2[R_{n\ell}(r)]^2$ . In Fig. 3 we show the Shockwave implementation in action. The radial parts of the wave functions are expressed in terms of associated Laguerre polynomials. They are analytically evaluated using Lingo, and then plotted as the user changes the values of the quantum numbers  $n$  and  $\ell$  and hits the "Plot Curves" button. In the figure, the curves, axes, and legends are drawn with Lingo. The code snippet attached to the "Plot Curves" button for generating a curve is something like the one shown below:

```
v = new(#vectorShape)
v.vlist = [ ]
repeat with x = 1 to 100
  y = (any valid math expressions)
  x = x + 1
  add vlist, [#vertex point(x,y)]
end repeat
```

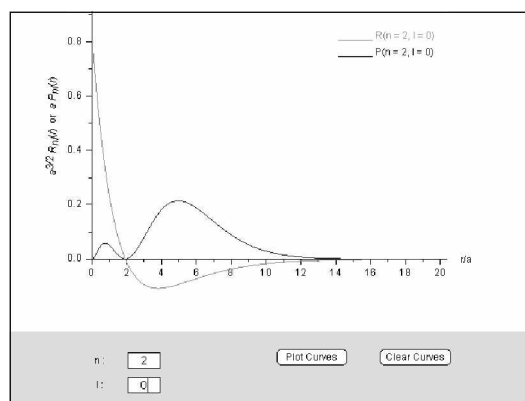


Fig. 3. Dynamic curve plotting with Lingo. The curves are generated by entering the values  $n = 2$  and  $\ell = 0$  in the text fields and pressing the "Plot Curves" button. If the screen is cluttered with many curves, one can press the "Clear Curves" button to clear the curves.

In the above code, plotting of a vector shape object  $v$  is done simply by assigning vertex points defined by an appropriate math expression to the object's vertex list property. The text fields and buttons in the gray user interface area of Fig. 3 are created simply by dragging and dropping them from Tools palette shown in Fig. 2(a), saving a lot of coding time.

Students can keep adding curves by changing  $n$  and  $l$  values and pressing "Plot Curves" button again. If wrong values are entered, an alert window pops up, warning the user to enter correct values. Generally, this type of dynamic plotting as parameters change enables students to explore firsthand the properties of functions, and hence underlying scientific principles, which is often hard to achieve with static graphics.

### Tomographic atomic density plot

For visualizing atomic orbitals many textbooks use isodensity surfaces,<sup>22</sup> which may mislead some students to think that atom has a well-defined shape and electrons are confined and uniformly distributed inside the surface.

Alternatively, electron densities are shown to emphasize the probabilistic nature of the atomic

world.<sup>23</sup> However, usually the students are not informed that the densities shown are those on a plane that passes through the nucleus, which may instill in students that atom has a fixed density distribution. These problems can be alleviated with tomographic density plots. Again, we take up the visualization of the hydrogenic orbitals to demonstrate our approach.

Figure 4 shows a screen shot of dynamical graphing of the electron probability density for the  $3d_{x^2-y^2}$  orbital. Here, the  $xy$  slice plane given at the bottom right screen is dragged along the  $z$  axis, and the density displayed on the top right screen is dynamically updated depending on the position of the slice plane. One can also click either the  $x$  or the  $y$  axis, and the corresponding slice plane appears at the mouse position and it can be dragged along that axis.

We generated the density plot with following code snippet:

```
colorScheme = rgb(func(x,y,z)*255,0,0)
(the stage).image.setPixel(x,y,colorScheme)
```

In the above, (the stage).image.setPixel(x,y,color reference) is an Imaging Lingo command to set the color of the pixel of the image object on the drawing area (the stage) at the point (x,y) with the color reference. The color reference is computed by assigning the density  $func(x,y,z) = |3d_{x^2-y^2}|^2$ , multiplied by 255, to RGB color values. The highest value of the function is normalized to 1, and it is multiplied by 255 so that it may have the brightest color. In the above example, we adjusted the color red only, but it is possible to change both green and blue colors as well, giving diverse coloring schemes.

The setPixel() command is intended for a very high-resolution density plot, since it manipulates each single pixel. If speed is more of an issue than the resolution, one may use another Imaging Lingo command (the stage).image.fill(rect, col), instead. The parameter rect denotes the bounding rectangle whose top left corner is the point (x,y) and bottom right corner is the point (x+dx, y+dy).

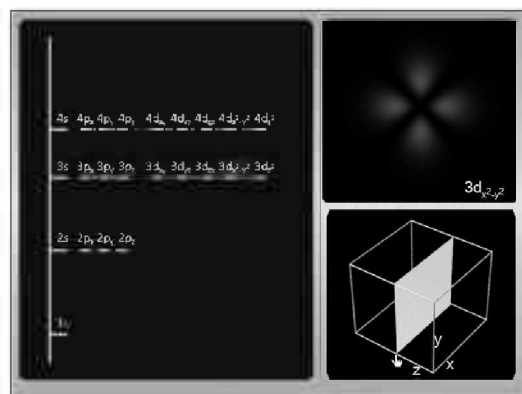


Fig. 4. Density plot of atomic orbitals. User selects an orbital by clicking on the orbital name on the left pane, and the selected orbital is highlighted with a purple color (Here  $3d_{x^2-y^2}$  is selected). Orbital density is dynamically updated on the top right screen, as the user drags the slice planes at the bottom right screen along either  $x$ ,  $y$ , or  $z$  directions.

where  $dx$  and  $dy$  are offset values in pixels. Since the fill command does not manipulate pixel by pixel, the required number of operations and hence the processing time can be reduced by a factor of  $dx \times dy$ .

### Simulation of Young's double-slit experiment

Director is a multimedia authoring program, and hence it shines when creating animations. In this subsection we discuss interactive animated simulation with Director. Many simulations in the literature are of Monte Carlo type, so we do the same here. The example we chose is a simulation of the Young's double-slit experiment that directly demonstrates the wave nature of matter.<sup>24</sup> When incoming electrons pass through a pair of slits and fall upon a fluorescent film, light bursts from the film exhibit an interference pattern. The interference pattern is given by

$$|\psi(x)|^2 \propto 1 - \cos\left(\frac{kdx}{L}\right), \quad (1)$$

where  $k$  is the wave number and  $d$  is the slit separation, and  $L$  is the distance between the slits and the detector screen.

Figure 5 shows the simulation in action. The "control panel" in the figure shows various GUI controls, which again have been dragged from the Tools palette of Director and dropped onto the stage. Users can adjust the wave number, the slit separation, and the screen distance by clicking on

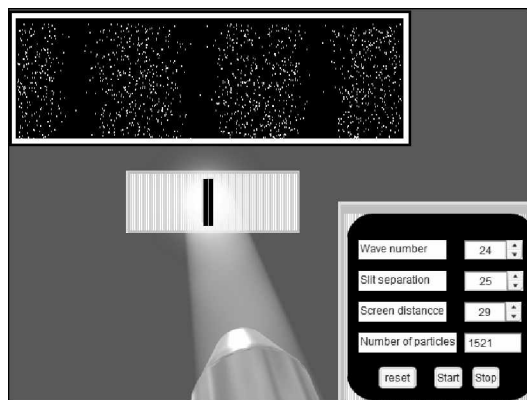


Fig. 5. Simulation of Young's double-slit experiment.

the NumericSteppers (the controls with up and down arrows), and start, stop, and reset the simulation with the corresponding buttons. A text box is provided to show the number of electrons fired.

The electron dots are generated by drawing small yellow circles with the following Imaging Lingo commands:

```
options=[#shapeType:#oval, #color:
        rgb(255,255,0)]
(the stage).image.draw(point1.point2.options)
```

In the above, point1 and point2 are positions of the top left and the bottom right corners of the bounding rectangle of the circle. The setPixel command could have been used in this case as well. However, then the dots generated would be too small to see. We avoid this by setting the size of the bounding rectangle to  $2 \times 2$  pixels.

We wanted to give students the illusion of gradual build up of the interference pattern. So, we used the frame action command `_movie.go(the frame)` that makes the playhead loop on a frame, and adjusted the frame rate to 30 and drew one dot per frame. Consequently, by this scheme students see an animation of 30 electrons per second being fired and landing on the film. The rejection method<sup>25</sup> is used to generate random deviates having the distribution function  $|\psi(x)|^2$  given by Eq. (1) for the position of the dots. This type of simulation technique may find applications in teaching a wide spectrum of scientific subjects that involve the kinetics or dynamics of many particles.<sup>26</sup>

## APPLICATIONS OF 3D LINGO

Another powerful feature of Director is its genuine 3D graphics capabilities. Three-dimensional scenes can be rendered using Software mode, OpenGL, or DirectX. The last two rendering methods take advantage of the power of 3D hardware graphics accelerator of the user's computer. Director has an extensive API for 3D Lingo

scripting, and we will discuss some applications of 3D Lingo in this section.

### Surface plot generation

One of the most popular 3D plot types is surface plot. The usual method for generating a surface plot consists of defining a data structure containing all the 3D points in the world coordinates, transforming these points to screen coordinates with transformation matrices, giving appropriate perspective, and drawing lines connecting the points. One also has to worry about the hidden line problem.<sup>27</sup> We take a quite different approach, taking advantage of 3D Lingo, and it cuts coding time significantly along the way. This technique is akin to terrain generation from height maps and basically involves two steps: (1) Generate a basic plane primitive with  $m \times n$  vertices (Fig. 6. Left). (2) Deform the plane mesh with data stored in the array vertexList (Fig. 6. Right) with the commands below.

```
myPlane.addModifier(#meshDeform)
myPlane.meshDeform.mesh[1].vertexList[i]
= my3dData[i]
```

Figure 7 shows the 3D surface plot of  $3d_{z^2}$  atomic orbital, generated with this method and wireframe rendered. If one wants to avoid the hidden line problem, the surface can easily be textured anyway one wishes. Furthermore, many operations to manipulate 3D models are available in the Code Library of Director. In order to view the model from different angles we dragged a "Rotate Camera Action" from the library and dropped it onto the model to rotate it while the left

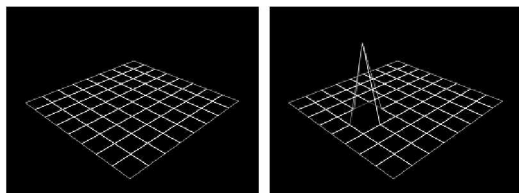


Fig. 6. Surface plot generation by mesh deformation. Left: Plane primitive. The plane corresponds to a data array initialized to zero. Right: Mesh deformation by a non-zero data point.

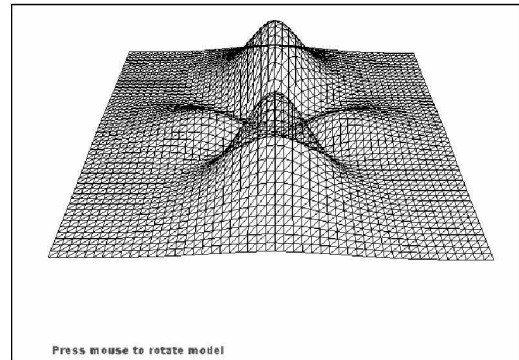


Fig. 7. 3D surface plot of  $3d_{z^2}$  orbital generated with the mesh deformation technique. The rotation of the model is achieved by dragging a code from the Code Library and dropping it onto the model.

mouse button is pressed.

### Classical and quantum angular momentum

A rotating or spinning object has angular momentum associated with the motion. In Fig. 8, the object is represented by the black-and-white ball and the angular momentum vector by the arrow. The blue sphere is the surface on which the tip of the angular momentum vector moves. Classically, if the axis of the spinning object has a constant inclination to the z-axis, the angular momentum vector will precess about the z-axis. In the previous frame of the simulation (not shown in Fig. 8) the user can drag the vector to change the inclination arbitrarily between 0 and 180 degrees, and see the animation of the precessional motion. However, quantized angular momentum cannot have arbitrary directions. Figure 8 shows that for the case of angular momentum  $L = 2$ , the projection of the vector along the z-axis can have only certain discrete values:  $L_z = m\hbar$ , ( $m = -2, -1, 0, +1, +2$ ). In this screen shot, the angular momentum vector jumped to an orientation corresponding to the button labeled as " $m = -1$ ". But the orientation in the xy-plane is completely unspecified due to the Uncertainty Principle. We designed the simulation to teach these facts. Thus, whenever the user clicks a button, the x and the y components of the vector are randomly oriented in the xy-plane, while the z component has a definite value. By

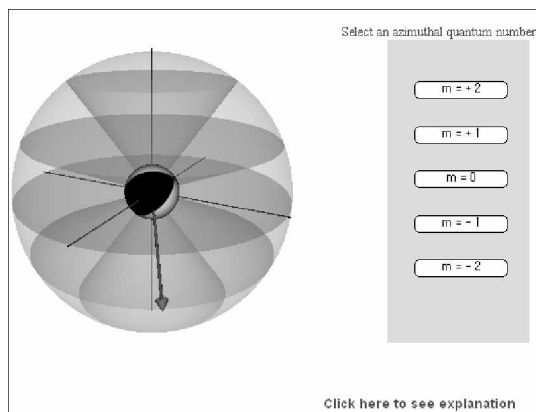


Fig. 8. Simulation to teach Space Quantization of angular momentum. If the user clicks on a button, the vector jumps to a position with a definite  $z$  value but random  $x$  and  $y$  values.

playing with this simulation, students can learn the concept of Space Quantization of angular momentum.

### PDB file viewer

The Protein Data Bank (PDB) is a repository and a file format of experimentally determined 3D structural data of molecules. A PDB file contains atomic coordinates, bibliographic citations, primary and secondary structure information, and may include other informations. In this section we discuss how we built a 3D viewer for molecules, whose structural data is given in the PDB format.

The PDB viewer uses some Net Lingo commands to download PDB files from the web. A part of the PDB file for the C540 Fullerene molecule, which consists of 540 carbon atoms, is shown below :

```
atom 1 - C  CA - 0-0.9502501
           -5.701202  11.05954  3 2 a 5 a 6 a
atom 2 - C  CA - 0-0.9502534
           -4.316603  11.05995  3 1 a 3 a 9 a
atom 3 - C  CA - 00.366632
           -3.888721  11.05994  3 2 a 4 a 12 a
atom 4 - C  CA - 01.180695
           -5.008749  11.05927  3 3 a 5 a 15 a
```

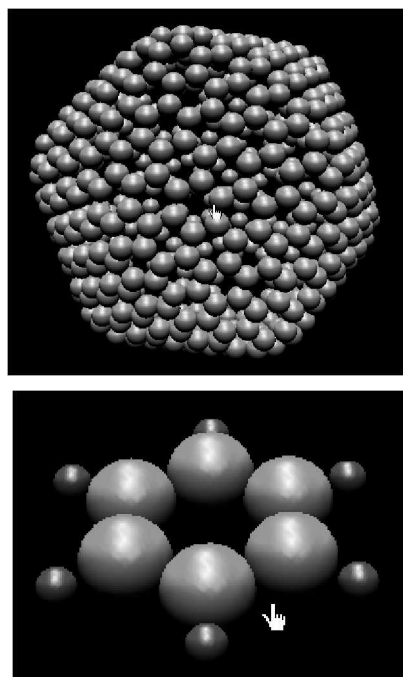


Fig. 9. Interactive rendering of a Fullerene C540 molecule (top) and a Benzene molecule (bottom). Carbon and hydrogen atoms are represented by green and red balls, respectively. Users can rotate the models and zoom in and out on the screen.

In this version of the PDB viewer not all informations in the file are needed. The informations used are the types of the atoms (whether they are carbons or hydrogens, etc.), the positions of the atoms in 3D space, and connections among the atoms.

The atoms are represented with colored balls and there is an option whether to view the bonds between the atoms or not. In Fig. 9 we show a C540 Fullerene molecule on the left and a Benzene molecule on the right, with bond viewing turned off. Users can drag the model to rotate it and zoom in and out to explore the structure at will. These creation of these controls are again facilitated with the Code Library.

### CONCLUDING REMARKS

The incorporation of interactivity is known to enhance the learning process.<sup>28</sup> We believe that

teaching science concepts, which is often abstract, would benefit even more from interactive simulations and visualization. However, the production of such interactive courseware should not tax busy science educators. It is desirable either to use a library of routines for such purposes or to use a RAD tool such as Director we used to produce the simulations reported here. We managed to build the most complex simulation in less than 100 lines of Lingo code, which is quite compact as compared to other development tools.

Showing the results of the simulations graphically in real time as they proceed is a powerful technique for enhancing students' intuitive understanding of science concepts and processes occurring in nature. The examples in this paper are largely selected to show atomic and molecular structures that may be used in physical chemistry courses. However, the simulation and visualization techniques can be applied to teaching concepts and chemical processes in other chemistry disciplines (and, for that matter, other sciences) that students may find difficult to understand and visualize in mind.

Over the years we found that some students have low self-motivation to study, and even the interactive simulations and visualization were not so effective for such students. This situation let us to create game-like virtual environments, using Director as the main development tool along with several other complementary softwares, and embed physical chemistry course materials in them.<sup>29</sup> The end-of-semester evaluation for the course was very encouraging, and we are working on creating more engaging educational materials. Incidentally, there appeared an article on creating educational games with VPython scripting language for use in physics courses.<sup>30</sup> Such an approach will undoubtedly enhance physics education substantially. Director comes with a physics engine (Havok in the previous versions of Director and Ageia PhysX in the current version), so similar or more elaborate physics-based virtual worlds can be created rather easily with Director and Lingo as well.

## REFERENCES

1. Christian, W.; Belloni, M. *Physlets: Teaching Physics with Interactive Curricular Material*; Addison-Wesley: Reading, U. S. A., 2001.
2. [http://www.adobe.com/products/director/resources/integration/flash/quick\\_comparison.html](http://www.adobe.com/products/director/resources/integration/flash/quick_comparison.html).
3. <http://phys.educ.ksu.edu/vqm/index.html>.
4. Cumararatne, C. *Computers in Physics* 1998, 12(5), 413.
5. Since 2004, Director also supports JavaScript syntax.
6. Lee, C. J. In *Proceedings, 19th International Conference on Chemical Education, Sookmyung Women's University*, Seoul, Korea, Aug 12-17, 2006, p 136.
7. Lee, C. J. In *Handbook & Abstracts, The 10th Asia Pacific Physics Conference, Postech*, Pohang, Korea, Aug 21-24, 2007, p 218.
8. McKagan, S. B.; Perkins, K. K.; Wieman, C. E. *Phys. Rev. ST Phys. Educ. Res.* 2008, 4(2), 020103.
9. Singh, C. *AIP Conf. Proc.* 2007, 883, 185.
10. Sadaghiani, H.; Bao, L. *AIP Conf. Proc.* 2006, 818, 61.
11. Styer, D. F. *Am. J. Phys.* 1996, 64(1), 31.
12. Johnston, I.; Crawford, K.; Fletcher, P. *Int. J. Sci. Educ.* 1998, 20(4), 427.
13. Birk, J. P.; Kurtz, M. J. *Journal of Chemical Education* 1999, 76(1), 124.
14. Boo, H. K. *Journal of Research on Science Teaching* 1998, 35(5), 569.
15. Harrison, A. G.; Treagust, D. *Science Education* 1996, 80(5), 509.
16. McKagan S. B.; Perkins K. K.; Dubson M.; Malley C.; Reid S.; LeMaster R.; Wieman C. E. *Am. J. Phys.* 2008, 76(4&5), 406.
17. Singh, C. *Am. J. Phys.* 2008, 76(4), 400.
18. Trey, L.; Khan, S. *Computers & Education* 2008, 51(2), 519.
19. Keller, C. J.; Finkelstein, N. D.; Perkins, K. K.; Pollock, S. J. *PERC Proceedings 2006*; AIP Press: Melville, New York, U. S. A., 2006, p 121.
20. Wieman, C. E.; Perkins, K. K.; Adams, W. K. *Am. J. Phys.* 2008, 76(5), 393.
21. <http://www.wwpdb.org/>
22. Goldberg, D. E. *Fundamentals of Chemistry*, 5th ed.; McGraw-Hill: New York, U. S. A., 2007, p 114.
23. McQuarrie, D. A. *Quantum Chemistry*; University Science Books: Sausalito, U. S. A., 1983, p 232.
24. Tonomura, A.; Endo, J.; Matsuda, T.; Kawasaki, T.; Ezawa, H. *American Journal of Physics* 1989, 57(2), 117.
25. Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; Vetterling, W. T. *Numerical Recipes in C: The Art*



- of Scientific Computing*, 2nd ed.; Cambridge University Press: Cambridge, U. K., **1992**.
26. Kim, S.-G.; Kim, J.-R.; Woo, K. H. *Journal of the Korean Chemical Society* **1999**, *43*(4), 469.
27. Ammeraal, L. *Computer Graphics for Java Programmers*; Wiley: Chichester, U. K., **1998**.
28. Evans C.; Gibbons, N. J. *Computers & Education* **2007**, *49*(4), 1147.
29. Lee, C. J. In *Proceedings, 19th International Conference on Chemical Education, Sookmyung Women's University, Korea, Aug 12-17, 2006*, p 306.
30. Lee, J. W. *Physics & High Technology* **2007**, *16*(9), 48.
-