CLIENT #1: ENGINEERING EDUCATORS

Many students, early in their engineering education at university, have a lot of difficulty understanding how to solve **real-world situations**. Solving these problems requires several steps. First, an **idealized model** is created for the real-world problem. The idealized model is then converted into a **free body diagram** showing the forces involved. Finally, a **series of equations** are used to calculate the forces shown in the **free body diagram**. While the process sounds simple, it requires students make a number of assumptions, reject even more assumptions and then choose the models and diagrams that best fit their assumptions. Learning when and how to perform these tasks is difficult as there are no clear rules and instead depend on heuristics and intuition.

Eventually, the goal is to have a complete tutoring system helping students gain the intuition needed for this to work. As a first step, they are looking for a system that grades students on selecting the assumptions needed to develop an idealized model from a real-world problem.

- The program should allow instructors to specify 1 or more real-world situations. Your program will need to display this real-world situation, so each situation will include an image file for your program to use. As the system grows, we expect the faculty will create more real-world situations for use in the program. Your program will to read in a text-based file to identify the current list of situations. This file should use a simple, structured format to make it easy for the Engineering faculty to add and modify the list of usable real-world situation.

- Each real-world situation will be associated with 1 or more idealized models. Every idealized models will be specific to a real-world situation; you can assume no overlaps exist. Each idealized model includes an image file that your program will need to display when asking about the assumptions upon which this model relies. Eventually, models will also have difficulty ratings, but it is unlikely those will exist in the next 6 weeks. Students are currently developing the idealized models for this program, so programs will to define a process with which the idealized models associated with each real-world situation are specified. While the listing will be updated and increased, you should assume the changes will occur between runs of your program.

- Each idealized model will be associated with 1 or more assumptions. Every assumption is simply a line of text. Some of the assumptions are needed (correct answers), some are not needed (incorrect answers), and some are complicating factors (not incorrect, but also not helpful). Assumptions are specific to their idealized model. As with everything else, it is expected that these lines of text will change with time. As with everything else, a simple, structured text-based format should be defined to hold these assumptions and read during program initialization. Because changes will occur, the format used to store assumption must include a way to specify if the assumption is needed, not needed, or are complicating factor.

- Assumptions that are not needed and assumptions that are complicating factors will also need to specify 1 or more reasons why they are not needed/complicating factors. Only one of these reasons will be valid. These will also change with time and so need some simple way to be specified.

- When the program begins, it should select a problem to have the student solve. For now, problem selection should be random. The eventual goal is to use idealized model's difficulty ratings to select problems. The program should show the real-world problem image, the idealized model image, and the list of assumptions. Users would then need to select the assumptions needed for this idealized model.

- After students select and submit the assumptions they think are needed, the correct result should be shown. Students should receive points for each correct choice and lose points for each incorrect choice. The number of positive and negative points will change with time and so the program will need to define a simple way for the instructor to specify this.

- For each selected assumption that not needed or was a complicating factor, the user should be given the option to earn some of the points they lost by selecting them. For now, display the reasons associated that assumption and letting them choose which was correct. The number of points earned by selecting this correct reason will change with time and so a simple way for the instructor to specify this is needed.