2010 American Control Conference
Marriott Waterfront, Baltimore, MD, USA
June 30-July 02, 2010

A Video Game for Teaching Dynamic Systems & Control to Mechanical Engineering Undergraduates

B. D. Coller

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.**

I. Introduction

Like many undergraduate mechanical engineering curricula, Northern Illinois University has a junior-level, course on dynamic systems & control (DS&C). Broadly speaking, students who successfully complete the course should demonstrate the following outcomes:

1) Derive differential equations that model a broad class of mechanical systems.
2) Determine stability of these systems and their temporal characteristics directly from the mathematical models.
3) Construct feedback loops with sensors, actuators and computing elements which stabilize the system or alter its dynamics in favorable ways.

In addition to these "hard," content-based outcomes, we may also have "soft" outcomes such as developing a genuine interest in the subject. We would like the first dynamic systems and control course to serve as a gateway to deeper study of the subject. We would like students to become comfortable with dynamic systems & control so that the framework presented in the course becomes a natural and intuitive way of thinking. When confronted with engineering problems outside their DS&C course, we would like students to competently apply a DS&C perspective.

In our experience, students are able to achieve many of the "hard" outcomes enumerated above by cramming mathematical recipes into their short term memories and then performing satisfactorily on the exams. However, it was rare for students to at our institution take a second DS&C course as an elective. It was even more rare to find a student choosing a capstone design project that incorporates feedback control.

To improve the dynamic systems and control experience, we experimented with a video game. Our experiment is modeled after an earlier project in which we introduced a video game into a required computational methods course. In that prior experience, we found that students taking the game-based computational methods course were more engaged [1] and they learned the material more deeply [2], compared to students taking a traditional textbook-based course.

In the remainder of the paper, we outline the foundations of congnitive science upon which we build our educational approach. We discuss how the game-based dynamic systems and control course is structured. Finally, we present evidence demonstrating the success of the effort.

II. Dissonance Between How We Teach and How People Learn

A. A Difference in Perception

Let's face it. Those of us with expertise in dynamic systems and control are highly mathematical, perhaps more so than the typical engineering professor. When we take a step back and look at the content of a first dynamic systems and control course, we see a perfectly logical collection of mathematical concepts, tools, algorithms, and theorems. 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

Manuscript received September 22, 2009. This work was supported by the National Science Foundation under Grant 0633162. Any opinions, findings, and conclusions are those of the author and do not necessarily reflect those of the National Science Foundation.

B. D. Coller is Associate Professor of Mechanical Engineering at Northern Illinois University, DeKalb, IL 60115 USA (phone: 815-753-9944 fax: 815-753-0416; e-mail: coller@ceet.niu.edu)

often suffices to achieve a passing grade.

B. Constructivist Theories of Learning

One of the most common teaching models one finds in engineering classrooms is that of direct instruction: a mostly one-way process in which the all-knowing instructor dispenses knowledge to the novice students who then are to absorb and internalize it. From this perspective, students are merely empty vessels that can be filled (slowly) with the professor's knowledge.

Cognitive science, however, paints a different picture of how learning actually works. 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 integrate 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. [3]

The early chapters of many elementary textbooks often discuss feedback in general and refer to common every-day devices such as thermostats and automotive cruise control. But this is normally a passive reading exercise rather than an engaging experience for students. After the introductory chapters, common textbooks become very axiomatic and deductive. The building blocks upon which new mathematical knowledge is constructed is prior mathematical knowledge.

In the minds of us DS&C experts, self-described mathematical geeks, this may appear to be a natural and logical choice. However, mechanical engineering undergraduates are different. Most did not choose mechanical engineering because they liked mathematics. They chose mechanical engineering because they like cars, airplanes, bicycles… They like to build things. They like to take things apart. They like to tinker and figure out how things work. The mental structures (schemata) our students possess are less compatible with the expository style of typical textbooks.

C. Connections to Real Machines and Devices

Authors and instructors often attempt to establish connections between the theory and the types of machines and devices students care about. In general, this is good. But it is important to examine it from the perspective of the constructivist framework of human learning.

For example, consider the homework problem shown in Figure 1. In this problem copied from Dorf and Bishop [4, p. 481], students are led to believe that the block diagram is that of the attitude control system of the awe-inspiring Boeing-Bell V-22 Osprey tiltrotor aircraft. In the homework problem, students are asked to find the range of gains that stabilize the aircraft and to calculate performance metrics for certain gain combinations.

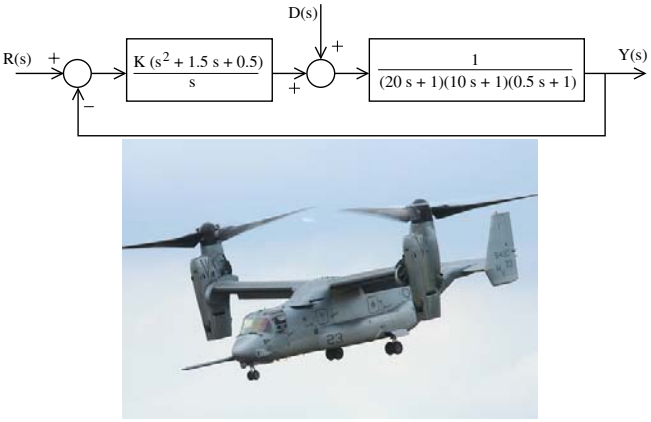

Fig. 1. Block diagram for a better-than-average homework problem [4]. Photograph from [5].

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, and the block diagram in Figure 1 with third order plant dynamics and PID controller is a vast over-simplification. Any credible connection between the mathematics and the real engineering system is severed. 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 score they receive toward their overall grade in the class.

III. 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 & Kuo [6]. 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.

Based on our earlier success in using a video game to teach computational methods [1], [2], we decided to try a game-based approach in Dynamic Systems & Control.

IV. 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.

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.

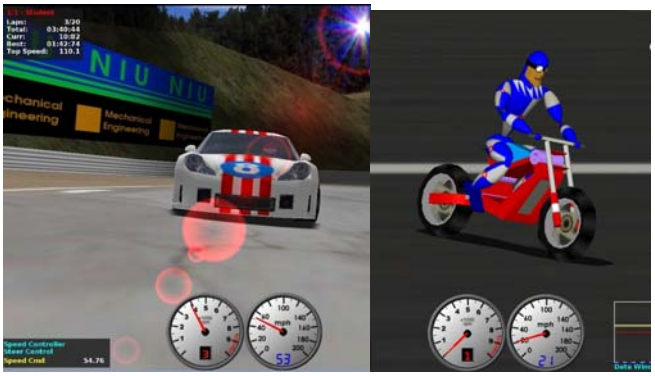

Fig. 2. Screen shots from the game EduTorcs.

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 effort to devise computational algorithms so that the car can drive itself around the track.

A. 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 below:

```
Void Driver::defaultDriver()
{
   brake = 0.0;
   gear = 1;
   throttle = 0.3;
}
```

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 modify their code as follows:

```
Void Driver::defaultDriver()
{
   brake = 0.0;
   gear = 1;
   throttle = 0.3;
   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. (See Figure 3.) 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.

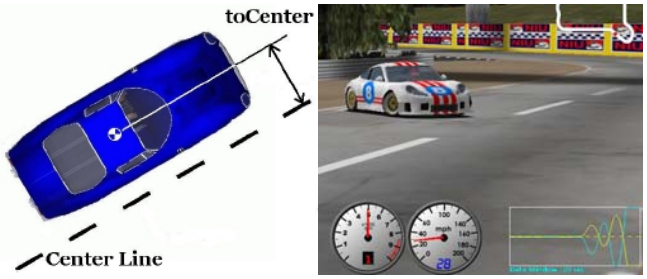

Fig. 3. Definition of `toCenter`, and our initial attempt at creating a steering controller.

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 initial steering strategy we encode into EduTorcs is one which continually steers the car toward the center line of the track. It makes intuitive sense. It seems like the strategy we employ when we drive or own (real) cars. (Although we usually drive in the center of our lane rather than the center of the road.)

Surprisingly, it does NOT work! When we compile the code 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 picture on the right 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. On the bottom right corner of Figure 3, EduTorcs provides a real-time plot of toCenter and steer as functions of time. Notice that the steer command and lateral displacement are exactly 180^o out of phase. The personal steer controllers that students have locked inside their heads advance the phase of the joystick input (steer input), compared to the proportional controller in our code snippet. 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.

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 with 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 derivative action or lead compensation) in creating stability. The lesson was learned organically without any theorems or integral transformations.

In our game-based dynamic systems & control course, we rigorously and mathematically examine the effects of derivative action/lead compensation from the perspectives of PID, root locus, and Bode-Nyquist in much the same way as a traditional course. However, we do it only *after* students have had the opportunity to develop an intuition about what it means, *after* they have developed the mental structures necessary to accommodate the concept into their understanding.

In the game-based course, the theoretical derivation not only serves to confirm their intuitive understanding. Students can make new use of it. The theory shows students how to scale their steering gains with speed, so they can safely steer their Porsche at 160 mph (257 km/hr). Also, with the theory and modeling skills, they can figure out the best location to place their lateral position sensor (which generates the `toCenter` measurement) on the car.

B. Rest of the Game

Throughout the course, we use the video game in a similar way to introduce other concepts. For example, we use an event within EduTorcs called "Follow the Leader" to introduce several concepts. The task is to design a throttle/brake controller that will keep the car's front bumper exactly 1 meter away from the rear bumper of the car in front of it. (See Figure 4). During one lap around a competition track, the lead car drives at a variety of speeds between 40mph (64 km/hr) and 100 mph (161 km/hr). Students "pass" the event if their driving algorithm keeps the integrated error below a certain critical value.

With a proportional controller, students get a steady state

error. It is straightforward to explain why this happens physically and mathematically. It is an authentic opportunity to introduce the final value theorem.

Fig. 4. Follow the leader event in EduTorcs.

At this point, we hand the task over to the students and ask them to figure out how to make it work. Because of noise in the inter-bumper distance measurement, a high proportional gain will not work. At moderate proportional gain, though, we have students break out their joysticks again. Thinking as a drivers rather a mathematicians, they figure out how to adjust the throttle in order to eliminate steady state error. By examining how the students' intuitive throttle adjustments vary with error, we can recognize an integral-like relationship. Again, we make sense of the concept and then investigate the mathematical ramifications.

As the semester progresses, challenges within the game get harder. For example, students develop "ride-by-wire" controllers for virtual motorcycles/bicycles. (Our motorcycle is shown on the right of Figure 1.) In this case, students/players provide desired turn rates (or desired lean angles) with the joystick. Then the controller computes a steering input that, first, keeps the bike balanced and, second, achieves the desired turn rate. It is a control problem for which performance measures such as percent overshoot, settling time, and rise time have all take on greater meaning than they do in generic homework exercises such as the V-22 Osprey problem depicted in Figure 1. If the "ride-by-wire" controller has a large overshoot or slow settling time then the bike becomes difficult to maneuver. Students work hard to understand and then to improve the controller performance characteristics so that they can execute agile maneuvers on the bike. Learning is situated in a context that has authentic meaning for students.

The course ended with a series of open-ended projects. Some of the students, for example, designed controllers that would robustly guide their cars through intricate stunt maneuvers worthy of a Hollywood action movie.

Other students designed controllers for riding a bike in reverse. This is equivalent to Klein's "unridable" rear-steered bike, a famously difficult system to control because of fundamental limitations due to an open-loop zero in the right half plane [7], [8]. Finding that students had a difficult time trusting the mathematics and simulation of the rear-steered bike, we actually built the bike (Figure 5, left side) and allowed them to attempt to ride it. This allowed students to make a direct connection between the mathematics and their physical experience. In this case, the simulated experience was not sufficient because students did not trust it.

Fig 5. Some student control projects: the rear steered bike and riding a "wheelie."

Another set of projects worth mentioning are those of students who developed controllers to balance the bike on its rear wheel. (See Figure 5, right side). Students developed one controller for longitudinal balance, i.e. keeping the front wheel off the ground, for a variety of different speeds and aerodynamic forces. Then, in order to maintain the "wheelie," students had to design a lateral controller in which rider lean is used to prevent the bike from tipping and falling to the side.

V. Assessment of Learning

When making dramatic changes to how one teaches a course it is important to measure effects of the changes as thoroughly and objectively as practically possible. In this effort, we began implementing a detailed assessment plan in the spring of 2007, the last semester that the dynamic systems and control class was taught without the video game.

To measure learning, 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 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. Concepts ranged from recognizing that complex poles produce oscillatory solutions to recognizing the utility of feed-forward mechanisms.

In Figure 6, 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 as determined by t-tests. 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.

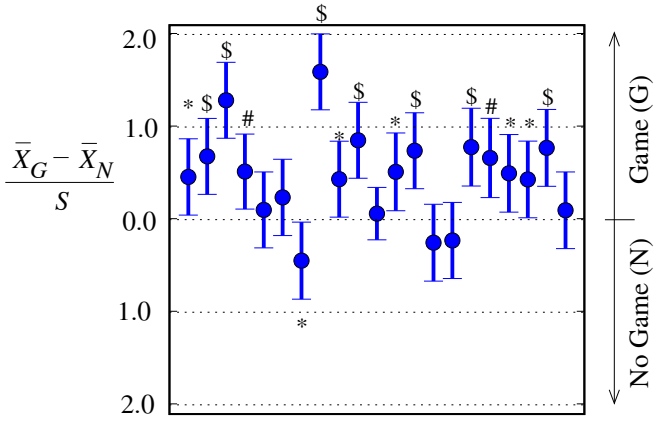

Figure 6. Normalized differences between game and non-game average scores on 21 concepts. Symbols indicate two-tailed statistical significance: *: $0.01 < p < 0.05$; #: $0.001 < p < 0.01$; $: $p < 0.001$.

Both courses were taught by the same professor, using the same textbook. Although the 2007 course was not taught with the video game, it was taught using an active-learning, inquiry-based approach outlined in Section II. Students taking both courses took the Mechanics Baseline Test [9] during the first week of the semester. Both groups of students scored almost identically on this test of mechanics and calculus knowledge as they entered the course. Therefore the only perceived difference between the groups is the type of instruction (game versus non-game) that each experienced.

The differences in concept tests scores reported in Figure 6 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.

At this stage, we have just begun analyzing the data we have collected. As we comb through the data, we will get a glimpse of how students with different learning styles, different motivation orientations, and different video game playing habits respond differently to our educational game.

In addition to the learning data outlined in Figure 6, we collected engagement data through a technique called the Experience Sampling Method [10]. Students are surveyed at random moments. Students answer questions like "Are you feeling challenged?", "Are you feeling frustrated?", "bored?", "happy?", "curious?", and others. When examined, through the perspective of flow theory [11], the answers provide insight into how engaged students are, in the moment.

Our preliminary results show that students in 2009, working on their game-based DS&C homework are more engaged than those in 2007 working on their non-game DS&C homework. The findings, so far, mirror those in our previous study of game-based numerical methods instruction. [1] The engagement findings might also partially explain why scores for the game-based group in Figure 6 are considerably higher. They might explain why we have a roughly five-fold increase in undergraduate enrollment in our intermediate dynamic systems and control class in Fall of 2010. They might also explain the dramatic jump in students this year pursuing capstone design projects with substantial DS&C components.

REFERENCES

[1] B. D. Coller and D. J. Shernoff, "Video game-based education in mechanical engineering: A look at student engagement," *International Journal of Engineering Education*. 25(2). pp. 308—318, 2009.

[2] B. D. Coller and M. J. Scott, "Effectiveness of using a video game to teach a course in mechanical engineering," *Computers & Education.* 53(?). pp. 900—912, 2009.

[3] M. J. Prince and R. M. Felder, "Inductive teaching and learning methods: Definitions, comparisons, and research bases," *Journal of Engineering Education*. 95(2). pp. 123—138, 2006.

[4] R. C. Dorf and R. H. Bishop, *Modern Control Systems*. Pearson 2008.

[5] Merlin_1. Osprey, 2007 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.

[6] F. Golnaraghi and B. C. Kuo, *Automatic Control Systems*, 9th Edition, Wiley, 2010.

[7] K. J. Astrom, R. E. Klein, and A. Lennartsson, "Bicycle dynamics and Control." *IEEE Control Systems Magazine*, pp. 26 – 47, August 2005.

[8] B. D. Coller and J. Szalko, "How to ride Klein's unridable bike." (In preparation.)

[9] D. Hestenes and M. Wells, "A mechanics baseline test." The Physics Teacher, 30, pp. 159 – 165, 1992.

[10] M. Csikszemtmihalyi and R. Larson, "Validity and reliability of the experience sampling method." *Journal of Nervous & Mental Disease.* 175 (9), pp. 525 – 536, 1987.

[11] M. Csikszemtmihalyi, *Flow: The Psychology of Optimal Experience.* Harper Perennial, 1990.