Teaching Dynamic Systems & Control with a Video Game

B. D. Coller
Northern Illinois University, DeKalb, Illinois, USA
coller@ceet.niu.edu

Abstract: *Dynamic Systems & Control is one of the most difficult courses to teach in the mechanical engineering curriculum. The subject is very mathematical and the mathematical framework is unfamiliar to novice students. Recently we began using a video game to demonstrate and teach content of the course. The game provides a natural way to align instruction with constructivist theories on how people learn. Herein, we describe the game and present preliminary results demonstrating its effectiveness.*

Introduction

Most undergraduate mechanical engineering curricula in the United States have a required third or fourth-year course on dynamic systems & control (DS&C). The theory and topics we cover in the course lie at the heart of much of modern technology. The principles provide the underlying theory which guides commercial airliners in for safe landings during howling storms. This theory forms the "brain" of industrial robots which, via jujitsu-like maneuvers, weld the body of a Lexus together. And when we launch a spacecraft, the theory gives us a high degree of certainty that the craft will arrive in Saturn's orbit exactly 6 years, 8 months and 7 days later. Broadly speaking, students who successfully complete the introductory DS&C course should demonstrate the following outcomes:

- Derive differential equations that model a broad class of mechanical systems.
- Determine stability of these systems and their temporal characteristics directly from the mathematics.
- Create feedback loops with sensors, actuators and computing elements which stabilize the system or alter its dynamics in favourable ways.

When we instructors see the collection of mathematical concepts, tools, algorithms, and theorems that make up the course, we understand it. We see how all the pieces fit together to form a coherent whole. We see the utility. We see the limitations. We see the symmetries, and we see the beauty in the equations.

This is very different from what our students see. When they encounter the equations in rapid-fire succession, they are often overwhelmed by the Tsunami. The mathematics is unnatural for them. For our mechanical engineering students, it is the first time that they are required to actually use Laplace transforms. To replace the concrete and easily understood variable "time" in one's equations with a complex Laplace variable that represents a combination of exponential growth and oscillation frequency seems counterintuitive. Our mechanical engineering students are unaccustomed to thinking of dynamic systems as input/output systems that can be chained together like components of a stereo.

When students encounter such situations, they often resort to coping mechanisms. They treat the mathematics as a set of thought-free operations that can be combined into recipes and committed to memory. Obviously, this is not what we want our students to get out of the course. Yet, the strategy often suffices to achieve a passing grade.

Dissonance Between How We Teach and How People Learn

One of the most widely accepted and empirically confirmed models of how people learn is that of Constructivism. That is, human learning is *constructed*. Learners build new knowledge, based upon the foundation of previous learning:

> New information is filtered through mental structures (schemata) that incorporate the student's prior knowledge, beliefs, preconceptions and misconceptions, prejudices and fears. If the new information is consistent with those structures it may be integrated into them, but if it is contradictory, it … is unlikely to be truly incorporated into the individual's belief system – which is to say, it will not be learned. (Prince & Felder, 2006).

When one examines popular controls textbooks (e.g. Franklin, Powell, and Emami-Naeini, 2006; Ogata, 2009; Dorf and Bishop, 2008), however, one finds a dominant axiomatic and deductive style that seems to assume that students are empty vessels that simply need to be filled with knowledge. Most homework problems in the texts are purely mathematics problems with no obvious connections to engineering. There is also a set of problems which seem to be too good to be true. For example, in Figure 1, we show a block diagram that Dorf and Bishop (2008, p. 481) claim is the attitude control system for the Boeing-Bell V-22 Osprey titltrotor aircraft.

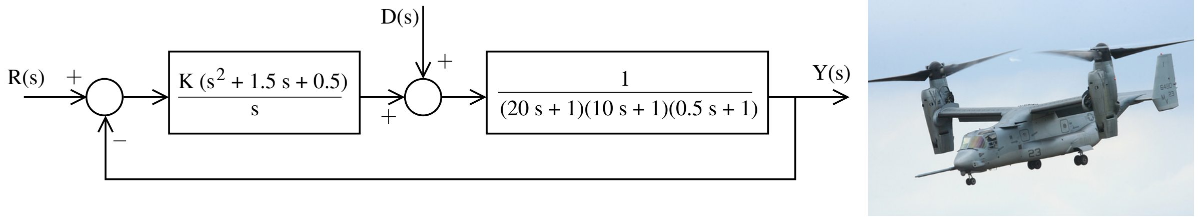

D(s)
R(s)
+
$\frac{K(s^2 + 1.5s + 0.5)}{s}$
+
+
−
$\frac{1}{(20s+1)(10s+1)(0.5s+1)}$
Y(s)

Figure 1: Block diagram for a better-than-average homework problem (Dorf and Bishop, 2008). Photograph from (Merlin_1, 2007).

At first glance this appears to be a fascinating problem. Students are to find feedback gains that stabilize the system and to find performance metrics for certain designs. However, when one types "V22 Osprey crash" into the search field of YouTube, one realizes that the dynamics of the part-airplane, part-helicopter vehicle are much more complicated than those of the third order open-loop transfer function depicted on the left of Figure 1. The story about the block diagram representing Osprey dynamics is really just a ruse. The homework problem is an over-simplified math problem that has little if any direct connection to engineering. From a constructivist perspective, the Osprey problem has additional drawbacks: most students do not have any direct experience with a tiltrotor aircraft. They do not have a gut feel for what would be a good amount of overshoot, or an appropriate settling time. Therefore, they cannot intrinsically place a value on the quality of their controller design. Value only comes from the marks they receive toward their overall grade in the class.

Previous Attempts at Creating an Active and Constructive DS&C Course

In the past, we have attempted to incorporate inquiry-based experiential learning into the dynamic systems & control course by focusing student activities and assignments on several simple canonical dynamics and control problems: mass-spring-damper systems, pendula, inverted pendula, DC electric motors, kinematic models of vehicle steering, simplified models of vehicle longitudinal dynamics, and more. In all cases, students used or created their own Matlab/Simulink simulations, sometimes with animation. The simulations had much in common with the "Virtual Experiments" modules in the upcoming textbook by Golnaraghi and Kuo (2010). For the electric motor and inverted pendulum problems, we provided physical hardware for students to experiment with.

Students created mathematical models for the systems, tested the utility and limitations of the mathematical models, and designed model-based controllers for the systems. Many of the assignments asked students to explore the space of physical parameters and controller gains. Some were open-ended design problems.

In course evaluations, students almost uniformly praised the concrete learning experiences. They claimed that the modules helped clarify theoretical content of the course and aided their learning. However, it did not appear as though students were connecting with the subject in a deep way. Although the inverted pendulum problem has much in common dynamically with a Segway Personal Transporter, the pendulum just is not as interesting. Although learning to control an electric motor provides a foundation on which one can design a robot arm for a Mars rover, studying the physics of the motor itself is not as exciting as the things one can potentially do with it. Our students chose mechanical engineering because they like to tinker. They want to build machines that do cool things. By focusing on simple systems that could be components of more interesting machines, or mechanical metaphors of more interesting machines, we wondered if we were missing the target in our attempt to create an effective and engaging learning environment.

At the time we began contemplating how to improve the dynamic systems & control course, we were experimenting with a new approach to teaching a course in numerical methods for mechanical engineers: using a video game. Students used techniques covered in the numerical methods course to solve computational problems within the game. As our studies of learning and engagement in the game-based course (Coller and Scott, 2009; Coller and Shernoff, 2009) showed definite promise, we decided to try using the game in dynamic systems & control. This paper describes that effort and outlines some of our preliminary results.

EduTorcs, the Video Game

Our video game is called *EduTorcs*. At its heart, our game is a sophisticated vehicle simulator. It has a computational model for automobile physics. A-arm suspension kinematics, steering rack/pinion/tie-rod kinematics, full 3D rotations, transmission, differential, engine characteristics, sway bars, and tire mechanics are all included in the model. Recently we have added a bicycle/motorcycle model to the game. The computational model of the bike include the physics of telescopic fork and swing arm suspension, full 3D rotations, tire mechanics, rider lean, and gyroscopic effects of the spinning wheels.

We have built our video game on top of an existing open-source game called *Torcs* (www.torcs.org). *Torcs* provides the game framework and graphics engine for our game. It synchronizes our simulations so that they run in real time, and it gives *EduTorcs* the look and sound of commercial video games similar to *Need for Speed* or *Gran Turismo*. See Figure 2 for screen shots of the game.

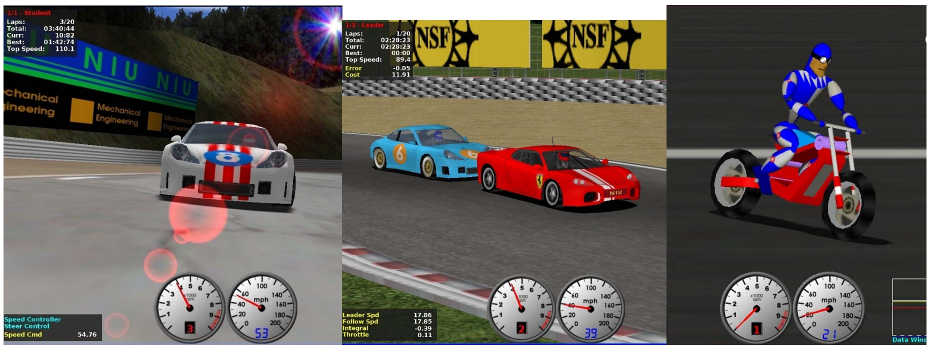


Figure 2: Screen shots from the video game *EduTorcs*.

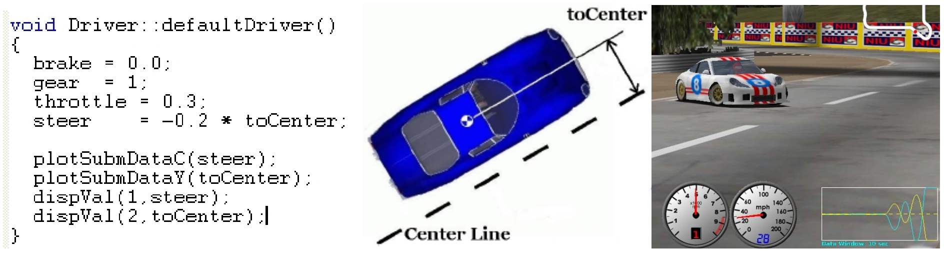
Even with all its similarities, students normally do not "play" our game *EduTorcs* like a traditional video game. They primarily interact with the game through a software interface we have created. Instead of spending countless hours, joystick in hand, honing one's eye-hand coordination and reaction skills, our mechanical engineering students improve their "driving" skills by applying tools and techniques of dynamic systems & control, and by applying sound engineering decision-making to the problem. The game's student interface provides access to certain data directly from the simulation. Students write driving algorithms in C++, and their programs get linked to the game at run time.

Our reason for choosing a car driving theme for the game, rather than rockets or airplanes, is because (almost) all our students know how to drive (in real life). Following the constructivist paradigm, we ask students to build upon this foundational knowledge in an effort to devise computational algorithms so that the car can drive itself around the track.

First Steps in the Game

Good video games are designed so that the initial challenges within the game are relatively easy to accomplish. Then, as the player's skills develop, the challenges intensify. Likewise, in *EduTorcs*, we start with a simple task: write a small algorithm that will steer the car around a serpentine track at modest speeds.

When students first run EduTorcs, their car sits motionless on the track. To get the car to move, one may write a short program similar to the one shown on the left of Figure 3.



```
void Driver::defaultDriver()
{
  brake = 0.0;
  gear  = 1;
  throttle = 0.3;
  steer    = -0.2 * toCenter;

  plotSubmDataC(steer);
  plotSubmDataY(toCenter);
  dispVal(1,steer);
  dispVal(2,toCenter);
}
```

Figure 3: First steps: getting the car to drive around the track.

The first line of the program, `brake=0.0`, tells the simulation to disengage the brakes. The second line, `gear=1`, puts the transmission in first gear. The third line, `throttle=0.3`, is equivalent to pressing on the gas pedal, 30% of full throttle. If the program contains just these three lines, then the car will ease forward, slowly picking up speed until the first turn in the road. Then, the car drives off the track and smashes into the wall. Clearly, the driving algorithm needs a steering command.

To get the car to steer, we suggest that students start with a command similar to the fourth line of code in Figure 3: `steer= -0.2*toCenter`. The variable `toCenter` is defined by the student interface. It contains the distance [in meters] of the car's lateral sensor from the center line of the track. The signed variable is positive when the car is to the left of the center line and negative when the car is to the right. Therefore, when the car is on the center line the `steer` command is set to zero and the car drives straight ahead. When the car is to the left of center, the `steer` command becomes negative, meaning that the car turns to the right. When the car is to the right of center, the `steer` command becomes positive causing the car to turn to the left. The farther the car is from the center line, the larger the steering command.

The driving function gets called every 0.02 sec. Therefore, driving commands are updated 50 times per second, and students get to see the mechanism of feedback in action. The steering strategy encoded in Figure 3 is to continually steer the car toward the center line of the track. It sounds like a good strategy that will work in straight sections and in turns.

When we compile the code of Figure 3 and run it within *EduTorcs*, we see that the car is able to complete the first turn in the practice track. Shortly afterward, however, the car begins zig-zagging. The rightmost picture in Figure 3 shows the car as it is experiencing the growing lateral oscillations, shortly before it crashes into the side wall.

This is where we hand the problem over to the students. We ask them to fix the controller, to make it steer smoothly around the track as if a sober human was driving the car.

In doing so, we provide them ample guidance. To begin, we ask students to run a part of the game which allows players to plug in a joystick and drive the car like in a traditional video game. There is an important difference, though. *EduTorcs* will record data from the joystick input. Afterward, we can examine the data and observe how the feedback controllers locked inside our unconscious minds are able to execute aggressive maneuvers and then damp out the lateral oscillations.

Students *discover* the distinguishing feature of the controllers inside their minds which permits them to damp out the oscillations. Their personal controllers advance the phase of the joystick input, compared to the controller of Figure 3. What does this mean? The phase advance is the result of our minds anticipating. We begin executing the turn before the car crosses the center line. To make the software-based controller work, students must incorporate that same type of anticipation. All of them figure it out, some with a little help.

It has been our experience that engineering students like to build things. They like to tinker. They like to figure out how to make things work. With the video game, all the tinkering takes place in the virtual world. Nonetheless, we suspect that tinkering virtual objects exercises the same cognitive muscles. At the same time, students are absorbing important concepts of automatic control. First, they are witnessing the important role of feedback. Secondly, they discover the powerful role of anticipation (also known as lead compensation or derivative action) in creating stability. The lesson was learned organically without any theorems or integral transformations.

In a typical textbook, the latter principle is presented by considering a system that can be represented by a second order differential equation. First, one takes the Laplace transform to derive the open-loop transfer function. One then inserts the transfer function into a block diagram with a feed back loop. After inserting a proportional + derivative controller into the compensator block, one derives the closed-loop transfer function. Because the original system was second order, one can obtain explicit formula for the closed-loop poles. By examining how the derivative gain affects the real parts of the poles, one can detect the stabilizing effect of derivative action. To have a true understanding of the principle one must be able to follow and comprehend all the steps in the derivation.

In the game-based dynamic systems & control course, we go through the same derivation, but we do it a few weeks *after* the initial *EduTorcs* exercise. By the time they see the long series of mathematical manipulations, students have a chance to develop an intuitive understanding how derivative action works. They appreciate the technique, and they might have a deeper motivation to see the theory behind it. The game provides a mechanism for us to turn the traditional deductive style of dynamic systems & control education on its head.

Rest of the Game

As the game progresses, challenges within the game become more difficult. Students eventually develop controllers for driving the bike, including stabilization of "wheelies" and riding the bike in the reverse direction so that the steered wheel is in the back. Page limits prevent us from describing the rest of the game in much detail, so we will simply state that, from an instructional perspective, the way we use the game in the rest of the course is similar to that described in the previous section. That is, we use the game as an authentic way to introduce students to key concepts *before* we bombard them with mathematics. Concepts include steady state error, integral action, lag compensation, root locus design, Bode-Nyquist design, non-minimum phase systems, and more.

Assessment of Learning

Recognizing that we were about to make a dramatic shift in how we teach dynamic systems & control, we began conducting an experiment in the spring of 2007, the last semester we taught the course without the game. We developed two tests to assess students' conceptual understanding of course material. The first test was administered roughly a week before the midterm examination. Students took the other test about a week before the final examination. We told students that these were practice tests. Their performance on the tests would not affect their grade in the course, and that they could use the exams to identify their weaknesses to help them study for the upcoming exams that did count. By designing our assessment this way, we believe we were more likely to capture students' understanding of the material, while filtering out the effects of last-minute studying for an exam. Also, since students are unlikely to study for a test that has no impact on their grade, we were comfortable giving the exact same practice test each year of our study.

On the two tests, there were 69 multiple choice questions covering 21 concepts in the dynamic systems & control course. In Figure 4, we show differences between class averages for each of the 21 concepts. When the difference is positive, students in the game-based 2009 course scored better (on average) than students taking the non-game course in 2007. In the figure, differences are normalized by the pooled standard deviations for each topic. The error bars denote 95% confidence intervals for the differences between means. Sample sizes for the game and non-game classes are 46 and 50 respectively for the first 11 concepts and 45 and 49 for the remaining concepts.

Students in the game based course and non-game course scored almost identically on the Mechanics Baseline Test (Hestenes and Wells, 1992) at the beginning of the semester. Both took the dynamic systems and control course from the same instructor, using the same textbook. Yet the differences in concept tests scores reported in Figure 4 are clearly lopsided. Students taking the game-based course score better on 18 out of 21 of the concepts. Fourteen of these are statistically significant at a level $p<0.05$ (two-tailed). There is only one concept in which the non-game students scored significantly better.

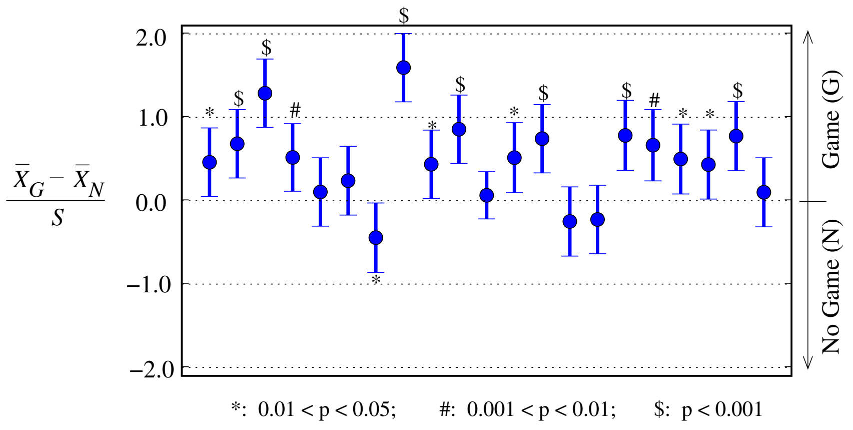


Figure 4: Normalized differences game and non-game average scores on 21 concepts.

Accompanying these learning gains, we have found a dramatic increase in the number of students who choose to take the advanced controls course as an elective. Also the number of students who choose a senior capstone design project with a major control component has increased significantly. The increase is due to students who have taken the dynamic systems and control course. On average, it appears that students are connecting with the subject more. In the upcoming months, we plan to comb through our data to see whether students with different learning styles, different motivation orientations, and different video game playing habits respond differently to our educational game.

Acknowledgement

The author gratefully acknowledges support from the National Science Foundation of the United States under Grant number 0633162. Any opinions, findings, and conclusions are those of the author and do not necessarily reflect those of the National Science Foundation.

References

Coller, B. D. & Scott, M. J. (2009). Effectiveness of using a video game to teach a course in mechanical engineering, *Computers & Education*, 53(3): 900 – 912.

Coller, B. D. & Shernoff, D. J. (2009). Video game-based education in mechanical engineering: A look at student engagement, *International Journal of Engineering Education*, 25(2): 308 – 318.

Dorf, R. C. & Bishop, R. H. (2008). *Modern Control Systems*, Upper Saddle River, New Jersey: Pearson Education, Inc.

Franklin, G. F., Powell, J. D., and Emami-Naeini, A. (2006). *Feedback Control of Dynamic Systems*, 5th Edition, Upper Saddle River, New Jersey: Prentice Hall.

Golnaraghi, F. & Kuo, B. C. (2010). *Automatic Control Systems*, 9th Edition, John Wiley & Sons.

Hestenes, D. & Wells, M. (1992). A mechanics baseline test, *The Physics Teacher*, 30: 159 – 165.

Merlin_1 (2007). *Osprey*. Photograph obtained from http://www.flickr.com/photos/merlin_1/365734300/ on 4 Aug. 2009. Photograph covered by Creative Commons License that allows use for non-profit purposes and does not allow derivative works. Use of the photograph does not constitute endorsement by the photographer.

Ogata, K. (2009). *Modern Control Engineering*, 5th Edition, Upper Saddle River, New Jersey: Prentice Hall.

Prince, M. J. & Felder, R. M. (2006). Inductive teaching and learning methods: Definitions, comparisons, and research bases, *Journal of Engineering Eduction*, 95(2), 123 – 138.