Programming physics softwares in Flash

Koo-Chul Lee[a,*], Julian Lee[b]

[a]Department of Physics, Seoul National University, Seoul, Republic of Korea
[b]Department of Bioinformatics and Life Science, Soongsil University, Seoul, Republic of Korea

Abstract

We discuss various aspects of programming physics education software in Adobe Flash. Since the authoring environment for Flash is initially developed for non-programmers, it is easy to learn even for those having no previous knowledge in programming language, although some previous experience in programming may help in mastering advanced features of ActionScript, an object-oriented programming language used for developing Flash programs. The most attractive feature of Flash is its strong graphic capability not available in other standard programming languages.

Key words: physics education; computer simulation

1. Introduction

For many years, physics-education researchers repeatedly showed that traditional introductory physics courses with passive-student lectures are of limited value in enhancing conceptual understanding of the subject[1,2]. Also, recent researches indicate that the use of interactive engagement strategies can increase the effectiveness of conceptually difficult courses well beyond that obtained with traditional methods [2–8].

The interactive physics education can be most effectively implemented by web-based computer simulations[3,7]. In fact, the advent of computer not only revolutionized the physics research by creating a new discipline of computational physics, it also enabled effective implementation of interactive education through development of various simulation softwares[9].

The interactive education through computer simulation also makes it possible to teach material to students which used to be considered far beyond their grasp, since there is no demand for elaborate mathematics, in contrast to the approach based on analytical solution of physics equations. Many papers have made the point that the introduction of computers into physics teaching changes not only how the subject is taught, but also what is taught[7,8].

In this work, we discuss various aspects of developing interactive physics education software using Adobe Flash. Flash was originally developed by Macromedia as a simple animation tool for the designers of web pages. Starting from Flash MX (version 6), Macromedia introduced an extensive set of ActionScript, an object-oriented programming language which provides the developer with a fine control over the behavior of the software. The end product of Flash can be published though web and thus can be run on any platform. The most attractive feature of Flash is its strong graphic capability not available in other standard programming languages. Also, since the authoring environment was initially developed for non-programmers, it is easy to learn Flash even for those having no previous knowledge in programming language. Also, starting from ActionScript 3.0, compiler is available for free.

* Corresponding author.
Email address: dcknsk@yahoo.com (Koo-Chul Lee).

Flash software is distinct from others in that the basic building block is a frame, which represents visual graphic at each instant of time, separated by fixed time interval which is typically 1/12 seconds. Thus Flash software is most suitable for simulation of dynamical equations. In the following, we will demonstrate some examples of interactive simulations based on Flash that enable students to understand basic concepts in physics.

2. Bus Simulator

One-dimensional kinematics is explained at the beginning of virtually any course on introductory physics. The bus simulator demonstrated in this section helps students develop understandings on basics concepts in one-dimensional kinematics such as instantaneous velocity or acceleration. The figure 1 shows the snapshot of the simulation. The anima-

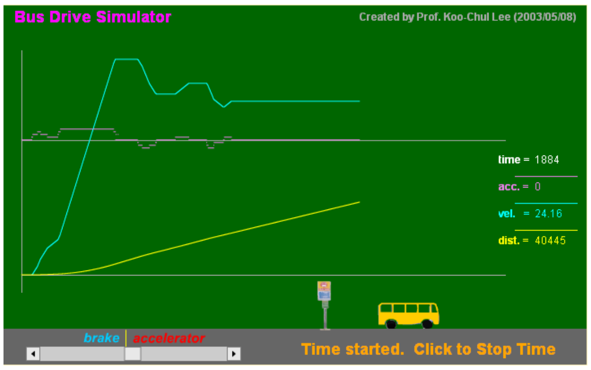

Fig. 1. A snapsnot of the Bus Simulator

tion starts and stops by clicking the button represented as a text at the bottom, and the motion of the bus is controlled by accelerator and brake, represented as a slide. As the motion continues, graphs of various quantities such as velocity and acceleration are plotted as functions of time.

In the ActionScript the method **button_mc.on Release** is executed at the time the button is clicked. This method contains various commands for initializing graphics, but the most important command is

id1 =setInterval(runthis, 10);

where the function setInterval repeatedly executes the function passed as the first argumnt, the function **runthis**. The second parameter specifies the time interval, which is 10 mili-second in this example. The function **runthis** is defined below:

```
1: function runthis (){
2: tt++;
3: xx = tt*0.2;
4: sPosition = driver.getScrollPosition();
5: acc = accunit *(driver.getScrollPosition()-40);
6: if ((sPosition ==0)&&(vel !=0)) {acc *= 20;
   brakeSound.start();}
7: vel = vel + acc;
8: if (vel < 0) {vel =0; driver.setScrollPosition(minPos
   + (maxPos - minPos)/2); }
9: if (vel > velmax) { vel = velmax;
     driver.setScrollPosition(minPos + (maxPos -
   minPos)/2);}
10: yy += vel;
11: pos = pos + vel*0.5;
12: bus._x= pos;
13: ptA = [xx+20, 149.25-acc*150]; ptB = [xx+20,
    150.25-acc*150];
14: curve_mc.lineAB(ptA, ptB, 1, violet);
15: ptA = [xx+20, 299.5-vel*8]; ptB = [xx+20,
    300.5-vel*8];
16: curve_mc.lineAB(ptA, ptB, 1, cyan);
17: ptA = [xx+20, 299.5-yy*0.002]; ptB = [xx+20,
    300.5-yy*0.002];
18: curve_mc.lineAB(ptA, ptB, 1, yellow);
19: if (pos > 575 ) { pos -=575; }
20: if(pos < -25) pos +=575;
21: write(dtime , tt, 590, 162, 59, 20, white, "_sans",
    12, false);
22: write(dacc , acc, 590, 192, 59, 20, violet, "_sans",
    12, false);
23: write(dvel , doubleFormat(vel,2), 590, 222, 59,
    20, cyan, "_sans", 12, false);
24: write(ddist , doubleFormat(yy,0) , 590, 252, 59,
    20, yellow, "_sans", 12, false);
25: updateAfterEvent(); }
```

The time **tt** is incremented at the 2nd line, which is proportional to the horizontal position of the current points plotted in the graphs, **xx**, as specified at the 3rd line. The lines 4-5 determine the acceleration **acc** from the position of the slide, **sPostion**. The 7-th line increments the velocity **vel** by the acceleration, and the line 10 increments the position **yy** by the velocity. The horizontal coordinate of the bus in the animation, **pos**, is obtained from **yy** by a linear transformation, as specified at the line 11.

As can be seen from the code, it is rather straightforward to implement the basic ideas in physics using the object-oriented language ActionScript.

3. Orbits of celestial bodies

The motions of celestial bodies are governed by Newton's laws of motion and law of gravity, which are expressed as a differential equation, just like any other dynamical equations in physics. Since analytic solutions for differential equations are found only in exceptional cases, there has always been a strong tendency in traditional physics textbooks to include material which is capable of being solved with relatively simple mathematics, giving the students wrong impressions that most of the physical problems can be handled easily with analytical methods. However, in most of the cases, differential equations have to be solved using numerical methods.

Denoting the number of celestial bodies as n, the dynamical equations satisfies by these objects can be written as a first-order differential equation in $2n$-dimensional space,

$$\begin{aligned}\frac{d\mathbf{x}_i}{dt} &= \mathbf{v}_i \qquad (i = 1, \cdots n)\\ \frac{d\mathbf{v_i}}{dt} &= -G\sum_{j\neq i}\frac{m_j(\mathbf{r_i}-\mathbf{r_j})}{|r_i-r_j|^3} \qquad (i=1,\cdots n)\end{aligned} \tag{1}$$

Analytic solutions can be found for $n = 2$, but no analytic solution exists for $n > 2$, and one must resort to numerical methods. In fact, a first-order differential equation of the form

$$\frac{d\mathbf{y}}{dt} = f(t, \mathbf{y}) \tag{2}$$

can be solved using the 4nd order Runge-Kutta (RK4) method[10], that finds the approximate solution:

$$\mathbf{y}(t+\Delta t) = \mathbf{y}(t) + \Delta t(f_1 + 2f_2 + 2f_3 + f_4)/6 \tag{3}$$

with

$$\begin{aligned}f_1 &\equiv f(t, \mathbf{y}(t))\\ f_2 &\equiv f(t+\Delta t/2, \mathbf{y}(t) + \Delta t f_1/2)\\ f_3 &\equiv f(t+\Delta t/2, \mathbf{y}(t) + \Delta t f_2/2)\\ f_4 &\equiv f(t+\Delta t, \mathbf{y}(t) + \Delta t f_3),\end{aligned} \tag{4}$$

which is accurate to the order of $O(h^4)$.

Since the motions of celestial bodies reduce to two-dimensional motions for the two-body problem, it can be described by orbits on the computer screen. A snapshot of the two-body simulation is shown in figure 2, where the motions of the celestial bodies represented by small circles are generated in real time using RK4 method, and compared with the orbits obtained by analytic solution.

In the ActionScript source code for the simulator, the function **runthis** is repeatedly executed with the command **setInterval(runthis, 10)**, just as in the case of Bus Simulator. The function **runthis** in turn calls the method **Ode.rk4(funcArrHO, t, xv, h)** in order to generate numerical solution of the differential equation at each instant of time, using RK4 method.

The function that is input to the method Ode.rk4 is the Newton's law of gravity in this case, but the method is general enough to incorporate any form of force law.

The software can be exposed to students in several ways depending on their levels of achievements. On the most elementary level, students are allowed to interact only with the animation itself, in order to get a feeling on how good a numerical solution can perform compared to analytical method. For a more advanced class of students, the source code itself can be shown, in order to teach the numerical method for solution of differential equations.

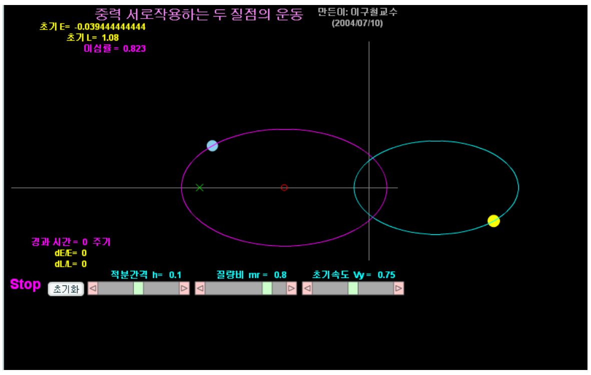
중력 서로작용하는 두 질점의 운동
만든이: 이구철교수
(2004/07/10)
초기 E= -0.039444444444
초기 L= 1.08
이심률 = 0.823
경과 시간 = 0 주기
dE/E= 0
dL/L= 0
적분간격 h= 0.1
질량비 mr = 0.8
초기속도 Vy = 0.75

Fig. 2. The snapshot of the animation played by the software, which simulates the motions of two celestial bodies that interact with gravity.

An example of the equation with no analytic solution is the equation (1) for $n > 2$. The animation captured in figure 3 was generated by a software describing the motion of eight celestial bodies in the solar system, consisting of the Sun, Mercury, Venus, Earth, Mars, Jupiter, Saturn, and the Moon. The software was created using ActionScript 3.0, but the content of the source code is essentially same as the previous one, the only difference being that RK4 method is applied to eight-body problem that has no analytic solution.

We note that the motions of the bodies described by the equation (1) are no longer two-dimensional for $n > 2$, and the only thing we can draw on the computer screen is a two-dimensional projection of the three-dimensional motions, as shown in figure 3. In order to make the users perceive three-dimensional illusion, various visual effect such as gradients and shadows are used. In this example, an imaginary plane is placed at the $x - y$ plane, and the shadows of the bodies are displayed.

Visual effects such as this will enable students to mentally reconstruct the three-dimensional motion of relatively simple objects such as celestial bodies, but will be of limited power for describing motions of complex three-dimensional objects such as bio-molecules. Obviously, the development of educational softwares for describing such objects awaits the progress in hardware technology leading to the wide-spread use of stereographic screen or viewers, and the development of softwares for creating truly three-dimensional animation in the entirely new environment.

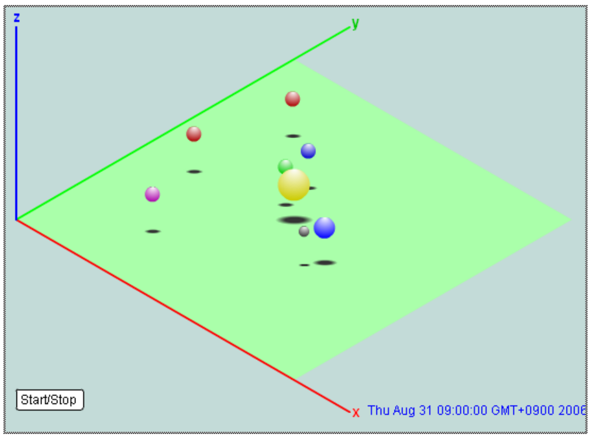

Fig. 3. The snapshot of the animation played by the software, that simulates the motions of eight celestial bodies in the solar system.

4. Summary

In this work, we have demonstrated some of the interactive physics education softwares developed using Flash. The object-oriented nature of the ActionScript used in Flash authoring environment makes it easy for unexperienced programmers to develop interactive education softwares, and the powerful graphic ability enables students at the user-end to get involved with interest and enthusiasm. Also, by exposing the source code to students with higher levels of achievements, numerical approaches for solving physics problems can be taught. We expect that interactive software will be developed with Flash also in various advanced fields in physics such as classical mechanics, electromagnetism, thermodynamics, quantum mechanics, special and general relativity, biophysics, so on, so that students can grasp relevant basic concepts intuitively.

Acknowledgements

This work was supported by the Soongsil University research fund.

References

[1] L.C. McDermott, E.F. Redish, Am. J. Phys. 67 (1999) 755.

[2] R.R. Hake, Am. J. Phys. 66 (1998) 64.

[3] Z. Zacharia, O.R. Anderson, Am. J. Phys. 71 (2003) 618.

[4] D.E. Meltzer, K. Manivannan, Am. J. Phys. 70 (2002) 639.

[5] R.A. Burnstein, L.M. Lederman, Physics Teacher 39 (2001) 639.

[6] E.F. Redish, R.N. Steinberg, Physics Today 52 (1999) 24.

[7] I.D. Johnston, Am. J. Phys. 64 (1996) 245.

[8] E.F. Redish, The Conference on Computers in Physics Instruction, E.F. Redish, J.S. Risley, eds., Addison-Wesley, Redwood Ca, 1988.

[9] K.-C. Lee, How to teach statistical thermal physics in an introductory physics course, http://phya.snu.ac.kr/ kclee/howto/; Understanding basic concepts of thermodynamics by throwing dice, http://phya.snu.ac.kr/ kclee/dice/.

[10] W.H. Press, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes, Cambridege University Press (1988) 704.