Learning Engineering Dynamics with a Videogame: A Look at How Students Play the Game

Dr. Brianno Coller, Northern Illinois University

Brianno Coller is Presidential Teaching Professor at Northern Illinois University. Early in his academic career, he studied complex dynamics and control of nonlinear systems such as turbulent boundary layers, turbomachine instabilities, aeroelastic instabilities, bicycle dynamics, and traffic. More recently he has been studying the complex nonlinear dynamics of students learning engineering in the context of a videogame.
Learning Engineering Dynamics with a Videogame: 
A Look at How Students Play the Game

Introduction

For the past few years, I have been developing a videogame called Spumone for students to play as they learn Engineering Dynamics. In the game, players/students pilot a vehicle (called the spuCraft) through a two dimensional, subterranean, simulated world. Although they are given Xbox-like gamepads to control their spuCraft, students quickly learn that most of the challenges in the game have been intentionally designed to be nearly impossible to achieve through eye-hand coordination and an intuitive understanding of the vehicle’s dynamics alone. Completing the challenges requires piloting the spuCraft with mathematical precision. To succeed in the game, students/players must leverage physical principles to derive the mathematical rules to prescribe exactly how much force to apply at exactly the right moment.

Figure 1. Screenshots of the Lift, Spiro, and Espoo challenges in the videogame Spumone.

As we have been testing and developing the game these past few years with students taking their sophomore level engineering dynamics course, we have been monitoring their conceptual understanding of the subject with the Dynamics Concept Inventory. In each semester we have used the game, students scored roughly 0.8 standard deviations higher compared to a control group of students which took the course before Spumone was available. Results were statistically significant. Furthermore, our results with Spumone were consistent with previous successes in incorporating videogames into upper level dynamic systems & control courses.

Our results also align with the message coming from a growing group of education scholars who have been urging educators from all domains to learn lessons about how good videogames engage their players in complex problem solving tasks. When one observes people playing videogames, one finds that they plunge into it. They have no need for a manual. Goals are clear and feedback as to whether they are achieving those goals is immediate, abundant and unambiguous. Players are able to achieve initial success fairly quickly, but challenges intensify progressively to keep players at the edge of their abilities. Therefore, time on task is neither mundanely repetitious nor overwhelmingly difficult. Furthermore, to help players progress, the most successful videogames establish environments that encourage active and critical learning and have incorporated, whether intentionally or by accident, highly effective learning strategies into the game play. Players/learners can take risks in a space where real-world consequences are lowered. Necessary knowledge and skills are discovered “from the bottom up” through a cyclic
process of hypothesizing, probing, and reflecting. Information is given to players/learners at just the time they will be able to make sense of it and to use it. In a videogame, knowledge is powerful because it can be put to productive use.

I make no claim that Spumone measures up to the ideal playing/learning environment described above. However, it would be interesting to take a deeper look into how students are using Spumone, and to look for affordances provided by the game that are benefiting the learning process. The study described in this paper is more exploratory in nature, with a goal of finding discernible patterns of play and patterns of learning within the “click stream” captured by the game log files.

Videogame Challenge: Spumone Drop

Spumone contains more than a dozen challenges and sub-challenges in which students have to put concepts from engineering dynamics to practical use. Topics span most of the sophomore-level course: direct $F = ma$, work-energy principle, impulse-momentum principle, angular impulse-momentum, polar coordinates, rigid body kinematics, and rigid body kinetics.

In this paper, we focus on the first challenge that students encounter in Spumone. It is called “Drop.” The objective in designing the drop challenge was to provide a simple game-based problem with which players/students could achieve a quick success, yet be forced to confront a common conceptual hurdle that many novice students who have passed their first calculus-based physics course have not overcome.

The story within Spumone begins with the little spuCraft being dropped into a deep hole, for reasons that are not known to the player. When a player pushes upward on a gamepad thumb stick, a signal is relayed to all three thrusters of the spuCraft. Since the thrusters and the masses are symmetric about the vertical axis through the center of the vehicle, there is no net moment about the center of mass. Thus, the craft does not rotate. All forces (gravity and thrust) are vertical; the dynamics of the spuCraft are one dimensional (assuming one does not crash, get “flowered”, or shot by lasers).

![Figure 2](image)

Figure 2. Screenshot of the first part of the Spumone Drop challenge along with a schematic diagram of the spuCraft for the Drop World.
**First Part.** A couple hundred meters down the hole there is a bridge and a landing pad upon which the student/player must pilot the craft to a gentle stop. The legs of the spuCraft are fragile and easily break off during hard landings. This first part of the challenge is a simple, fun, warm-up exercise where students play *Spumone* like a traditional videogame. It is similar to a 1-D version of the classic arcade game *Lunar Lander*. As students pilot the spuCraft, arrows attached to the center of mass of the vehicle indicate its current velocity and current acceleration. As shown in Figure 2, position, velocity, and acceleration are displayed numerically on the right side of the game screen. A recent history of velocity and acceleration are plotted in the lower right corner. The purpose of this first part is for students to get a *feel* for the relationships between position, velocity, and acceleration, to complement their mathematical definitions of the dynamic states.

**Second Part.** After safely landing on the bridge (with a sparkler that the student needs to stop and pick up along the way), the bridge explodes and the spuCraft, once again, is dropping. This begins the challenging part of the assignment.

Part way down the hole, one encounters a vertical conveyor belt attached to the side wall. See Figure 3a. The side of the belt facing the spuCraft is moving downward with a constant velocity which is displayed on a billboard. Also shown on the billboard is the velocity of the spuCraft.

![Figure 3](image.png)

*Figure 3.* Partial screenshots showing aspects of the bottom half of the *Spumone* Drop challenge.
As the spuCraft crosses the light gate (orange line in Figure 3a), a sticky red pellet is deposited onto the belt. As shown in Figures 3b and 3c, the pellet emits a thin horizontal beam of light that travels with the pellet, at the constant speed of the belt.

Meanwhile, a bunch of slowly drifting flowers start crossing the path of the spuCraft. The flowers look innocent enough. But looks can be deceiving. When any of the flowers come in contact with the spuCraft, that part of the vehicle immediately evaporates. This usually breaks the reflection symmetry of the craft. It veers off course and typically crashes into a side wall, sometimes spectacularly. The challenge attempt is unsuccessful.

To protect the spuCraft from falling flowers, one must activate the halo as shown in Figure 3c. To generate the halo, one must fly the spuCraft so that its halo sensor lies completely within the narrow light beam emanating from the sticky pellet. This is delicate. The velocity of the spuCraft must be constant and the constant velocity must exactly match that of the belt, and the light beam and halo sensor have to be aligned. Under these conditions, the halo provides a shield that pops the flowers, converting them into clouds of harmless dust. Any misalignment, though, the halo disappears, leaving the spuCraft vulnerable.

If one is able to travel the full length of the first belt without dying, one will find a second belt moving at a faster rate through an even denser field of flowers. While playing Drop in the “Novice” mode, getting past the second belt intact will be sufficient to successfully complete the challenge. When in intermediate mode, one must make it past the first two belts and then pass a third (after lasers shoot off the feet of the spuCraft, giving it a different mass). In “Expert” mode, one must pass all three belts in a very short amount of time.

**Engineering Problem Solving and Spumone Drop**

The Drop challenge, like most Spumone challenges has been carefully designed to be nearly impossible to complete manually, with gamepad and eye-hand coordination alone. To keep the spuCraft halo sensor within the narrow beam of light for dozens of meters of travel requires superhuman precision. Furthermore, each time one runs Spumone Drop, the mass of the spuCraft is different and the belt speeds are different. Therefore, solutions to the problem are quantitatively different each time the challenge is attempted. Thus, the degree to which Spumone is a game, it is primarily a game about engineering problem solving. In order to successfully complete the Drop challenge, a student is going to have to solve a dynamics problem – an engineering problem – so that they can derive a mathematically precise expression for how much thrust to generate, and when to generate it.

**Model of Problem Solving in Engineering Dynamics.** As we look at examples of students interacting with the game in this article, we observe them work through an engineering problem-solving process. The formal problem solving process which students began learning in their engineering statics, is developed further in engineering dynamics and is depicted diagrammatically in Figure 4.

McCracken and Newstetter interpret the problem solving process which we teach our students as process through which one attempts to transform a problem, initially posed in a textual
representation, into diagrammatic representations (e.g. free body diagrams), and symbolic representations (e.g. kinematic and kinetic equations). The model depicted in Figure 4, contains these different representations along with the processes (“Evaluation”, “Modeling”, “Formulation”, “Manipulation”, “Check”) by which one generates and interprets the different representations.

Figure 4. A model of problem solving in Engineering dynamics.

The problem solving model depicted in Figure 4 has much in common with that proposed by Litzinger et al, also patterned after McCracken and Newstetter, to study problem solving in engineering statics. Also, there is considerable overlap with the structured approach discussed by Costanzo & Gray.

To concretize the discussion of the model in Figure 4, I provide a hand-written example, in Figure 5, of how I would expect students to solve the novice-level Drop challenge, given how they have been trained to solve textbook problems. The figure labels different stages of the problem solving model. For example, the first two lines of Figure 5, are labeled Boundary Conditions. In this step, students are to explicitly list the parameters that are given, as well as initial and final conditions. The “Given” and “Find” fields also represent the boundary conditions of the problem solving process itself, listing the information one begins with and the final goal state of the task. In Figure 4, I use the term Evaluation to indicate the process of generating Boundary Conditions. For typical textbook problems, evaluating the boundary conditions is often a matter of extracting relevant text from the textual problem statement. In the Spumone exercise, determining boundary conditions requires more of an evaluation of how the simulated system – with its conveyor belts, sticky pellets, light beams, and halos – works.

Next, in Figure 5, come the Diagrammatic Representations of the problem in the form of a Free Body Diagram (FBD) and Mass-Acceleration Diagram (MAD). According to the terminology in
Figure 4, the diagrams are obtained through a process of *Modeling*. This is followed by a process of *Formulation* to generate a *Symbolic Representation* of the system, using Newton’s second law. By *Manipulating* the symbols and graphically applying kinematic relationships, one is able to obtain a *Result*.

*Figure 5.* Example solution to the novice-level Drop Challenge.
One small Check of the result is shown in Figure 5 when I explicitly verify that the expression for thrust has proper units. However, the ultimate Check of whether the result is valid is by implementing it within Spumone, and verifying that it works. This is considerably different from checking a traditional textbook problem where one verifies a result by comparing it to an answer posted in the back of the book, or by having another authority (e.g. instructor, teaching assistant) grade the homework.

**Motion implies a force misunderstanding.** As indicated in Figure 5, solving the novice-level Drop challenge is a matter of recognizing that the spuCraft must travel at constant velocity as it passes the belt. In order to move with zero acceleration, all forces must balance. One simply has to turn on the weight-balancing force precisely when the velocity of the spuCraft matches that of the belt.

The solution may seem almost trivial to the seasoned dynamics instructor. However, it bumps up against some deep-rooted, false beliefs that many people inherently possess, even those who have taken courses in physics. This is the Aristotelian view that an object moves in the direction it is pushed. And if the forces acting on the object add up to zero, according to this reasoning, then the object must come to a stop. The misunderstanding comes from a lifetime of observing a world with friction forces that bring objects to a stop.

Thus, the goal of the novice Drop challenge is to provide an opportunity for many students to confront this misunderstanding, an experience that gets repeated in other Spumone challenges.

**Implementing a Solution in Spumone.** Once a student has a “solution” to a Spumone problem, they may implement it through an interface called the SpuPilot. Below, I provide a brief overview of the interface so that when I present logs of student work, later in the article, the reader can understand what they are doing.

![Figure 6](image_url). GUI where students write mathematical equations for the thrust signal.

One of the windows in the SpuPilot provides a graphical user interface (GUI) where students can write mathematical equations for the thrust signal to be sent to the spuCraft. See Figure 6. Variables that students can use to write their equations are: \(m\), the spuCraft mass; \(m_{Feet}\), the...
mass of only the feet of the spuCraft; \( g \), acceleration due to gravity in the Drop world (not 9.81 m/s\(^2\)); \( \text{maxThrust} \), the total maximum force that the thrusters can produce; \( v_y \), the vertical component of velocity of the spuCraft; \( v_b1, v_b2, v_b3 \), velocities of the three belts; and \( d_1, d_2, d_3 \), distances of the halo sensor from the light gates of each of the three belts.

If the mathematical function in the equation box evaluates to a number between 0.0 and 120.0, the maximum total force that the thrusters can produce (\( \text{maxThrust} \)), then the three thrusters combined produce a thrust equal to that number.

Figure 6a shows what the player/student sees the first time she enters the Drop portion of SpuPilot. The only input script is the default script called “Manual”:

\[
\text{maxThrust} \times (\text{stickVR}() + \text{arrowUp}())
\]

This simply tells \textit{Spumone} to assign a thrust signal depending on the amount the right thumbstick of the gamepad has been deflected vertically by the player, and whether the up arrow on the keyboard is being pressed.

To give a thrust command which balances the weight of the spuCraft is shown in Figure 6b. Here, we created a new input script with the name “weightless,” with the simple function \( m \times g \).

Although we created a new input script, \textit{Spumone} defaults to the “Manual” script when one enters the Drop World. To get \textit{Spumone} to switch over to the “Weightless” script, one has to create a “Switch” as shown in Figure 7a. It is quite simple. One simply chooses an input script (in this case Weightless) and a button (in this case Button 1) on the gamepad that one wants to associate with the input script. This will cause \textit{Spumone} to switch to “Weightless” whenever button 1 is pressed. One can create a second switch in the SpuPilot to switch back to “Manual” mode whenever a second button is pressed. With this setup, a player could switch back and forth between the two input scripts, just by pressing buttons on the gamepad.

The reader may recall that in \textit{Spumone} Drop, one wants to switch to the “Weightless” script at exactly the moment that the vertical velocity of the spuCraft matches that of the belt. This is going to be nearly impossible to achieve by pressing Button 1 at precisely the right time. It is even more difficult to do so twice consecutively on belts traveling at different speeds.

Fortunately one can write a rule in the SpuPilot to automatically activate a switch when specific mathematical conditions are satisfied. This precise action is is what happens in Figure 7b when one defines an automatic switching condition. One simply checks a box, and writes the equation

\[
v_y = v_b1
\]

in the text box. Under this scenario, as soon as the player pushes Button 1, switch is armed and \textit{Spumone} begins evaluating the “Automatic switching condition” hundreds of times per second. When the equality \( v_y = v_b1 \) is satisfied, the weightless script is automatically engaged. The spuCraft should be able to safely traverse the length of first belt. Belt 2 and Belt 3 can be programmed similarly.
Data Collection

The data reported here were collected in the Fall semester of 2012. In that semester, 32 students took the sophomore level engineering dynamics course at Northern Illinois University and completed it. All but two students provided consent to their course data (including test scores, survey responses, and Spumone logs) being used for research purposes. Data from these two students have been omitted from the study. In addition, data from two other students, deemed “atypical” were omitted. One was a student repeating the course for the third time. The other was a graduate student taking the course for special reasons.

Timing. Students were given Spumone, a gamepad, and the Drop assignment at the beginning of the third week of the semester. By that time, all students had successfully completed “Mastery Quizzes” on qualitative 1-D kinematics and 1-D dynamics of particles. For these quizzes, students are required to take and re-take different versions of the quiz, in the presence of the instructor or a teaching assistant, until they perform flawlessly. Therefore, prior to the Spumone Drop assignment, all students had demonstrated that they can perform the steps of the engineering problem solving process outlined in Figure 4, at a level necessary to complete the videogame-based challenge. Most students had also completed typical textbook-like homework assignment which gave them additional experience with the problem solving process for engineering dynamics.

Figure 7. Switch windows in the SpuPilot.
Spumone Logs. As students “played” Spumone throughout the semester, the game generated a detailed log file, recording every instance they ran the program, and “significant” events that occurred in each program run. When students ran the Drop challenge, specifically, Spumone recorded the following information for the second part of the challenge:

- Whenever a student entered the Drop challenge, the game logged the date, time, and which mode (Practice, Novice, Intermediate, or Expert) was running, the starting point of the challenge (top of the hole, or the bridge), and a time stamp for the SpuPilot output file that was being used.
- If SpuPilot scripts had been changed since the previous run, a copy of the SpuPilot output file was pasted into the log so that one could examine the scripts that students wrote.
- Every use of a manual switch or automatic switch was recorded. The log file indicated which was used.
- Every crash into falling flowers, along with with the belt number where the crash occurred was recorded.
- Every crash into the walls, including conveyor belt and other machinery was recorded, along with approximate location of the crash.
- Every successful completion of a challenge, including time to complete and number of flowers destroyed by the halo was recorded.

All log files were encrypted so that students were not able to read or modify them. Students were informed, verbally and in writing, that their activities with Spumone were being logged on their local machine. At the end of the semester, 24 students voluntarily submitted their log files via email to the instructor. Three of the students either “lost” files or experienced computer crashes that wiped out data from the early part of the semester. Other students simply chose not to submit their log data, but they did not rescind permission for using other course data.

Examination of How Students Played the Game

On average, the 21 students submitting logs for the entire term, played Spumone 490 times during the 15 week semester. The most number of plays was 1082. The least was 201. In an attempt to distill useful themes from an enormous amount of log data, I wrote Python scripts that generated bar charts of categories of events. I carefully read the hundreds of SpuPilot output files corresponding to the Novice level of the Drop challenge.

Examination of the log data revealed an uncommonly detailed view into students’ problem solving processes. Normally when a student turns in an assignment, we get a heavily redacted expression of their effort. Omitted from the submission are all the false starts and mistakes that the student is aware of making. In contrast, the Spumone log files contain all instances where a student tried out an idea, in the virtual environment of Drop World. The explorations, failures, and successes are all recorded chronologically.

The drawback of the log file data is that they only contain direct artifacts from a small portion of the problem solving process outlined in Figure 4. We only get to directly observe the Results block of the diagram, and the consequences of Checking those Results in the game. However,
given the relative simplicity of the Drop challenge, it was thought that it might be possible to infer what is happening in other parts of the problem solving process.

Rather than overwhelm the reader with log data from all 21 students, I present the activities of three students labeled P, Q, and R whom I believe to be representative, in a macro sense, of the different ways in which students played the Novice level of the Spumone Drop challenge. The letters P, Q, and R were chosen for convenience. Any relation that may exist between these letters and the students’ actual names is coincidental.

**Student P.** The way that Student P played the novice and practice levels of the Drop challenge is depicted in Figure 8. The horizontal lines represent timelines on two separate days. Vertical lines above the horizontal axis represent instances in which the student played the game. The practice level is essentially the same as the novice level, except there are no flowers to kill the spuCraft. The purpose of the practice level is to provide a setting in which students could safely explore the dynamics of the spuCraft and test ideas.

Vertical lines below the horizontal axis indicate instance in which the game was played with a newly revised set of input scripts, switches, user-defined functions, or other settings controlled by the SpuPilot. If you count the marks, you will find that Student P played the game 24 times at this level, and made changes in the SpuPilot 11 times.

Details of Student P’s activities are provided in the Appendix. In summary, the record shows that this student took a systematic and well thought out route to solving the problem. After landing successfully on the bridge and then unsuccessfully attempting through fly in the bottom, flower-laden part of the domain a few times, it is evident that Student P watched the online tutorial on writing input scripts and programming manual switches. He/she came back after a from a 15 minute break with examples from the tutorial programmed into the SpuPilot.

After that, it appears that Student P explored the dynamics of the spuCraft. He/she kept the game in the practice level, and started experimenting with input scripts quite different from those.
covered in tutorial. Then, after a short break, Student P wrote a script indicating that the thrusters should produce a force equal to the weight of the vehicle. At this point it is clear that the student understood what is required in the problem (*Boundary Conditions*), and that he/she had successfully worked through the *Modeling* and *Formulation* phases of the problem solving process. The only problem was that the input script was not scaled properly for the previous version of the SpuPilot. The problem was quickly resolved and Student P had a spuCraft which would descend at constant velocity.

The only part left was to work through the kinematics (*Manipulation* phase) of getting the spuCraft to descend at the correct velocity. The student watched video on setting up automatic switching conditions. At that point, it was clear that he/she understood what to do, but entered the switch condition incorrectly. The student was able to fix the problem quickly and pilot the spuCraft through the first belt unscathed. On the next attempt, Student P had an appropriate automatic switching condition set up for the second belt as well and successfully completed the novice level Drop challenge. He/she completed it a couple more times, demonstrating a robust engineering solution.

Student P’s work is characterized by an extraordinarily methodical approach to solving the problem. The task was split into parts. When a subpart was complete, it was tested at least twice. The two time gaps in the record indicate evidence of metacognitive skill. The student recognized when he/she did not understand something, and turned directly to the tutorial videos. The student implemented the examples in the videos rather than passively watching them, and quickly transferred information contained in the video to solutions that solved the problem.

Student P took about an hour, including video tutorials, to complete the novice level Drop challenge. He/she went on to solve the Intermediate Drop challenge in seven plays and three pilots.

**Student Q.** One look at Figure 9 makes it clear that the usage profile for Student Q is markedly different than that of Student P. In this case the student played Spumone 207 different times and tested 39 different pilots before coming to a solution. He/She did all his/her work on the Novice Drop challenge on a single day, over a span of about 5 hours (with gaps). Many details of Student Q’s work are provided in the Appendix.

After successfully landing on the bridge, Student Q spent the next 35 minutes flying the spuCraft manually in the bottom half of the Drop world. It is not clear from the log what he/she was trying to accomplish during this time. He/She might have been exploring the dynamics of the spuCraft somewhat, but the fact that Student Q was able to make it safely past the first belt once during the 35 minutes suggest that he/she was deliberately spending a significant amount of time in interval 1 trying to complete the challenge without performing any analysis. Technically speaking, Student Q was attempting to “game the system.”

---

20, 21
After interval 1, though, it appears that this student saw the futility in his/her approach and had given up on manual flight. After the gap in time, we see evidence of having watched the video tutorials: scripts from the tutorial appear in the log. In particular, it appears that the student was fascinated with one of the thruster scripts of the form $\frac{vy}{vb1}$. He/She attempted several different variations of this form. What makes this choice interesting is that I suspect that the student observed from the tutorial that a thruster script of this form produce a behavior in which the spuCraft approaches (asymptotically) a constant terminal velocity. It appears that Student Q had successfully created valid Boundary Conditions for the system but had failed to follow the other steps in Figure 4. The expression $\frac{vy}{vb1}$ and all the other permutations he/she tried did not come from any physical principles. Instead, it appears to have been a Ptolemaic attempt at equation-fitting. He/She was trying to find the correct combination of symbols that would yield the correct terminal velocity, without success.

One of the problems with this approach that the student appeared to discover was that the mass of the spuCraft was different each time the program was run. Therefore the terminal velocity was different each time, even if the input script was not changed. Suddenly, Student Q started including mass in his/her permutation of equations.

Next, the student started squaring velocities. He/she must have felt like this was a productive path, because it continued for more than a dozen iterations. Eventually, the sequence of scripts evolved into term that looked a lot like kinetic energy and other terms that looked like work. We had not yet covered work energy in the engineering dynamics course. It appears as though he/she was referring back to notions of the principle he/she had learned in a prior physics class, albeit incorrectly. The important point is that he/she started to think of this problem as a physics problem and that stuff learned in class might be relevant.

After this watershed moment, the fact that the thrust force shall equal the weight came rather quickly. However, it took a long time for the student to figure out how to use automatic switches to get the right velocity. Instead of using an automatic switch, Student Q did something very clever. Referring back to his/her experience with ratios of velocities, he/she chose a thruster script of the form

\[
vy * m * g / vb1 .
\]
Notice that when the speed of the spuCraft matches that of the belt, $\frac{v_y}{v_{b1}}$ is equal to 1.0 and the thrust force exactly balances the weight as desired. If the spuCraft is moving too slowly, then $\frac{v_y}{v_{b1}}$ is less than 1.0 and the thrust force is less than the weight, causing the craft to speed up. Likewise, if the craft was moving too fast, the thrust would be bigger than the weight, causing it to slow down. Student Q had created a nice feedback controller, and it was completely intentional. In the student’s write-up that he/she turned in, he/she called it the “auto-correct.”

Unfortunately the auto-correct did not converge quickly enough to be of use. The student had to eventually learn how to use the automatic switch, and was finally able to complete the challenge. Student Q was able to complete the intermediate level of Drop in six tries, making just one change to the SpuPilot.

Although Student Q’s problem solving process was torturous to read through, in the end, he/she was able to solve the problem as intended. Furthermore, it is fair to say that Student Q’s work was more inventive and creative than the cleaner and more intellectually pure work of Student P. In the end, Student Q might have learned more.

**Student R.** Student R played the novice and practice levels of *Spumone* Drop a total of 87 times, using 47 different pilots over a time span of nearly four hours (including gaps). The activity profile is shown in Figure 10.

![Figure 10](image)

*Figure 10. Problem-solving and game-play activity profile for Student R.*

Except for the period in which Student Q apparently tried to “game the system.” The activity of Student R looks very similar to the early equation-fitting activity of Student Q. The equations that Student R experimented with did not appear to come from any physically-based problem solving process. Units are inconsistent between terms. Few terms have any of the canonical forms (e.g. work, energy, momentum) that come from physical principles. The most important difference between Student R and Student Q, is that Student R did not show any clear sign of growth or development over the course of the game session. Eventually, Student R did provide a solution that worked, but it came instantly, almost out of nowhere, as if one of the student’s classmates had given him/her the “correct” answer.
After completing the novice level of Drop, Student R was not able to quickly build upon the previous solution and complete the intermediate-level Drop challenge. After a while, a working solution for intermediate level suddenly appeared, without any clear rationale in the log file, just as it had appeared in the novice level.

Discussion & Reflection

Although these are just three out of the 21 different log files examined (by human eyes), the patterns of playing Spumone Drop Novice appeared remarkably robust. Seventeen of students seemed to fall squarely in one of three categories characterized by Students P, Q, and R.

Overall, 5 of the 21 students showed evidence of fluency in the problem solving process, similar to Student P. That is, these five students had log files which indicated that they quickly embarked on an engineering problem-solving approach to the problem, similar to Figure 4, and demonstrated proficiency in the approach.

Eight of the students showed characteristics more similar to Student Q in that they spent considerable time trying to get non-physical equation to exhibit the correct behavior. After initially flailing, all these students/players eventually realized that the problem-solving framework and conceptual ideas we had been developing in class would be the most appropriate way to approach the game-based challenge.

Sadly, four students total exhibited characteristics similar to Student R. These are the students who started off by equation-fitting and never showed evidence of developing toward a physics-based, engineering, problem-solving approach.

When first discovering these results, I was a bit surprised, particularly about group Q. As an instructor, I try to be optimistic about students turning in work that reflects their own understanding, rather than a classmate’s understanding. For the most part, there is evidence to support this optimism, at least in the first Spumone assignment. Given that the Spumone write-ups that students turn in, as part of the assignment, are generally of high quality, I was surprised to see the long arduous path that it took for many students to get there. This was particularly surprising, given the relatively low level of difficulty I considered the problem to be. It was also surprising given that all students had recently shown competence at solving more difficult one-dimensional dynamics problems in mastery quizzes.

Cognitive Resources. Interestingly, back in the early 1980s, diSessa ran a computer-based experiment in which he had middle school students pilot a virtual turtle around a two dimensional, frictionless, simulated world. The objective was to get the turtle from one corner of the screen to the diagonally opposite corner by giving it impulses that were geometrically constrained to certain direction. The kids were stymied by the Newtonian dynamics. They expected the momentum in a certain direction to immediately halt when the object was given an impulse in a perpendicular direction. When shown that a ball rolling on a table exhibited the same behavior as the virtual turtle, they still didn’t believe it.
The most interesting part of the study was when diSessa was able to get a freshman M.I.T. student who had successfully completed several high school and college physics courses to participate in the same computer-based activity. Surprisingly, the M.I.T. student exhibited unworkable strategies and understandings remarkably similar to those of the “naïve” twelve year old children. As diSessa explains:

… she did not, indeed for a time could not, relate the task to all the classroom physics she had had. It is not that she could not make the classroom analysis… It is more that her naïve physics and classroom physics stood side by side but unrelated, and in this instance, she exercised her naïve physics.

However, late in the session, diSessa describes a watershed moment in which the M.I.T. student made the connection between what was happening on the computer screen and the formal physics knowledge she possessed. At this “crucial stage,” she began interpreting all her previous observations into the scientific framework and the problem made sense. Using the classification scheme put forward in this paper, it is clear that this M.I.T. student would reside in group Q.

A quick review of the literature shows that the phenomenon of students neglecting their formal training when trying to solve physics problems that are somewhat different than the classical textbook problems they are accustomed to, is not uncommon. Mazur reports that when giving concept problems to students, one asked whether they should answer the questions according to what was taught or “by the way I think about these things?”

According to Redish’s Resource Model of how students learn physics, knowledge is a collection of elements of “resources” that are linked together by cognitive construction. The resources exist in long-term memory. When transferred to working knowledge, they are easily processed by the individual as a seemingly coherent manifestation of truth. Knowledge structures are the connections between knowledge elements that form a resource. Although some have used the term schema, Redish uses “knowledge structure” to emphasize that such structures are often in a state of flux for novices.

Resources and knowledge structures of experts are relatively complete, connected, highly organized, and internally and externally consistent. In contrast, the resources and knowledge structures of the novice are smaller, and more fragmented. They may be internally and/or externally contradictory.

Another important distinction between experts and novices, for our purposes, is that of control. This is the process by which individuals activate (or elicit) specific resources in certain contexts. Again, experts have highly refined control mechanisms which effortlessly activate resources that are appropriate and helpful in the context of the problem being solved. As we have seen in this paper, novices are less capable of activating the appropriate resources in new contexts. From this perspective, our groupings P, Q, and R, refer to students’ ability to activate problem solving resources in new contexts, perhaps videogame contexts in particular. It is hoped that making students solve problems in different contexts can help students develop important cognitive connections.
**Performance on concept tests.** The primary measure that we have been using to quantify the impact of the Spumone on learning has been concept tests consisting primarily of qualitative questions. As part of this research I have been fond of plotting pretest scores on one axis and posttest scores on the other axis\(^{26}\) as shown in Figure 10. “Pretest” refers to students’ performance on a set of questions taken from the Force Concept Inventory and the Mechanics Baseline Test. “Posttest” refers to students’ performance on the Dynamics Concept Inventory. The scores shown in Figure 10 have been normalized to zero mean and a standard deviation of 1.0.

I have found this plot interesting because, before Spumone was integrated into the engineering dynamics course, the correlation between posttest performance and pretest performance was significantly correlated. Students who performed well on the pretest ended up doing well on the posttest, and vice versa. However, after introducing Spumone, much of the correlation disappeared\(^{26}\). Figure 10 shows the relatively weak correlation between posttest and pretest.

![Figure 10. Pretest-posttest scatter plot with labels indicating categorizations of students.](image)

For this study, I find it interesting to overlay a line of slope 1.0, passing through the point (0,0). Students whose pretest/posttest data points lie above the line are those who gained, relative to their classmates, over the duration of the semester. Students who lie below the line declined relative to their classmates.

In Figure 10, I have also added the P, Q, and R designations to the plot of each student who turned in their log file. Visual inspection seems to indicate that students in the P group tend to lie above the line while those in the R group lie below the line. Thus, the ability to map cognitive
resources associated with problem solving onto the videogame context might be a predictor of course performance. Recall that the data for this study were collected in week three of a fifteen week semester. If one could efficiently categorize students into P, Q, and R groups, based on Spumone log data, then one might have a good diagnostic tool for identifying at-risk students. Obviously, the sample is small and more research needs to be conducted.

References

2. Coller, B.D., Preliminary results on using a video game in teaching dynamics, in ASEE Annual Conference. 2012: San Antonio, TX.
3. Coller, B.D., First Look at a Video Game for Teaching Dynamics, in ASEE Annual Conference. 2011: Vancouver, BC.


APPENDIX

Details of Student P’s interaction. Student P modified the SpuPilot 11 times. The 11 modifications are labeled in Figure 7 so that I can refer to them below.

- In the first play of the game, Student P landed successfully on the bridge. The first play was in the Practice level allowing the student to see the entire domain for the Novice level.
- Before playing the game a second time, Student P entered the SpuPilot and switched to Novice level and switched the starting point to the bridge so that he/she would not have to land on the bridge again.
- After playing a second time, there is an 18 minute gap. During that gap, it is clear that Student P watched the 9-minute tutorial video on defining input scripts and switches within the SpuPilot. It is clear because the Pilot 2 scripts have examples from the tutorial coded in and switches back to Practice mode. The student tests the example scripts and switches a few times.
- In Pilot 3, he/she switches back to Novice mode and runs the game a few more times without using switches.
- In Pilot 4, the student defines one of the input scripts as vb1. I’m not sure what he was thinking here, but the result could not have been too satisfying because vb1 is negative, which would get mapped to zero thrust.
- In Pilot 5, the student makes a breakthrough. He/she writes the input script as m*g, apparently recognizing that the correct thrust is one which perfectly balances the weight. He/she tests the results twice. Since the student neglected to scale by maxThrust the script evaluates to a number much greater than one, producing a full thrust and a broken spuCraft.
- The next day, in Pilot 6, Student P starts with a new input script: m*g/1.2. It appears that he/she is trying to scale the thrust to get a proper thrust signal, but the scaling coefficient is off by a factor of exactly 100. Not sure what is happening.
- Apparently dissatisfied with the previous script, he/she tries g/m in Pilot 7. It seems more like a haphazard permutation of the symbols rather than the result of serious thought.
- Bingo! In Pilot 8 gets the correct input script m*g/maxThrust. There are no automatic switches programmed yet, only manual switches.
- Results of testing must be good. In Pilot 9, the student shifts focus from getting the right amount of thrust to turning the thrust on at the right instant. He/she enter vb1 into the automatic switching condition. The student is on the right track; he/she just has not expressed the switching condition correctly.
- Now there is a 15 minute gap. My guess is that the student spent it watching the second tutorial video on setting up automatic switching conditions. When he/she comes back in Pilot 10, the condition in the first automatic switch is set up correctly. Testing shows that the spuCraft successfully made it past the first belt and crashed on the second.
- In Pilot 11 (the final pilot), the student correctly defines the automatic switching condition for the second belt. After a couple minor miscues while flying the spuCraft, the student successfully completes the Novice challenge as indicated by the first red oval.
• Later in the day, Student P shows that is successful and repeats it two more times.

Details of Student P’s interaction. Intervals are labeled in Figure 8.

• After landing on the bridge safely and switching over to novice mode, Student Q spends the next 35 minutes (Interval 1) trying to complete the challenge by controlling the spuCraft manually, without any input scripts or switches. To me, it seems like a futile exercise. Occasionally the student gets past the first belt.

• During Interval 2, there is a 35 minute gap during which there is very little activity in the log record. However, at the end of the gap, there is an input script $v/vb1$. This is one of the examples in the first video tutorial on input scripts and switches.

• During Interval 3, the student experiments with input scripts such as $vbl - m\times g$; and $(-vb1+vy+g)/120$; and $(-vb1+vy+g)$.

• During the transition between Interval 3 and Interval 4, the student reverts back to input scripts that take the form of ratios of velocities $(vb1-vy)/vb1$; and then back to $vy/vb1$ which was one of the examples in the tutorial.

• In Interval 4, Student Q begins experimenting with input scripts that take the form of ratios of kinetic energies: for example
  $$\frac{(1/2)*m*vy^2/d1}{(1/2)*m*vb1^2};$$
  $$\frac{(1/2)*m*vy^2}{(1/2)*m*vb1^2}. Why kinetic energies?$$

• Interval 5 is relatively short in duration, but contains a flurry of activity. Here, the student systematically changes a constant in the denominator. For example:
  $$\frac{(1/2)*m*vy^2}{(2/3)*m*vb1^2};$$
  $$\frac{(1/2)*m*vy^2}{(5/6)*m*vb1^2};$$
  $$\frac{(1/2)*m*vy^2}{(11/12)*m*vb1^2};$$
  and the pattern keeps going. It appears to be a brute-force attempt to find a script that produces a constant velocity at the velocity of the first belt. Eventually, the student makes a guess that overshoots. Then he/she begins to backtrack. The process is bound to lead to frustration. Every time the student plays the Drop challenge, the craft has a different mass and the belt has different speed. The process Student Q is pursuing will not converge. This probably explained why he/she abandoned it after many iterations.

• Interval 6 follows a fairly large gap in time. It appears that the student used that time to think more deeply about the problem. The pilots proposed in Interval 6 take the form
  $$\frac{(((.5*m*vb1^2)-(0.5*m*vy^2))/d1)+m*g}{maxthrust}. Now we have equations that genuinely come from the work energy principle. Note also that the results are scaled properly. Unfortunately, one cannot use this formulation to solve the Spumone Drop problem; it will not work.

• Interval 7 follows another gap in time. Again, it is fruitful. The student comes up with the following input script: $vy/vb1*(m*g)/120$. This is real clever. Notice that when vy and vb1 match, the input script becomes $m*g/maxThrust$, exactly what we said it should be. But when the velocities do not match, the velocity ratio provides a feedback effect. When the velocity is too slow, the thrust decreases so that spuCraft speeds up, and vice versa. The student recognizes this. In his/her write-up, he/she calls it the “auto correct”.
In principle, the student does not need a trigger for this input script because the feedback automatically brings the craft to the right speed. In practice, however, the transients take too long to die out and the spuCraft flames out.

After nearly five hours of work, the student finally figures out triggers and gets it all to work.

**Details of Student R’s interactions.** The activity profile is displayed in Figure 9.

This student showed very little evidence of using Newton’s laws or any other physical principles to solve the problem. Below is a list of input scripts the student tried:

\[
\begin{align*}
g &- (\text{maxThrust}/m); \text{and } vb1/vy; \text{and } vb1; \text{and } \text{maxThrust} \times (vb1/vy); \text{and } \\
vb1/\text{maxThrust}; \text{and } 1-(vy/vb1); \text{and } 2*(vy/vb1); \text{and } (vy/vb1)/2; \text{and } \\
(vy/vb1)/3; \text{and } (m*g)/vb1; \text{and } (vy-g)/vb1; \text{and } -1*vb1/vy; \text{and } \\
(-1*vb1/vy)/\text{maxThrust}; \text{and } (1-(vy/vb1))/\text{maxThrust}; \text{and } m*g/vb1; \text{and } \\
(vy/vb2)*\text{maxThrust}; \text{and } (vy/vb1)*(vb1/vy); \text{and } (vy/vb1)/m*g; \text{and } \\
(vb1/vy)/m*g; \text{and } m/\text{maxThrust}; \text{and } \text{maxThrust}/m; \text{and } m*g; \text{and } \\
(vb1/vy)*g; \text{and } (vb1-vy)/vb1; \text{and } (vy-vb1)/vb1; \text{and } \\
\text{maxThrust}*(vy-d1); \text{and } (vy/vb1)*g; \text{and } 0.1; \text{and } 0.1*m*g; \text{and } d1; \text{and } \\
0.5*m*g; \text{and } m*g/\text{maxThrust}; \text{and } 0.9*m*g/\text{maxThrust}.
\end{align*}
\]

It looks as if the student, almost randomly, tried 32 permutations of the available symbols until he stumbled upon \(m*g/\text{maxThrust}\), indicated by the asterisk in Figure 8. At this point, he/she made several attempts to complete the challenge using the switch to turn on the thrust manually. Next, the student added a thrust script that produced 90% of the weight. The student apparently used this second script to ease the spuCraft to the proper speed more slowly. After several more attempts with the manual switch, the student was finally able to complete the Drop-Novice challenge, technically at least. He/she did not attempt to define a trigger.

Because Student R did not use or even attempt a trigger, he/she had a difficult time with the Drop-Intermediate challenge. Then suddenly, without any indication in the log that he/she had any understanding of the dynamics, the student had an input script appropriate for the third belt and triggers for all three belts. Given the performance on Drop-Novice, it seems plausible that Student R received the scripts from another student.