Gamification of modeling to promote expert-like behaviour and competence in applied physics among university students

Daniel Kastinen

February 2021

1 Introduction

Traditional university level education in introductory physics does not contain a significant amount of interactive engagement as it is mostly based on passive lectures. Research shows that interactive engagement promotes learning by a significant amount [Hake, 1998, Prince, 2004]. Traditional education is however not completely void of interactive engagement as laboratory work is a common part of most introductory physics courses. However, many studies show that simply implementing laboratory work in a course without the proper direction and iterative development does not bring that much benefit [Holmes and Wieman, 2018, Redish, 2004]. Using laboratory work specifically tailored for certain teaching purposes and learning outcomes can greatly benefit the learning of students [Redish, 2004]. With the swift development of computers both in computing power and hardware cost laboratory work is no longer restricted to physical experiments but can also consist of simulations. Recently a slew of e.g. online physics simulations have been developed to be used in interactive engagement oriented laboratory work [Zhou, 2018]. At the same time, it is being discovered what works and what does not when it comes to the use of physics simulations in education [Wieman et al., 2008]. Laboratory work such as discussed above usually entails group work. It has also been shown that including a certain amount of cooperative grouping and cooperative learning, i.e group-work, can be highly beneficial to learning outcomes if done correctly [Heller et al., 1992].

However, learning and knowledge acquisition is not all there is to becoming an expert in a subject. One must also adopt certain habits and behavioural patterns to become like an expert. This was studied in detail by Adams et al. [2006] where it was shown that students are not adopting expert-like attitudes and behaviour. Surprisingly, is that it was shown that students know very well what expert-like attitudes and behaviours are. They simply do not adopt them themselves. The reason for this can only be speculated about but a reasonable proposition is simply that students have not experienced situation's where expert-like attitudes and behaviours are required or justified. These situations usually comes to pass during doctorate studies and other research activities. When one realizes the need for the expert-like attitudes and behaviours in a situation they are much more easily adopted as their worth is internally affirmed. This hypotheses is supported by Kapon [2016] where a research-like situation was created in university level education and significant improvements in attitudes and behaviour was measured. In Kapon [2016], the research-like situation created a zone of proximal development, where the problem was slightly above the competence level of the student but guidance was minimal to emulate research. The lack of guidance and need for a deeper understanding to complete the task facilitated learning and behavioural change, different to traditional teaching. However, as is also the main criticism of such methods [Kirschner et al., 2006], the project was very time consuming with compared to the small amounts of assimilated knowledge.

The current generation of students have grown

up in a society where games are readily available and plentiful. Many students today can sit down in front of a new digital game and figure out the mechanics of that game world with any instruction in only a few minutes to hours. It is reasonable to assume that this ability comes from a constant exposure to digital games. However, the methodology of discovering the rules of a game world and physics research is practically the same but the latter being much more formalized. In fact, e.g. the additional enjoyment of a well designed game has made gamification a new and exiting area of research [Dicheva et al., 2015].

As was shown in Redish [2004], it is possible to create module like teaching activities that are tailored to achieve certain goals, that are simply slightly modified to the course where they are implemented. Following the design methodology of Redish [2004] and Heller and Hollabaugh [1992], we have created a small module of physics simulation laboratory work that specifically promotes both expert like behaviour by invoking the zone of proximal development, creating a situation that cannot be overcome without expert-like attitudes and behaviours, and teaches applied physics skills. As opposed to traditional research like elements in education, this module is accelerated by gamification in order to tap into the existing behaviour of exploration and discovery and by cooperative learning by performing the module in groups. Depending on the time available for the module, the physics to be simulated can be tailored to the needs of the course and education as a whole. Either the physics is built on existing knowledge by the students, shortening the time needed, or it is used to introduce practically a recently covered subject, requiring more time for the students to get familiar with the rules of the game. The equipment needed is a single computer per group, thus allowing this module to be highly flexible and used in almost any setting where a stronger presence of these learning goals are needed.

2 Education Component

2.1 Intended Learning Outcomes

The intended learning outcomes of this module are

I Improvement in expert-like attitude

II Functional knowledge of the physics to be simulated

III General physics modeling skills

IV Rudimentary data analysis skills

V Cooperation skills in research-like work

VI New perspective on what is "current physics"

2.2 Prerequisite Knowledge

Before attending this module the students should have at least

- Experienced some amount of scripting or programming
- Have had a broad overview of the simulated physics (there should be some unknown that need to be discovered)

2.3 Course Context

Based on the prerequisite knowledge and intended learning outcomes, it is recommended that this module take place somewhere between the end of the first year of Bachelor studies and prior to the last year of a Masters studies, i.e. somewhere between year 1-4 of University studies. The breath of possible courses is due to highly variable difficulty of the laboratory work. If more components of the physics to be modeled by the students are unknown, it will become significantly harder to discovery the correct model, rather then e.g. a single unknown component. For example, if the module is placed in a course at the end of the first year of Bachelor studies and we can assume this physics courses contain knowledge on Newtonian gravity. Then a good

compromise to make is to make the physics inside the game Newtonian gravity, give the students the masses and positions of the perturbing object(s) in the game, but have the gravitational constant be different than in real life. It now quickly becomes apparent to the students that the shape of implemented models are correct but the scaling is wrong and they need to analyse the games data output to discover the new coefficient in order to beat the level. This implementation leans more towards learning outcomes (I) and (IV-VI) than (II-III).

Depending on the extent of the module, anything between 1-4 ECTS is recommended for the laboratory work.

As was mentioned, the physics can be adapted to the course in question. We will below provide some initial suggestions:

- Electromagnetism course:
  Objects have positive and negative charge, constant magnetic field perpendicular to the screen, Lorenz force governs the player, possible challenge is to find the magnetic field strength and charges of the objects in the players path / the charge of the player itself.
- Introductory mechanics course:
  The player experiences friction with the game surface where the friction coefficient is unknown, the kinetic friction coefficient might also change depending on region in the level and there might be a constant force to simulate that of a tilted plane.
- Introductory physics course:
  Physics inside the game is Newtonian gravity, position and masses of perturbing objects are known but gravitational constant is different. Alternatively, the constant is known but the level is designed in such a way that Hohmann transfer orbits are needed to reach the goal.
- Quantum mechanics course:
  An obstacle in the game can be "quantum tunneled" trough (lets propose a high probability, say ¿50%). Once the effect is discovered the students need to analytically solve for the energy needed to tunnel trough with high enough probability for it to be profitable due to limited attempts in the game and figure out if they can reach this energy in the game.

Of course the above are only a fraction of the possible implementations available, it is up to the course examiner to decide on the most appropriate game content that matches with the course contents and intended learning outcomes.

2.4 Material

All the needed material for the game is available openly at this Github repository: `https://github.com/danielk333/TSDR`. This code should be forked and tailored to the course in question. The tailoring requires basic knowledge of Python 3 programming. Additionally, there is game instructions in the repository but ideally a specific laboratory instruction document should be constructed for the specific course depending on how the module is implemented.

2.5 Teaching and Learning Plan

The module itself can be divided into three segments:

1. Introductory session:
   Possible pre-test, introduction to the exercise, division into groups, hand out of information, checking that everyone can run the game and understands the task, initial questions from the students.
2. Laboratory work sessions:
   At least first session supervised, supervision on reaming sessions depend on the course context. Students play the game, model the physics and perform the necessary data analysis. Many questions are predicted to arise not in the very beginning of exploration but in the middle, where the goals have become clear (e.g. find the gravitational constant) but the path there is not (fit the implemented model with the force

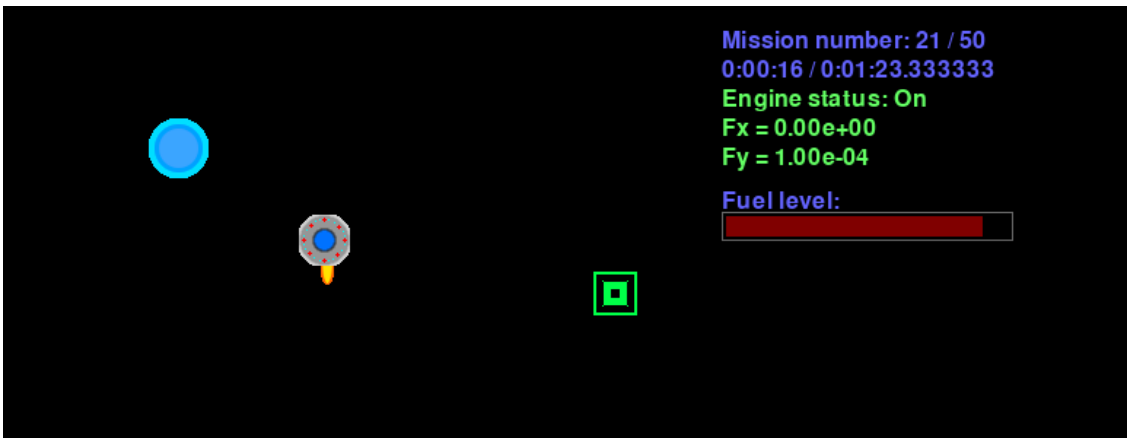

Figure 1: A frame of the game experiment mode in action.

data output from the game using e.g. lest squares minimization with the gravitational constant as a free parameter). Each sessions should end with a class discussion on the thoughts regarding what the goals are to beat the game and how to reach them, and the relationship between the work performed by the students and real life research and work. All groups should keep detailed logbooks on their thinking and work: what experiments are they running, what information do they look for, what problems do they experience, what are their solutions, ect. This together with valid solution code is the examining moment for the module.

3. Solution reviews:
   All group solutions to both the force model implemented and the thrust program implemented to reach the goal are presented by the group in front of the class. The solution is visualized by prepared software that only the examiner has and the solutions are briefly discussed in the class. Possible post-test.

2.6 Game

The game is about exploring an new realm with unknown physics using an autonomous probe, the goal being to move on to the next domain. The player has the possibility to either launch an automated probe with a specific thruster program into the new realm, generating a logfile of output data as measured by the probe, or to simulate the trajectory of the next probe using a model of the physics in this new realm. There is only a limited amount of launch attempts so one cannot be wasteful and just brute force the thruster program to find the solution. Much like spacecraft today, simulation prior to excursion is extremely useful when one can only afford to try something once or twice. Modeling the physics to match what happend in the first probe launch and then writing the thruster program is playing the game.

To test the game in the the current state of the base repository

1. Clone the repository[1]
2. Run `pip install -r requirement.txt`
3. Run `python run.py` to display the initial help needed to play.
4. The save game data is in the `game/data/save_data.npy`, to restart the game: delete this file.

To install on student computers

1. Clone the repository[1]
2. Run `pip install -r requirement.txt`

[1] `https://github.com/danielk333/TSDR`

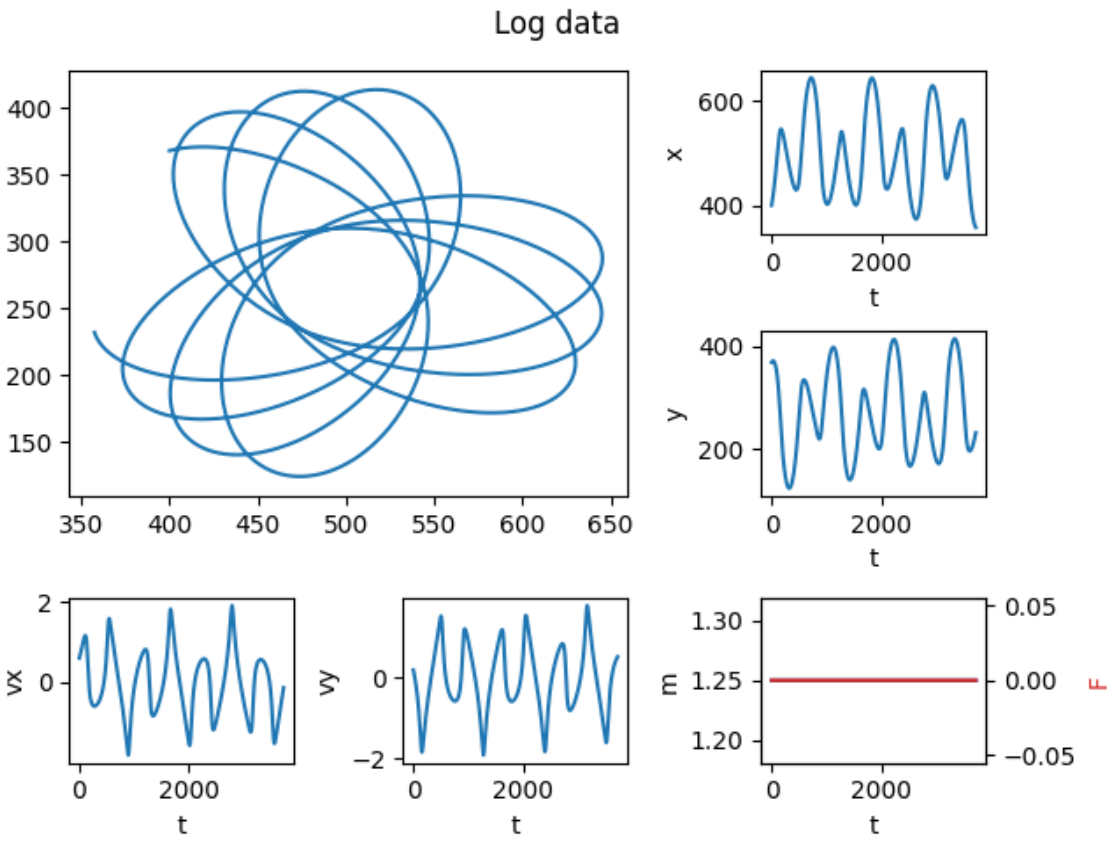

Figure 2: The quick-look plot of the log output by the game of a experiment using no thruster. The log is CSV formatted and easily loaded into any analysis software of choice.

3. run `python compile.py "path/to/game"`. The path should NOT be inside the repository.

4. Delete the repository.

5. Run the game with `python run.py [args]` from the `path/to/game` directory. All data output from the game will now be stored in this directory.

After the above steps, there should be a compiled game package in the given path, together with a `run.py` where the student input is stored. When this file is run with `python run.py` help text is displayed with available game commands. When the game is setup in an actual classroom, there should also be a prepared document with instructions in this folder specific for the course in question together with other possible needed information, e.g. a `numpy` cheat-sheet.

When deployed in a course, an additional layer of encryption should probably be put on the game configuration and the pass-codes file so that the game decrypts the files to read the data from them. This prevents to students from "cheating". This is currently not implemented by default so the files are visible in plain-text.

The game has 3 major modes:

- exp [realm number]: Experiment mode, launches an experiment into the new realm. The probe records all data it can, including its position, velocity, mass and thrust generated. The goal is to reach the next portal, otherwise the probe will be stranded and lost, although we always get the logfiles back.

- log [log number]: Log quick-look mode, plots the logfile from an experiment to view. This data will need to be more carefully analyzed but a first look is always good to have.

- sim [realm number]: Perform a simulation of an experiment. The simulation uses the model given by the player to predict the trajectory of the probe. If the physics model is correct, this simulation can be used to figure out how to create the automated thruster program that will get us to the goal!

- bank [passcode]: Redeem a passcode to receive more attempted experiments, i.e. more probe launches.

The main reason for the "bank" mode is the possible permanent loss state of the game. Some students might simply try to spam solutions and then not have more attempts to complete the exercise. The limitation on probe launch attempts is needed as it forces the students to model the physics so the next launch is known to succeed. If they need more chances, a system to earn passcodes can be put in place. One possible method is to have a set of analytic problems related to the tasks in the game, e.g. curve fitting, derivation of Hohmann transfer orbits, problems with the Lorenz force, ect. Handing in solutions to these problems earn them passcodes.

A typical game round might be

1. Clear the thruster control system so that it is completely turned off.

2. Launch an experiment with `python run.py exp`.

3. Watch the outcome, get familiar with the environment.

4. Plot the log with `python run.py log`, look at the forces that affect the probe.

5. Load the logfile into some analysis program written by the player, create an analytic model of the force, write it in the `def force(...):` function in `run.py`.

6. Run a simulation with the new force model using `python run.py sim`. If the trajectory looks like the logfile, you have discovered and modeled the physics!

7. Write a parameter controlled thruster program, run a simulation again using `python run.py sim`, but this time tune the parameters using the TK interface.

8. Find a thruster program that gets to the goal, put those parameters into the `get_control_params` function.

9. Launch a new experiment with `python run.py exp` and cross your fingers! Hopefully you win!

As of writing, the `run.py` file that students initially are faced with is listed in Figure 3. An example screenshot of the game running is illustrated in Figure 1. An example of the output from the games log quick-look function is illustrated in Figure 2 and finally, a screenshot of the simulation game mode is shown in Figure 4. Here the defined parameters of the thurster algorithm is available for quick editing to find the correct parameters needed to win the game, given that the physics have been correctly modeled.

2.7 Assessment

The assessment of the students will employ mostly formative authentic assessment [Lombardi, 2008, Lyon and Teutschbein, 2011] on the form of

- Small written summary (less then 1 page) of the modeling implementation and thruster control (summative traditional assessment)
- Continues logbook kept by groups on methodology of performing the laboratory work, from start to finish (formative authentic assessment)
- End-of-session summary on ideas and progress from each group, short class discussion (formative authentic assessment)

3 Evaluation

3.1 Pre- and post-test

As all intended learning outcomes besides (I) should be tested by the overall course examination, only the change in expert-like attitudes of the student remain to be tested. For this pourpuse we propose to use CLASS version 3 [Adams et al., 2006] as this was also used to initially determine the expert like attitudes of students. Version 3 has six item that are not scored as they were not productive in their current form, these questions should ideally be changed to better sample the intended learning outcomes of the module that are not already covered by the overall course testing. An example of this might be the change in viewpoint on "current physics" or the change in attitude towards applied physics and modeling.

3.2 Interviews

To facilitate an iterative approach to module development according to Redish [Ch.6 2004] a small sample of interviews with students should be conducted after completion of each course.

```
import sys
import numpy as np

from game import run


def get_control_params():
    '''These are the parameters that control the "thrust" algorithm.
       They will appear as float variables in the simulation.'''
    params = dict(
        t_start = 1e3,
        force_y = 0,
        force_x = 0,
    )
    return params


def thrust(pos, vel, t, m_wet, **params):
    '''Thruster controller: takes in ship position, velocity, time and
       the current fuel mass.
    Returns the vector force that the thrust should generate for that
       time.

    In the keyword arguments "params" are the parameters to control the
        algorithm.
    When running the actual experiment the parameters used will be
       fetched from "get_control_params".
    When simulating a mission, the parameters can be changed on the fly
        since it is a simulation.
    '''
    if t > params['t_start']:
        return np.array([params['force_x'],params['force_y']])
    else:
        return np.array([0.0,0.0])


def force(pos, vel, t, m):
    '''Model of the physics to be used in the simulation, a function of
        position, velocity, time and mass of the object experiencing
       the force.'''
    return np.array([0.0,0.0])

if __name__=='__main__':
    run(get_control_params, thrust, force, sys.argv)
```

Figure 3: The entry point `run.py` for the game. This is where students enter their thruster algorithm and model the force in the game.

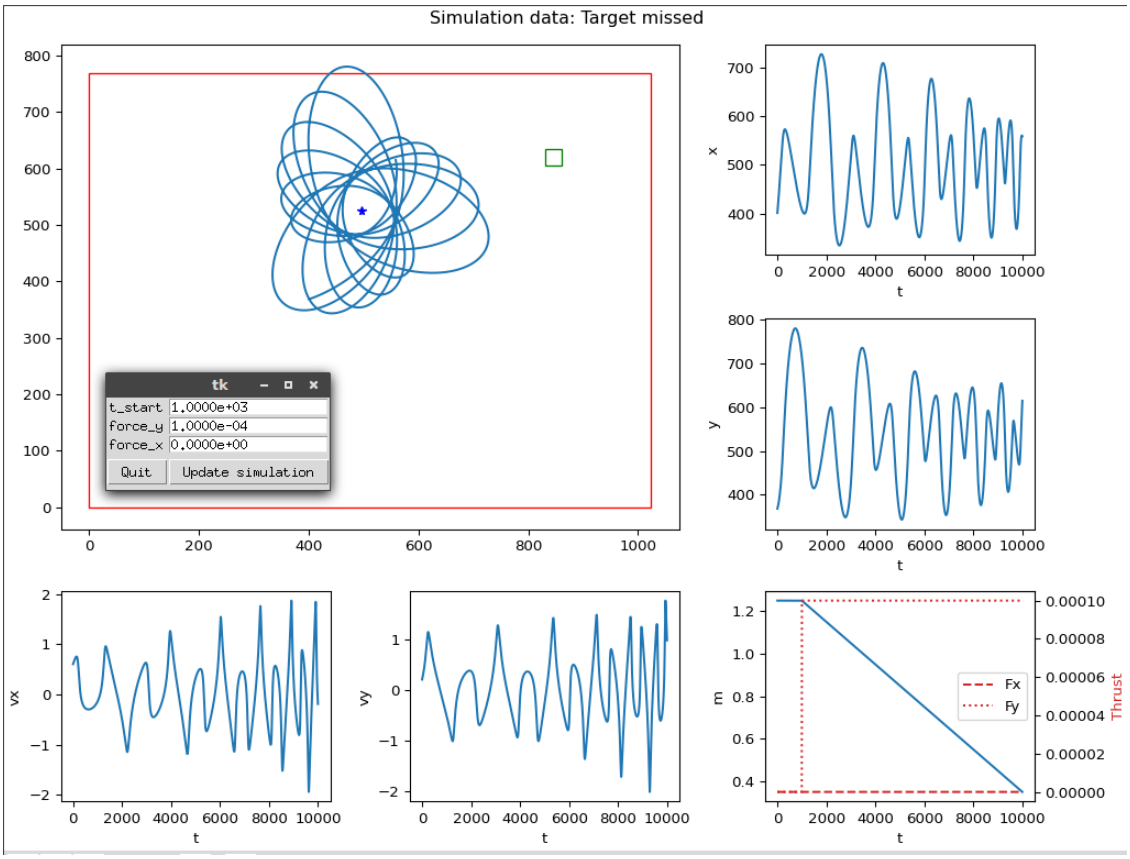

Figure 4: A screenshot of the simulation component of the game. This uses a slightly correct force model and the standard thruster program as also described in 3. All parameters listed are available trough the input-panel so that fine tuning of the algorithm parameters in preparation for the actual experiment probe launch.

References

Wendy K Adams, Katherine K Perkins, Noah S Podolefsky, Michael Dubson, Noah D Finkelstein, and Carl E Wieman. New instrument for measuring student beliefs about physics and learning physics: The colorado learning attitudes about science survey. *Physical review special topics-physics education research*, 2(1): 010101, 2006.

Darina Dicheva, Christo Dichev, Gennady Agre, and Galia Angelova. Gamification in education: A systematic mapping study. *Journal of Educational Technology & Society*, 18(3):75–88, 2015.

Richard R Hake. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American journal of Physics*, 66(1):64–74, 1998.

Patricia Heller and Mark Hollabaugh. Teaching problem solving through cooperative grouping. part 2: Designing problems and structuring groups. *American journal of Physics*, 60 (7):637–644, 1992.

Patricia Heller, Ronald Keith, and Scott Anderson. Teaching problem solving through cooperative grouping. part 1: Group versus individual problem solving. *American journal of physics*, 60(7):627–636, 1992.

Natasha G Holmes and Carl E Wieman. Introductory physics labs: We can do better. *Physics Today*, 71:1–38, 2018.

Shulamit Kapon. Doing research in school: Physics inquiry in the zone of proximal development. *Journal of Research in Science Teaching*, 53(8):1172–1197, 2016.

Paul A Kirschner, John Sweller, and Richard E Clark. Why minimal guidance during instruction does not work: An analysis of the failure of constructuvist, discovery, problem-based, experimental, and inquiry-based teaching. *Educational Psychologist*, 42(2), 2006.

Marilyn M Lombardi. Making the grade: The role of assessment in authentic learning. *EDUCAUSE Learning Initiative*, pages 1–16, 2008.

Steve W Lyon and Claudia Teutschbein. Problem-based learning and assessment in hydrology courses: Can non-traditional assessment better reflect intended learning outcomes? *Journal of Natural Resources and Life Sciences Education*, 40(1):199–205, 2011.

Michael Prince. Does active learning work? a review of the research. *Journal of engineering education*, 93(3):223–231, 2004.

Edward F Redish. Teaching physics with the physics suite, 2004.

Carl E Wieman, Katherine K Perkins, and Wendy K Adams. Oersted medal lecture 2007: Interactive simulations for teaching physics: What works, what doesn't, and why, 2008.

Zheng Zhou. Websites for physics demonstrations and computer simulations: A non-educational quality evaluation. *ProQuest LLC*, 2018.