LESSONS LEARNED FROM TEACHING DYNAMIC SYSTEMS & CONTROL WITH A VIDEO GAME

B. D. Coller
Department of Mechanical Engineering
Northern Illinois University

Abstract

Playing digital games on personal computers and game consoles is a massively popular form of mediated entertainment, which is particularly effective at commanding the fascination and attention of adolescents, young adults, and some not-so-young adults. There is a growing number of education scholars who argue that video games (or at least the aspects that make them so engaging) should migrate into the classroom as well.

Since 2005, we have used a customized race car simulation game to teach a computational methods course to mechanical engineering undergraduates. The introduction of the game has been a demonstrable success. However, when we adapted the game-based instructional model to a different course, a dynamic systems and control course, the improvements, so far, have been less dramatic. In this paper, we re-think how a video game can be used to teach Dynamic Systems & Control.

Introduction

In the Spring of 2008, we began teaching a core mechanical engineering course, Dynamic Systems & Control (DS&C), with a video game. At its heart, the video game is a sophisticated vehicle simulation that runs in real-time. On the surface, though, it has much of the look and feel of a commercial video game. A screen shot of the game, *EduTorcs,* is shown in Figure 1.

Students do not "play" the video game in the usual way. They interact with the game through a software interface. Instead of spending countless hours, joystick in hand, honing one's eye-hand coordination and reaction skills, the mechanical engineering students improve their "driving" skills by applying engineering analysis to the problem. They write driving algorithms in C++, and their programs get linked to the game at run time. Although they drive a virtual car in a virtual world, students solve authentic engineering problems. To succeed in the game students must think and act like engineers.

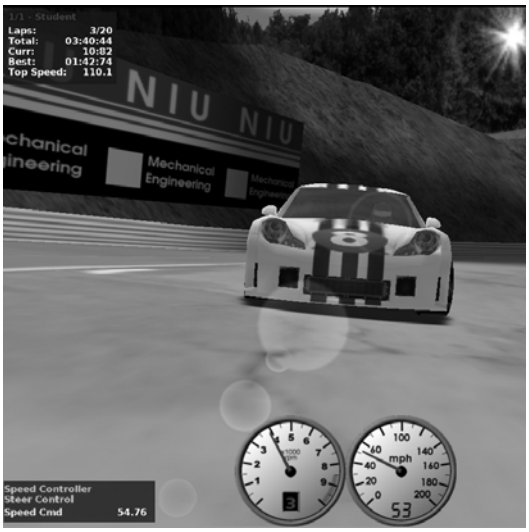


Figure 1: Screen shot of our game, EduTorcs.

We originally developed the game for use in a computational methods course. In that course students devise algorithms for driving the car as fast as possible around the track. In addition to steering, the driving programs needed to calculate optimal times to shift gears; to determine the maximum speed it could navigate corners; to compute the best time to begin braking before entering a turn; and much more. When the game was first introduced in the computational methods course in 2005, we saw immediate and dramatic improvements in learning outcomes.

While teaching the game-based computational methods course, we recognized the Dynamic Systems & Control (DS&C) course as another potential venue for the educational video game. In DS&C, students could more deeply explore the input/output behavior of the car, and they could develop more sophisticated steering algorithms, path-following algorithms, and speed control algorithms. DS&C is a notoriously difficult class to teach. The mathematics are abstract; it is difficult and unnatural for mechanical engineering undergraduates to think in the Laplace domain. By embedding the theory into an engaging context that builds off their prior knowledge, we hoped, and still hope, to make it more clear.

When we implemented game for the first time in DS&C, however, we did not experience the dramatic gains in student performance that we saw in the computational methods course. Part of this may be explained by an unusual traumatic event that profoundly affected everyone on our campus a year ago. However, we suspect that part may lie in fundamental differences between the two courses. We suspect that the particular game-based instructional model that worked so well in the computational methods course might not be appropriate for the game-based DS&C course. In the current paper, we explore this possibility more deeply and describe changes we are making in Spring of 2009, when the course is offered the second time.

What Makes the Game-Based Computational Methods Class Work

As described in the previous section, our success in using the *EduTorcs* video game in our computational methods course served as motivation for attempting to use the game in the DS&C course. The computational methods course also served as a template for how we would design our game-based learning activities in DS&C. In this section we outline a typical assignment in the computational methods course so that we could later illustrate why the approach might not be appropriate for DS&C. More detailed examples from the numerical methods course may be found in Coller[1].

A representative game-based learning activity.

A few weeks into the computational methods course, we give students a challenge that resembles a drag race. Students' cars are placed at the beginning of a long straight section of track. Their goal is to cross the finish line, 700 meters away, within a very short amount of time. The winning strategy sounds simple: just steer straight ahead and go as fast as possible. However, to complete the race in the allotted time, one's driving program must shift gears at precisely the right moments. Otherwise valuable fractions of seconds will be squandered as the car races in a suboptimal gear.

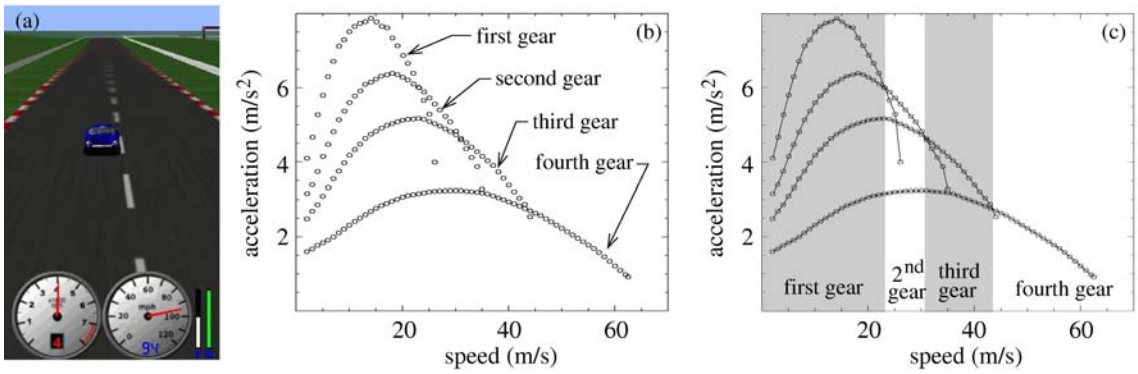


Figure 2: Strategy for computing optimal shift points in the game.

After considerable thought, students recognize this as a root finding problem. If they could calculate where acceleration versus speed curves for each gear intersect (see Figure 2), then they would know precisely when to make the shift. Students had to write generic computer programs that would calculate the shift points (find roots) for a variety of automobile types, engine types, and transmission types. Incidentally, they also had to calculate acceleration from discretely sampled speed data and then fit the data with continuous curves in order (two other topics of the course) for the root finding to work.

A Traditional Root Finding Problem

To contrast the game-based exercise with the type of activity that students typically encounter in a traditional lecture/textbook based course, consider the following root finding problem from Rao[2].

(Problem 2.6[2]) The normal stress induced at the inner fiber of a torsional helical spring is given by

$$\sigma_i = \left\{ \frac{4C^2 - C - 1}{4C(C-1)} \right\} \frac{Mc}{I},$$

where $I = \pi d^4/64$, $c = d/2$, $C = D/d$, M is the bending moment, D is the mean coil diameter, and d is the wire diameter. Find the value of C that corresponds to a stress of σ_i = 55,000 psi when M = 5 lb-in. and D = 0.1 in.

When publishing companies advertise their numerical methods textbooks, including the book by Rao[2], they frequently highlight the engineering applications covered in the books. The problem described above is typical of the engineering applications that one finds in numerical methods text books1. For mechanical engineering students, the problem makes a connection to a prior mechanics of materials course

A Comparison of the Types of Problems

On the surface, it is clear that both tasks described above are root finding problems embedded in an engineering context. But that is where the similarities end.

Suppose we ask the question, "Why would a student care about this problem?" The normal stress induced at the inner fiber of a torsional helical spring is not something that naturally inspires the imagination of most 20-year-olds, not even the engineers-to-be. Finding the "correct" answer is not likely to tell them anything that they naturally want to know. Since the correct answer only serves to give them credit toward their grade, students are likely to ignore the engineering context and treat it as a generic math problem.

By design, the game-based experience is different. Students begin writing driving algorithms in the first week of the semester. After getting the rudimentary driving code to work, they begin probing the limits of their capabilities. The question of when is the best moment to shift ges is one that naturally arises, before we get to the subject of root finding.

With luck, students working on the torsional spring problem will learn a numerical root finding technique. But, which technique? And what will they learn about it? Chapter 2 of Rao[2] presents eight root finding techniques that can be used in a variety of circumstances. It turns out that any of the eight can be used to solve the torsional spring problem. The problem does not stipulate which technique to use. There is no value to choosing a technique that converges quickly, compared to one with slow convergence. The problem only needs to be solved once so there is little benefit to choosing a technique whose iterative process starts easily. In fact, there is no need to use any of the numerical methods covered by the textbook. Students may use a plotting package to solve it graphically. They may perform a manual search by punching numbers into a pocket calculator.

They may find a canned routine that generates the root(s) without requiring any thought at all.

In the gear shifting problem, choice of root finding technique is critically important. For example, any technique that requires a derivative is doomed to fail: differentiation of discretely sampled data is inherently noisy. Furthermore, students need a technique that has robust convergence properties. Their shift point calculation methods are supposed to work with any car and any transmission, information students do not know a priori. In summary, students have to make value judgments that arise naturally out of the problem. This is what happens in engineering practice. Students working on the game-based problem learn to think, act, and value as engineers do. They take on identities of engineers rather than mere engineering students. As such, the game is used to create a strikingly different type of learning environment compared to that of the textbook.

Effects of Game-Based Learning in Computational Methods

We found that, in the computational methods course, our game-based approach had a dramatic impact on student learning and attitude. We summarize the results below, providing references to articles in which the results are published.

- ***Finding #1: Students taking the game-based numerical methods course spend more time on homework than students in other mechanical engineering courses.*** One of our first findings came from a series of surveys of students in all undergraduate mechanical engineering courses at Northern Illinois University (NIU). As part of ABET assessment, students reported how much time, outside of lecture and lab, they spend on course work. The results were striking. Students taking the game-based numerical methods course, on average, spent more time doing homework than students in any other undergraduate mechanical engineering course, including the capstone senior design course, other core courses, and technical electives. The differences in hours reported were statistically significant and dramatic. The average time invested in numerical methods was 35% higher than the next highest course. It was twice the average of all other courses. Details are published in Coller & Scott[3].

- ***Finding #2: Students taking the game-based numerical methods cours appear to learn the material more deeply than students taking traditional textbook-based numerical methods courses.*** Using a concept mapping technique, we[3] compared the "computational methods knowledge" of students taking the game-based course to other students taking a more traditional computational methods course. Participants taking a "traditional" course were obtained from six different classes, taught by four instructors at two universities.

  Our rubric detailed four different levels of learning. The first two levels simply quantified how many different concepts and topics they could readily recall from the course. One these first two levels the game-based students and traditional students scored similarly; there was no statistically significant difference.

  The other two measures quantified deeper levels of learning expressed in the concept maps. The third measure counted how many "defining features per main topic" that students could list for their concepts. The fourth counted how may connections students made between topics. On these latter two measures, game-based students outperformed their counterparts by nearly 3-to-1, and by more than 10-to-1, respectively.

- ***Finding #3: Students appear to be more engaged when working on video game-based homework than other engineering homework.*** We measured student engagement with a technique called the Experience Sampling Method (ESM). For

three one-week periods during the semester, students wore wristwatches that were pre-programmed to beep at randomly chosen times. When the watches beeped, students filled out a brief survey, indicating what they were doing and their mood and perceptions toward the activity in which they were involved. The ESM allowed us to obtain a snapshot of student experiences 'in the moment,' and therefore did not rely on memory to reconstruct engagement from past experiences.

We conducted a series of Hierarchical Linear Models to compare within-person differences of the composite engagement variables while working on homework with the video game compared to homework for other classes (without a game). We found that *Intellectual Intensity*, *Intrinsic Motivation*, and *Engagement* was significantly higher when working on the video game. Results are presented in Coller & Shernoff[4].

- ***Finding #4: Long after the course is over, students who have taken the game-based numerical methods course see more value in the course than students taking a traditional numerical methods course.*** Just before graduating from their program, students were given a list of all required courses in the mechanical engineering curriculum. We asked students to rank the courses on a scale from 1 (low) to 10 (high) indicating "how important [they] feel each course was in [their] education toward becoming a mechanical engineer." Students were advised to evaluate the importance of the course subject and material rather than the quality of instruction.

  Students who took a traditional numerical methods class, on average, gave it the lowest ranking of all mechanical engineering courses. Students who took the game-based numerical methods course ranked that course almost exactly in the middle of all courses. The difference was statistically significant. This result is published in Coller & Scott[3].

A First Attempt at Game-Based Dynamic Systems & Control

When we originally created our game-based DS&C course, we attempted to adopt the instructional model provided by the successful game-based numerical methods course. That is, we sought to embed authentic engineering challenges into the game that would require students to deeply explore the dynamics of the vehicles and devise creative control strategies.

An Example Challenge

One of the challenges we gave to the students is something called the "Thread the Needle" event. The game would control the speed of a sports car as it raced toward four other cars parked on the track. Just before crashing into one of the parked cars, the student's driving program would receive a signal to execute and evasive steering maneuver that would avoid the first car and barely squeeze between two other parked cars slightly farther down the road. The evasive maneuver would need to work over a range of randomly generated speeds.

Figure 3: Screen shot of "Thread the Needle" event.

The idea was that students would determine the controller performance specifications (rise time, maximum, overshoot, and settling time) from the speed of their car and the spacing between the parked cars. Then they would carefully design controllers to meet the specifications. If they do not design their controller correctly they would get immediate feedback in the form of a smoldering pile of wreckage. Of course, they would be able to start fresh by restarting the game. Students would persevere until they figured out the details.

Unintended Consequences

The problem with the task described above is that students can satisfy the requirements without going through the exercise of exploring the fundamentals of how controller parameters affect performance measures. Instead, students could simply choose parameters through an iterative process of trial and error. Since there are only two controller gains at this stage, they will eventually stumble upon a workable solution.

In the computational methods course, there were built-in mechanisms to prevent such ad-hoc solutions. The mechanisms were not practical in the DS&C course.

Part of the difficulty lies in the nature of feedback control systems. One of the characteristics that makes feedback so great (in the real world) is its robustness. It can be used to control systems whose dynamics are not precisely known. In the virtual world of our video game, this robustness property made it difficult to create challenges that would, by design, reject lousy approaches to the problem.

In some later assignments, we required students to solve the tasks using specific prescribed techniques. We did this reluctantly, because we realized that the new rules were imposed artificially. The requirements on how one solves the problems did not come naturally or authentically out of the challenge. It undercut one of the primary mechanisms which can transform a video game into a powerful learning tool.

Learning Measures & Observations

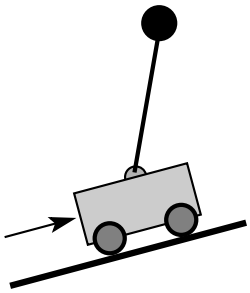
At the time of this writing, we have performed only a preliminary analysis comparing learning outcomes and engagement measures, comparing students who took the game-based DS&C course in Spring 2008 to those who took DS&C in 2007 without the game. There were a couple categories in which the game-based students appeared to outperform their non-game counterparts to a statistically significant degree. However the differences were not nearly as dramatic as was experienced in computational methods study[3]. In most measures, though, we did not observe any significant gains. There were no declines either.

Also, it is important to note that, due to highly unusual and traumatic circumstances, we have doubt as to whether any conclusions can be drawn from comparisons between students taking the game-based course last year and those taking the traditional course the previous year. In February of 2008, while we were in class, five of the students' classmates were murdered and another few dozen were injured when a gunman came onto campus and shot up a classroom. Classes resumed a little more than a week later, but the atmosphere on campus was profoundly impacted for the rest of the semester.

Nonetheless, in surveys, students almost unanimously reported very positive responses to the game. Almost all believed that the game significantly improved their understanding of the subject. Also, they were eager to provided feedback on how the user interface can be improved, and which new features should be added to the game.

A Bright Spot

One of the bright spots of the educational experience was the last series of exercises that students were asked to complete. Students were



Classical inverted pendulum
(on a slope)

Figure 4: The pendu-car.

given a car-driving, video game version of the classic inverted pendulum problem. We call it the Pendu-Car. Although it seems unnatural, we placed a pendulum atop a car, and we asked students to create steering algorithms that keep the pendulum balanced in its upright state. In addition to keeping the pendulum balanced, the car would need to steer itself around the track. Furthermore, parts of the track are banked (sloped sideways), adding one more element of difficulty to the challenge.

Students ranked this challenge as their best experience in the course. Based upon personal observation, I concur. Student could not get away with simply slapping on a PID controller somewhere. This challenge required students to think deeper and more creatively. They had to devise their own control architecture and test it conceptually and implement it. Students had to do a lot of tinkering, in a good constructive way. As will be described in a forthcoming article, learning measures related to this exercise were the ones in which we observed most improvement.

Reflections

"Learning is a cycle of probing the world (doing something); reflecting in and on this action and, on this basis, forming a hypothesis; reprobing the world to test this hypothesis; and then accepting or rethinking the hypothesis."

This is one of several fundamental learning principles, proffered by Gee[5], for which the medium of video games may be particularly well-equipped to address. It is one of the fundamental principles that guided both the computational methods and DS&C versions of *EduTorcs*. Rather than overtly provide information to students through textbook and lecture, we aspired to create a mediated environment in which students could experiment and make discoveries.

Nonetheless, Gee acknowledges that one must strike proper balance between overt information and immersion in actual contexts of practice. One cannot, he explains, give novices a set of ramps and balls, and then expect them to arrive at Galileo's principles of motion on their own. This experience of integrating a video game into a DS&C class has illuminated some of the difficulty in getting the balance right. In particular, an instructor must take care in choosing which principles should be discovered through the game.

The "Thread the Needle" event described earlier, in which students attempt to finesse the performance characteristics of a given control system, is not a good exploratory experience for

students. We now recognize that the learning exercise turned one of Gee's game-related learning principles on its head. Gee's "Amplification of Input" principle states that small improvements in players performance/ learning should be amplified by the game[5]. This provides students feedback that confirms good approaches to solving problems. Due to robustness properties of feedback control systems, "Thread the Needle" created "attenuation of input" instead. Both deep and shallow attempts to solve the problem could produce similar outcomes in the game.

Moving Forward

The game-based dynamic systems and control course is being offered again in Spring 2009. The challenges and exercises in the new course are more similar to the Pendu-Car project described previously. Instead of having students achieve specific performance metrics, we focus more on qualitative aspects. In devising a control strategy for the Pendu-Car, for example, one needs to think deeply about how the feedback architecture is constructed. How can one simultaneously control two strongly coupled dynamic states (pendulum angle and car position)? Our engineering students can figure this out on their own with minimal assistance from the instructor. Furthermore, students like doing it. It is the DS&C equivalent of tinkering in the machine shop… exploring how elemental parts connect together to form a whole.

We have also made significant changes to the user interface. In earlier versions of the game, designed strictly for the computational methods course, *EduTorcs* only communicated with student-written programs that generate driving commands.

Recently we have added a layer to the interface that will allow players to hook up a joystick to the game. Yes, this allows one to play *EduTorcs* like a traditional video game, but that is just aside effect. The game has

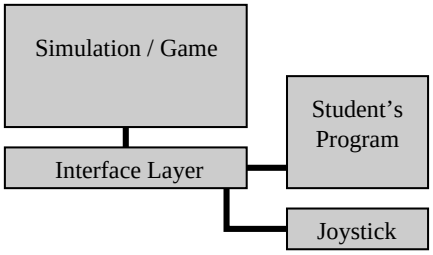


Figure 5: User interface for EduTorcs.

telemetry mechanisms that allow students to record their joystick steering commands, along with simulation data. When we examine the data generated during an aggressive lane change maneuver, for example, we see exactly how the subconscious controller inside our brains is able to damp out the car's lateral oscillations. Then, if we write a mathematical control law which produces the same effect, we see that the advance in phase can be produced by a derivative term. Thus, rather than introduce *derivative action* – one of the key concepts in the course – through transfer functions and block diagram algebra in the Laplace domain, we broach the subject by building on top of something that students already know: how to drive a car.

Along the same lines, we the game to study a problem that nearly everyone is intimately familiar with: riding a bicycle. Yet very few people are able to correctly explain bicycle physics. From a modeling, analysis and control perspective the motorcycle/bicycle is an incredibly rich problem. Within the virtual, but physically accurate, world of the video game, one can explore bicycle dynamics and control quite thoroughly. What happens if we enhance/remove/reverse the mechanical trail (caster effect) of the steered wheel? What happens if we enhance/remove/reverse the gyroscopic effects of the two spinning wheels? What if we reconfigure the bike so that the steered wheel is in the back? How do these changes affect stability and rideablity? How do we even quantify rideability?

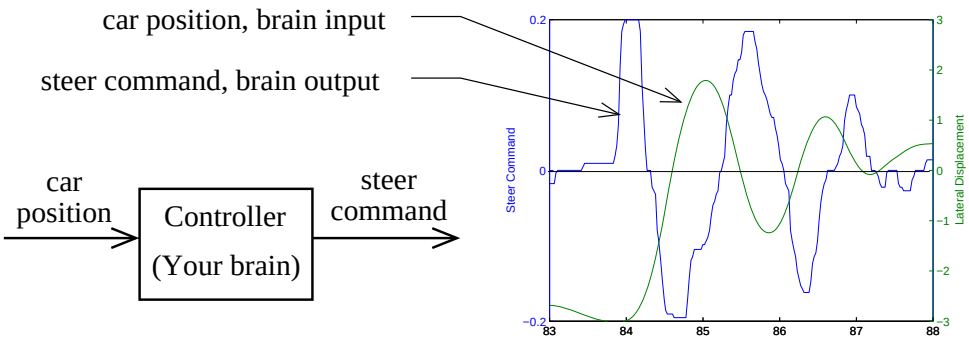


Figure 6: Students learn feedback control by exploring how their subconscious brains do it.

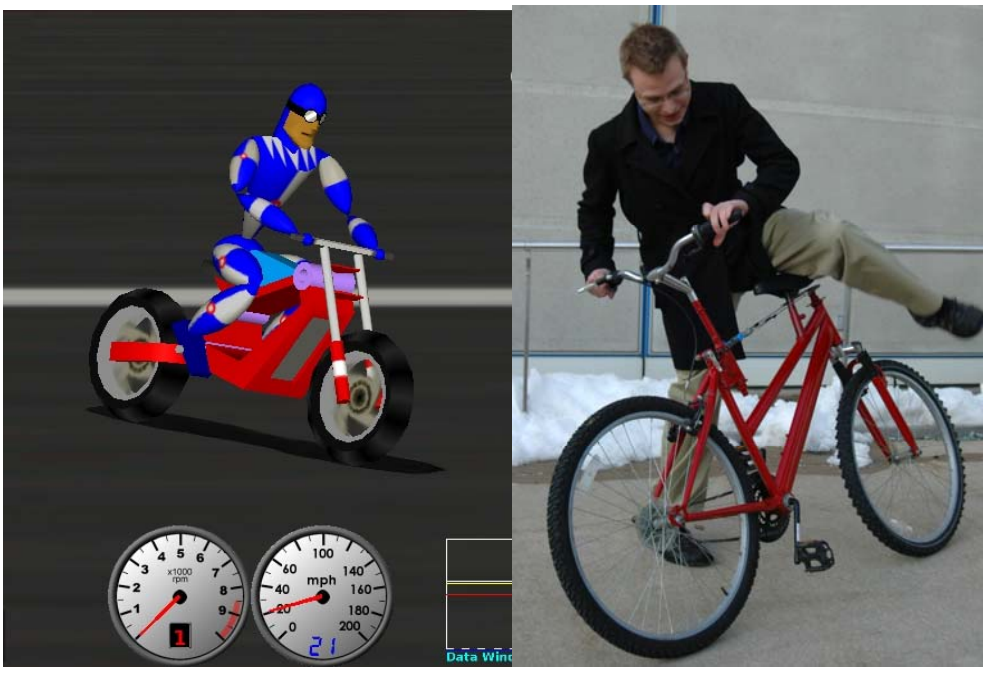


Figure 7: The EduTorcs bike, and the "unrideable" rear-steered bike.

The rear steered bikes like that shown on the right above are known to be “unridable” by humans[7,8,9]. The dynamics exhibited by the rear steered bike in the game seem so foreign that we had to build a real bike so that students can feel how strange it is. With the experience, students are able to return to the game, understand the difficulty from a physical and mathematical perspective, and develop drive-by-wire controllers so that novice game players can ride the virtual rear steered bike with a joystick. Although the bike is virtual, the engineering is real and authentic.

Postscript

At the time we are submitting this for final publication, the game-based Dynamic Systems & Control has been offered a second time. In the second attempt, we did observe significant improvements in student learning outcomes. Details will appear in a forthcoming publication.

Bibliography

1. B. D. Coller, "Implementing a video game to teach principles of mechanical engineering," *Proceedings of the 2007 American Society for Engineering Education Annual Conference*. 2007.

2. S. S. Rao, *Applied Numerical Methods for Engineers and Scientists*, Prentice Hall, 2002.

3. B. D. Coller & M. J. Scott, "Effectiveness of using a video game to teach a course in mechanical engineering," (pre-print) 2008, available at www.ceet.niu.edu/faculty/coller

4. B. D. Coller and D. J. Shernoff, "Video game-based education in mechanical engineering: A look at student engagement," International Journal of Engineering Education, (in press)

5. J.P. Gee, What Video Games Have to Teach Us About Learning and Literacy, Palgrave Macmillan, 2003.

6. K. Åström, R. E. Klein, and A. Lennartsson, "Bicycle dynamics and control: Adapted bicycles for education and research," *IEEE Control Systems Magazine*, 15, pp 26 – 47, 2005.

7. K. Åström, "Limitations on control system performance, "European Journal of Control, 6, pp 2 – 20, 2000.

8. R. Schwarz, "Accident avoidance characteristics of unconventional motorcycle configurations," SAE Paper 790258, 1979.

9. S. Suryanararayanan, M. Tomizuka, and M. Weaver, "System dynamics and control of bicycles at high speeds," *Proceedings of the American Control Conference*, pp 845 – 850, 2002.

Biographical Information

Brianno Coller received his Ph.D. in Theoretical and Applied Mechanics from Cornell University. He was a Postdoctoral Research Fellow at the California Institute of Technology. He is currently an Associate Professor of Mechanical Engineering at Northern Illinois University. Coller's primary field of expertise is in nonlinear dynamic systems and control. For the past several years he has engaged in research in engineering education: mostly in the use of video games to teach core engineering principles.