Using Empirical Modelling to Demonstrate Linear Kinematics

0412667

Abstract

Engineering mathematics can be difficult to understand. Often a student's first step in solving a problem will be to visualise it by drawing a diagram with pen and paper.

As described by M Klawe (1995), this paper introduces how interactive electronic tools can be advantageous for learning. An Empirical Model created with Eden scripts is described, and its implementation explained.

The software demonstrates the mathematical topic of linear kinematics by visualising the scenario on which an exam-style question is based. The trajectory of an object is randomly chosen, and the user is asked to calculate parameters of its flight. The software allows visualisation of each scenario and some experimental interaction by controlling the simulated motion of a particle. The program shows output in both graphical and numerical forms, to illustrate how they are linked. For example, how the flight path of a projectile changes under varying forces of acceleration due to gravity.

The model will primarily be of benefit as an educational tool, to help learn linear kinematics. This particular branch of mathematics is part of the A-level mathematics syllabus, and could prove to be a valuable learning tool for this.

Secondly, the study will highlight how the use of Empirical Modelling techniques can aid in the construction of such computer programs as described in our study; how they are 'oriented towards aspects of computing practice for which formal methods offer limited support'. The example we use is particularly suited to EM since the central concept of dependency is tightly linked with that of mathematical equations. When the user modifies one parameter of a kinematic equation, others must be altered accordingly.

1 Introduction

This paper introduces and describes the difficulties with learning. It has been proposed by M. Klawe (1995) that interaction is an invaluable method for improving understanding. A key concept within the field of Empirical Modelling is the construction of models which allow flexible interaction. It will go on to argue why the methods available within the field of EM are ideal for this kind of task. As an aside it will be discussed as to how other central EM concepts, such as dependency, make the task of creating this model simpler than would be within other paradigms.

The model associated with this paper has been constructed using the EM tool Eden. It uses the DoNaLD extension for drawing graphs, and SCOUT for creating windows and buttons. Details of its construction are written about later on in this paper. The software is a tool for learning a branch of engineering mathematics – linear kinematics. Specifically, the model visualises the motion of a projectile, for which the position is being calculated using kinematics. The user is shown exam-style questions regarding a random scenario. They can interact with a simulation of the scenario to aid in understanding the situation. Various levels of assistance help the user come to a numerical answer. They can simulate the launch the projectile, and observe its flight both numerically and graphically.

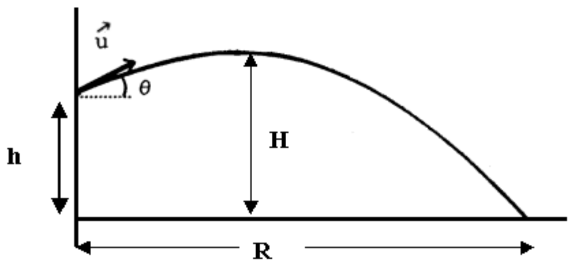


Figure 1 - Linear kinematics of a projectile moving under the force of gravity

Figure 1, shown above, is the general form of questions posed by this piece of software. A projectile is launched from a height h above the ground, at an angle theta from the horizontal, with an initial speed of u. The syllabus for A-Level mathematics requires the student to be able to calculate three pieces of information based on these initial parameters. These are:

- The maximum height obtained by the projectile (H meters)
- The horizontal distance the projectile travels before it hits the ground (R meters)
- The total time in flight between leaving the platform and reaching the ground (t seconds)

Air resistance is always neglected, and acceleration due to gravity is usually rounded to be 9.81ms^{-1}.

2 Related Work

2.1 Learning

It is fair to say that electronic devices are thoroughly employed for assistance with learning. Generic software packages such as Word and Excel are used for a multitude of tasks such as collecting information, performing calculations and writing reports. As any student will note, the use of PowerPoint slides during lectures and presentations is almost ubiquitous, as they allow simple presentation of information. In addition web browsers are used to access a plethora of learning material available over the internet.

It has been said that interaction affords understanding of the behaviour of such systems. Beynon (1997) comments “the correlation of experiences of different artefacts, and the acquisition of skills in their manipulation”. This means that the user better understands the system if they are able to continually modify the controls, and observe the subsequent system behaviour. EM software is built around this concept of allowing continual interaction. The user is able to directly modify the values of system variables and immediately observe the results of this interaction.

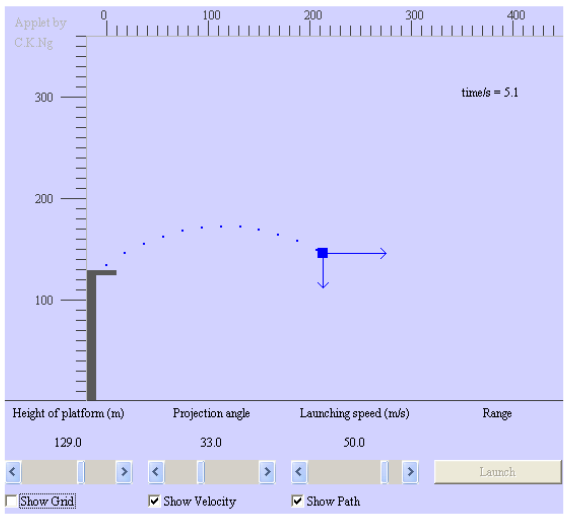
Another concept that this study hopes to exploit is that of repetition. It is widely accepted that repetition enhances the learning experience. It helps to re-enforce importance within the student’s mind. This model will continually generate new scenarios for the user to solve problems for, allowing them to repeat the type of calculation as much as they like. In a similar vein, it can also be beneficial to keep the user engaged by providing them with a variety of types of questions. Klawe and Philips (1995) propose that “[the] ability to transfer is enhanced by experiencing the learning in multiple modes and contexts”. I interpret this as meaning that a student can enhance the rate at which they learn by using a variety of tools and methods. Such as moving from working on paper, to working on a computer.

2.2 Related Software

2.2.1 Java applet

C. K. Ng has created a Java applet which attempts to illustrate linear kinematics[1]. As shown below in Figure 2, it displays a graph of a projectiles motion over time.

The user is able to choose parameters for the launching platform height, and the starting speed and angle of the projectile. When the ‘launch’ button is pressed, the object starts to move away from the platform. Its location over the simulation time is determined by the kinematics equations and continuously updated. Two arrows on the object indicate the magnitude of its horizontal and vertical velocity components.



time/s = 5.1

Height of platform (m) 129.0

Projection angle 33.0

Launching speed (m/s) 50.0

Figure 2 – Kinematics Java applet

When the projectile reaches the ground, animation in the applet stops and the GUI displays the time taken for the simulation, and the horizontal range away from the platform that the projectile has achieved.

2.2.2 Models

Many models from the EM projects archive[2] have been studied before the creation of the model in this paper. Key principles have been understood by viewing them. Notably, from the OXO model I have learnt and used techniques for drawing lines and shapes to the screen. I have also taken drawing techniques from the complex numbers learning tool, along with ways of taking and validating user input.

Finally, from Yung’s ‘Room Viewer’ the very central concept of dependency has been examined. In Yung’s model, the position of most of the objects is dependent on a single variable. This means that the model will retain its semantic integrity if certain parameters are altered. Moving the position of the corner of the room, for example, will cause the rest of the room to be redrawn in a different position. Similarly, within my kinematics model, the physics equations are defined as dependent on the values of their parameters. Furthermore the drawn projectile object is dependent on the value of the equations. This makes it very simple from a programming perspective to move the projectile throughout the time of the simulation, we simply change the time vari-

[1] Projectile Motion, http://www.ngsir.netfirms.com/englishhtm/ThrowABall.htm

[2] EM projects archive, http://www2.warwick.ac.uk/fac/sci/dcs/research/em/projects/

able and the dependencies trigger the visible object to be redrawn in the correct position.

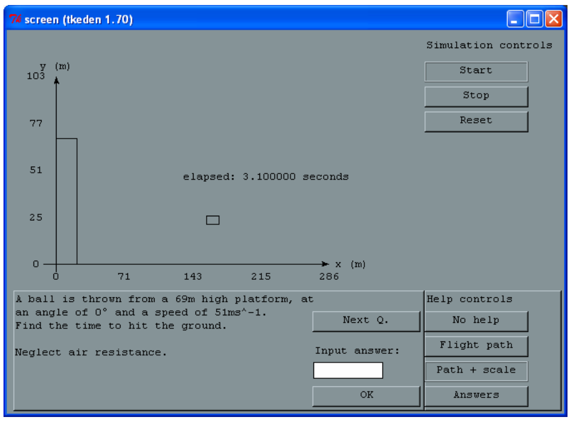
3 The Eden Model

3.1 Run instructions

The model was developed using the Windows version of tkeden 1.70[3]. It was only tested within this environment. It requires a minimum 800 by 600 screen resolution to view all controls and output.
The application is launched with the Eden script 'Run.e'. This in turn loads the other files necessary for all features of the model to work properly.

3.2 Usage

Upon loading the main Eden script '`Run.e`' the user is presented with the interface shown below in Figure 3. All interaction with the model is through this GUI, as opposed to using textual input via the Eden interpreter.



```
elapsed: 3.100000 seconds

A ball is thrown from a 69m high platform, at
an angle of 0° and a speed of 51ms^-1.
Find the time to hit the ground.

Neglect air resistance.
```

Figure 3 – Screenshot of GUI

Clicking the '`Next Q.`' button randomly generates a new exam question scenario, consisting of a platform height, angle of launch, and launching speed. The user is presented with a worded question based on this new scenario. They are able to type their numerical answer into the textbox. Clicking '`OK`' checks whether the answer is correct, and subsequently gives the user feedback.
In order to act as a learning tool, the model can provide assistance with solving the problem by means of a moving simulation of the scenario. Four levels of increasing assistance are available, controlled by four mutually exclusive (radio style) buttons. The options are; 'No help', 'Flight path', 'Path + scale', and 'Answers'. 'No help' allows no use of the simulation visualisation tool. With 'Flight path' selected the user can view the general motion of the projectile leaving the platform and stopping at the ground. 'Path + scale' adds values to the axes, roughly indicating the maximum height and landing position of the projectile. Finally, 'Answers' reveals accurate values for the scenario. With this selected, labels show the range of the projectile, maximum height achieved and time in flight. Providing a variety of levels of assistance ensure that the user is able to make some attempt at the answer, without making it too easy for them.

3.3 Internal Structure

Functionality for the program is split across six files in total. Five contain independent modules for various aspects of the model, and the final file contains definitions to link together the interfaces of the modules, allowing clear interaction between them. The modular architecture makes it easier to make changes to the model, or even re-use modules within other models. Most importantly it is much easier to understand.

3.3.1 Run

As mentioned already, '`Run.e`' loads the fives files containing all functionality for the model. It also links together the interfaces from modules which need to interchange parameters with each other. For example the DoNaLD graph plot is given a dependency of values produced by the question generator. This script also sets up some starting values for the GUI to display before the first question is generated.

3.3.2 Question Generator

'`question_generator.e`' is an Eden script which generates random scenarios as the basis of questions. Each new scenario consists of a platform height, angle of projection, and initial speed. All three of these variables are between 0 and a specified maximum value. This information is formed into a character string which contains an exam-style question.
Once the scenario has been created, the script also has the ability to find values for; the maximum height obtained, the range of the projectile before it hits the ground, and the total time in flight. The script can validate whether the user's answer to the question is correct. It actually accepts an input within some threshold of the actual answer to account for rounding errors.

3.3.3 Graph plot

Drawing of the 2D plot of the scenario is handed by a DoNaLD script called '`graphplot.d`'. The module has an interface which accepts the current posi-

[3] Eden software, http://www2.warwick.ac.uk/fac/sci/dcs/research/em/software/eden/

tion of the projectile as x and y co-ordinates, and a single value for the height of the platform. Both are in meters. It also accepts two values for the ranges of the scales to extend to. The script contains a variable which indicates what help level the model is at, the value of this determines what is drawn. At a basic level the script always shows the labelled axes and elapsed time. At the next help level, the object is drawn at the specified co-ordinates and the platform extends from the x axis up to this height. These are both redrawn every time their variables are modified. At the third help level a numerical scale is added to each of the axes. Values for this are calculated depending on the values of xmax and ymax. Finally, labels for answers to the simulation are shown if the highest level of help (four) is chosen.

3.3.4 Physics

Equations describing the motion of the projectile are contained within the Eden script '`physics.e`'. It defines the x and y position of the projectile by considering it as a point mass within the constraints of linear kinematics. It takes the inputs of initial velocity (u), angle of launch (theta), time (t), acceleration due to gravity (g), and platform height (h).

$$x = u\cos(\theta)t \quad y = h + u\sin(\theta)t - \frac{gt^2}{2}$$

It would be a simple matter to interchange this physics model for another, as long as the same interface is presented to the rest of the system. It would be simple to change acceleration due to gravity for example. Equations for calculating the x and y velocities at each time step are available, these could be used within an extension of the model.

3.3.5 GUI

The SCOUT interpreter is used to construct the GUI as shown in Figure 3. Four panes help to structure the visible content in a two-dimensional array. A graph for the simulation, along with controls to start, stop and reset it are at the top of the window. This is a viewport into which a DoNaLD script draws. Input and output for the current question are shown at the bottom of the window. Alongside this are radio buttons for the various levels of help. Apart from the graph, all visible content such as the window, borders, lines and labels are created with this script.

3.3.6 Button Events

Events which are triggered by clicking buttons on the GUI are handled by the script 'buttonevents.e'. The general form of the code handling one of these button events is shown in the extract below;

```
proc resetclick : resetButton_mouse_1 {
  if (resetButton_mouse_1[2]==4) {
    resetButton[17] = "sunken";
  }
  if (resetButton_mouse_1[2]==5) {
    resetButton[17] = "raised";
    t=0;
  }
}
```

This particular definition responds to the reset button being clicked. It is triggered by the redefinition of the variable '`resetButton_mouse_1`', which is handed by the SCOUT environment. For this definition, when the physical mouse is depressed, the button's style is changed to mimic it being pushed down, as within a standard GUI. Releasing the mouse button causes the button to pop back up, as the border is reset to its raised position. The real function of this particular button is to redefine the time variable to be 0.
Other buttons control starting and stopping of the clock which in effect moves the projectile during a simulation. The inbuilt clocking functionality increments a specified variable by 1, every time interval.

```
setedenclock(&t, 100);
removeedenclock(&t);
```

In the above example, setting the Eden clock causes the variable, named '`t`', to be incremented by 1 every 100 milliseconds. Calling the remove Eden clock function stops this event occurring.

3.4 Known Problems

Within the simulation display, the label for elapsed time shows six decimal places, many more than necessary. This is due to a problem with conversion of rounded numbers to strings. It would be possible to write a function which does this, but I believe it is not strictly necessary for correct output.

4 Analysis

4.1 Model Analysis

The system exhibits correct behaviour under all test cases it was exposed to during development. The state of the application determines the kind of input which the user is able to perform. In most cases this is done by 'greying out' buttons and disabling their sensitivity to mouse clicks. This means that the user cannot put it into an erroneous state. The GUI is simple and straightforward to use, though it does leave a little to be desired from an aesthetics point of view, the grey and black colour scheme is a bit plain.

As mentioned already, utilising the dependency relation within Eden leads to an elegant solution. Parameters which depend on each other are defined as a 'dependency'. When a variable on the right-hand side of an equation is updated, Eden automatically propagates the update to the left-hand side variable. Furthermore, the development of the application was aided by capability of the Eden environment to continually accept modifications to the model. This way I was able to experimentally develop, which can be a very quick way of testing settings.

4.2 Java Applet Comparison

In comparison to the cited Java applet, this implementation differs in a number of way. Firstly it provides additional functionality in learning kinematics by posing exam-style questions and validates the user's response to these. Furthermore, the user can choose varying levels of assistance to help them in solving the problems. A final small addition is that the Eden model provides the maximum height which the projectile achieves.
Although the source code for the Java applet has not been inspected, I suspect the architecture of this Eden model is much more elegant. Java requires explicit event handling to deal with variables being updated. Within Eden we don't need to concern ourselves with the order in which variables are recalculated.

4.3 System Value

This branch of mathematics is a topic within the syllabus for the A-level mathematics module M1 (Mechanics 1). It is also examined in parts of AS physics. As a teaching tool, the potential uses for this model are directly obvious. Students can test themselves by attempting to solve problems based on new random scenarios. Using the constructed model as a basis, it will be easily possible to extend its functionality to visualise specific problems given in text books, or as part of maths lessons.

5 Evaluation

5.1 Study Evaluation

I set out to construct an Eden model which would aid with learning kinematics. This model would pose randomly-generated exam-style questions, and provide a simulation of the scenario, allowing interaction to aid in understanding.
Overall, I am very pleased with this study and the model which has been constructed. Some papers have been discussed which consider electronic tools as effective learning aids. The system displays the advantages of using some Empirical Modelling principles of dependency, and learning by experimental interaction.
The model itself could be a useful aid for learning the kinematics module of the Mechanics 1 A-Level Mathematics syllabus.

5.2 Extensions

Further work on the model could extend its functionality. I suggest extensions to display motion in a third dimension, extending to present a further class of problems. I also think it would be beneficial to improve the user interface, to generally make it look more aesthetically pleasing, and so perhaps more attractive to use. I particularly recommend that the graphics of the simulation display should be improved. This extension would be simple to implement by altering just the graph plot DoNaLD definition file. An image could be loaded which would represent the projectile.

6 References

6.1 Papers

– "A classroom study: Electronic games engage children as researchers", M Klawe, E Phillips, CSCL '95 Conference Proceedings, (1995), cs.ubc.ca
– "The Electronic Classroom: A Handbook for Education in the Electronic Environment", Boschmann, Learned Information, (1996)
– "Empirical Modelling for Educational Technology", Beynon W. M., Technology. Proc. Cognitive Technology '97, (1997)

6.2 Eden models

cogRoe1999 (coordinate geometry)
complexGarner1999 (teaching complex numbers)
planimeterCare2005 (planimeter model)
jugsBeynon1999 (jugs)
roomYung1989 (room viewer)
oxoGardner1999 (O's and X's game)