# Appraisal

## Introduction

I have created a program that can be used by students and teachers at Merchant Taylors’ School in order to learn or teach the ‘Pulley systems on an Incline’ part of the Edexcel M1 Mathematics Specification. The program is split into two parts. In the first part, the user can create their own ‘Pulley systems on an Incline’ M1 model before seeing it animate based on what would happen to the model if it was left in the user defined configuration. In the second part, the user is tested on randomly generated ‘Pulley systems on an Incline’ M1 models to test their knowledge on the subject.

Throughout the process, I was in regular communication with Mr.Slator, Head of Mathematics at Merchant Taylors' School. He helped me to shape the system to the needs of the Mathematics department and it was he who I presented the final system to. He helped analyse whether the objectives had been fulfilled and gave me the final feedback on the system. His feedback will be included later in this appraisal.

## Fulfilment of Objectives

The Objectives were:

**To make the program consist of two parts, one which allows a user to create their own model of a pulley system on an incline and then see it animate, and another which questions the user on a randomly created model.**

This objective has been fully achieved. The fact that this objective has been completed means that the program provides an interactive learning and teaching environment.

**To make both parts of the program have the functionality to animate the model to show what would happen if the model was left in its configuration.**

Both parts of the program have animation functionality, albeit the causes of the animation are different. This functionality is very important to the program as both me and Mr.Slator think it will keep students engaged when using the program.

**To make the program correctly calculate the direction of motion (if any) of a model.**

This functionality was tested in Test 39-41. The tests all passed, and so the objective has been fully achieved. Achieving this objective contributes to making an accurate model.

**To make the program correctly calculate the initial accleration (if any) of a model.**

This funnctionality was tested in Test 42. The test passed, and so the objective has been fully achieved. Achieving this objective contributes to making an accurate model.

**To make the program correctly calculate any extra displacement that a Ball undertakes after passing the equilibrium position due to it’s own momentum.**

This funnctionality was tested in Test 43. The test passed, and so the objective has been fully achieved. Achieving this objective contributes to making an accurate model.

**To make the program such that it is usable as a stand-alone executable without the need for any other files.**

This was a very important request by Mr.Slator. I have achieved this objective fully by embedding any resources I use such as Image and Text files into the executable.

**To have 3 questions per model in the part of the program that questions the user on a randomly created model.**

This was again specified by Mr.Slator. This was an easy objective to achieve but that being said it has been achieved.

**To make the part where the user is questioned on the model only animate if the user has got all the questions correct.**

This functionality was tested in Tests 45-46. The tests passed, and so the objective has been fully achieved. This encourages the user to learn from their mistakes and get better at the topic so that they can their models animate.

**To have the program explain why a model remained stationary if this has occurred**

This fucntionality was tested in Test 41. The test passed, and so the objective has been fully achieved. This not only helps the user learn what causes certain configuration to be in equilibrium, but also tells the user why an animation hasn’t occurred so they don’t assume the program is broken.

**To make sure that any animated model is within the bounds of the M1 specification**

This was by far the hardest objcetive to achieve. The mathematical rigour needed to validate all inputs or randomly generated values such that only M1 problems are displayed was very difficult. I needed a large number of validation checks. These validation checks were tested in Tests 9-38, and fortunately all of them passed. To this end, this objective has been fully achieved.

**To display error messages within 2 seconds**

I didn’t test this functionality in my system testing as I felt that it would be better tested by the User. Mr.Slator used the program for about 30 minutes after I first gave it to him and he confirmed that error messages were displayed seemingly instantaneously. Therefore, this objective has been fully achieved.

**To make items displayed in the program scale with screen size**

My tests were done on a 21 inch 1080p monitor. Mr.Slator then used the program on a 52 inch 720p smartboard. All items within the program scaled to fit the same proportions of the screen in both cases. I realise that this isn’t technically sufficient evidence to fully justify that the objective has been achieved, as I would need to test the program on infinitely many displays to do this. However, given my knowledge of the code and this evidence, I would say this objective has been achieved.

**To make the program fullscreen**

This objective was achieved, and it wasn’t very hard to achieve it!

**To design the interface such that a user can understand and operate it with the aid of a user guide within 5 minutes**

When I first gave Mr.Slator the finished program, I timed how long it took him to understand the interface and operate the program with the aid of the User guide. It took him just over a minute, and so this objective has been fully achieved. The interface is so simple that I believe someone would be able to understand and operate this program within 5 minutes without the use of a user guide.

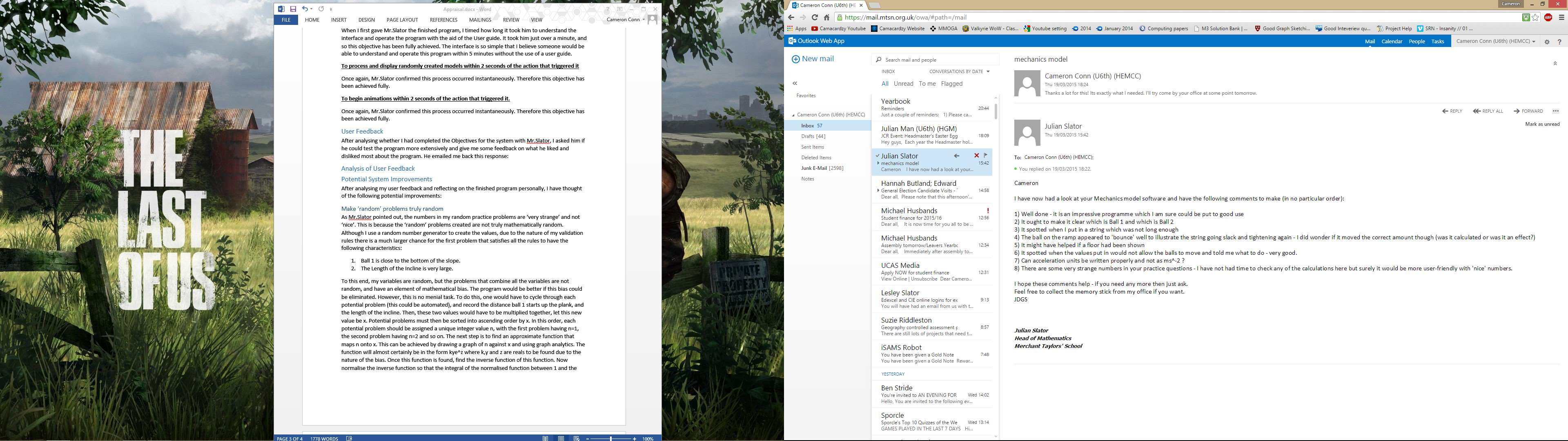
**To process and display randomly created models within 2 seconds of the action that triggered it**

Once again, Mr.Slator confirmed this process occurred instantaneously. Therefore this objective has been achieved fully.

**To begin animations within 2 seconds of the action that triggered it.**

Once again, Mr.Slator confirmed this process occurred instantaneously. Therefore this objective has been achieved fully.

## User Feedback

After analysing whether I had completed the Objectives for the system with Mr.Slator, I asked him if he could test the program more extensively and give me some final feedback on what he liked and disliked most about the program. He emailed me back this response:

## Analysis of User Feedback

I will now analyse each comment one by one:

1. Not much to say here other than that I am glad Mr.Slator liked the program!
2. I’m afraid to say that I got so lost in the intricacies of the Maths and the code that I didn’t even consider making it clear which was Ball 1 and which was Ball 2 within the interface. I checked the User Manual I created, and luckily it makes it clear which is which. However something as basic as this should be easily understood from within the program. Considering how easy this would be to correct, I will definitely be adressing this in my Potential System Improvements.
3. This is an example of the program limiting problems to ones of M1 standard. This was already an objective of the system that was passed due to the personal testing I completed, however the fact that Mr.Slator was pleased with the functionality in his own use means that the objective was very successfully completed.
4. After using the program again myself and checking the Mathematics and some questions from the M1 textbook, I can confirm that the bounce was the correct length.
5. Again another fair point that I didn’t think of. Once again I will be considering action to correct this in my Potential System Improvements.
6. This functionality was a fufilled objective, so no further comment is needed other than that it was important to get this functionality correct as Mr.Slator had originally specified it existed.
7. I agree with Mr.Slator that Ms^-2 doesn’t look particularly good. However, I don’t think I am going to change it. This is because Metres per Second Squared takes up to much space and would effect the rest of the UI too much.
8. Essentially what Mr.Slator means is that the Random Problems generated include either very long Plank Lengths or a very small Distance that Ball 1 starts up the Plank, or both. This makes a lot of the problems seem odd and similar. This is a problem that I will fully explain and address in my Potential System Improvements.

## Potential System Improvements

After analysing my user feedback and reflecting on the finished program personally, I have thought of the following potential improvements:

### Make ‘random’ problems truly random

As Mr.Slator pointed out, the numbers in my random practice problems are ‘very strange’ and not ‘nice’. This is because the ‘random’ problems created are not truly mathematically random. Although I use a random number generator to create the values, due to the nature of my validation rules there is a much larger chance for the first problem that satisfies all the rules to have the following characteristics:

1. Ball 1 is close to the bottom of the slope.
2. The Length of the Incline is very large.

To this end, my variables are random, but the problems that combine all the variables are not random, and have an element of mathematical bias. The program would be better if this bias could be eliminated. However, this is no menial task. To do this, one would have to cycle through each potential problem (this could be automated), and record the distance ball 1 starts up the plank, and the length of the incline. Then, these two values would have to be multiplied together, let this new value be x. Potential problems must then be sorted into ascending order by x. In this order, each potential problem should be assigned a unique integer value n, with the first problem having n=1, the second problem having n=2 and so on. The next step is to find an approximate function that maps n onto x. This can be achieved by drawing a graph of n against x and using graph analytics. The function will almost certainly be in the form kye^z where k,y and z are reals to be found due to the nature of the bias. Once this function is found, find the inverse function of this function. Now normalise the inverse function so that the integral of the normalised function between 1 and the largest n value is 1. Call this new function g(n). Now for a given n, the probability weighting that should be applied to that problem to eliminate bias is g(n).

### UI Improvements

If I had more time to create the program, I would certainly make some UI Improvements to improve the user experience and make it easier to use the program:

#### Labelling Ball 1 and Ball 2

Currently, Ball 1 and Ball 2 are not labelled within the program. For the user to know which is which, another source such as the user guide is needed. The User guide clearly explains which is which, but I still feel that if the Balls were labelled the program would be even easier to use. This change could easily be achieved by using image manipulation software to edit the images of the Balls such that they have a number on them for labelling purposes.

#### Adding a Floor

As Mr.Slator suggested in his final User Feedback, having a floor would make it easier for the user to see when either of the Balls had hit the floor. This could be implemented very easily by including another horizontal TLine.

#### Changing how the direction of motion question is answered

Currently, this question is answered by typing in ‘up’ or ‘down’ into a textbox. While the program accepts any capital letter/non-capital letter representation of these words, I feel it would be easier if the question was answered a different way. I would probably use radio buttons to make the selection more intuitive.

### Sharing problems across the school network

While not something suggested by Mr.Slator, I have recently thought of functionality that would potentially make the program better. If students are all using the program at the same time, I think it would be helpful if it was possible for one student to share a problem they are working on with another student. For example, if a student is stuck on a problem, they would be able to send it to a friend to look at so that they can help them. There could even be functionality that allowed the problem to be broadcast to the whole class. This could be achieved be using the UDP or TCP protocols (depending on the scenario) to send the variable values.