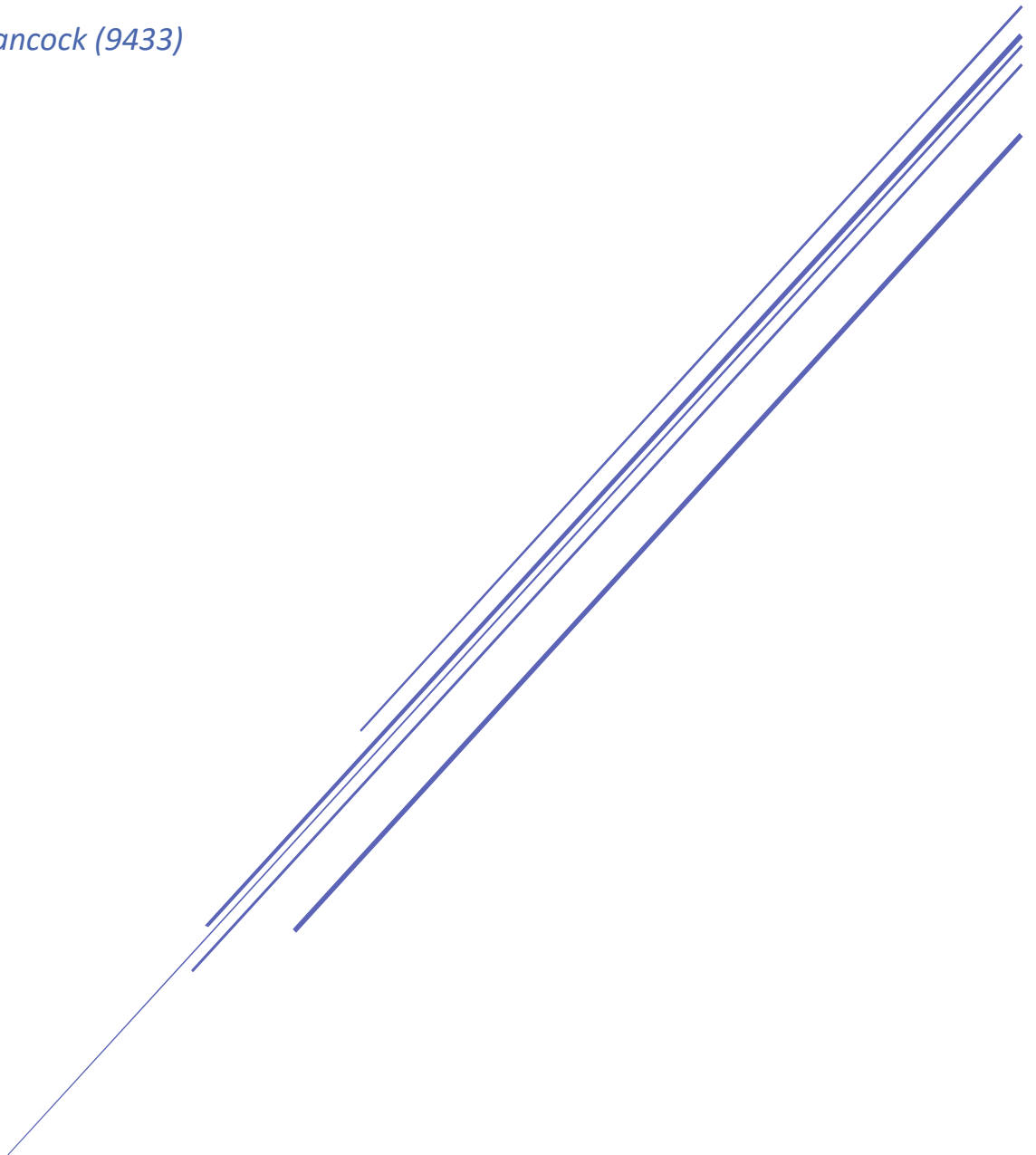


Gravitational Field Visualisation and Educational Aid

James Hancock (9433)



*Godalming College
Computing NEA*

Table of Contents

Contents

Table of Contents.....	1
Introduction	5
Investigative Methods	5
Questionnaires.....	6
For students	6
For teachers	6
Questionnaire Analysis	7
Discussion with a teacher	9
What utilities are in existence already?.....	10
Notable Simulations.....	14
Textbooks.....	16
Experiences in direct tuition	19
What are Energy and Potential Energy?	19
What is Gravitation and Gravitational Potential Energy?	20
Research Sources	23
Analysis	24
Previous System Questions.....	24
Current System Analysis	25
IPSO Diagrams.....	26
IPSO For a textbook based system.....	26
IPSO For a computerised question system	26
Document Specification	27
Data Flow Diagrams	28
Summary of Research and Analysis	30
Requirements.....	31
Simulation/Model Requirements	31
Learning Resource Requirements	32
Database requirements	33

Justification Of Requirements.....	34
Simulation/model Requirements.....	34
Learning resource requirements.....	34
Database requirements	35
Design.....	36
Form Design	36
Home Form	36
Form Navigation.....	36
Login Form	37
Account and results	38
Test Result Upload	38
UML Class Diagram For the simulation.....	39
Algorithms.....	40
Sum of gravitational Forces	40
Recursively Generating Field lines	40
Compression of the test mass list.....	41
External Libraries and modules.....	41
Data Flow Diagrams	42
Data flow with a student user.....	42
Data flow with a teacher user.....	42
Data Dictionaries.....	43
Students Table	43
Teacher Table.....	43
Classes Table	43
Results Table	43
Database Design.....	44
Entity Relationship Diagram.....	44
DDL string for database.....	44
SQL string for queries.....	44
Regex expressions for validating strings	46
Alpha Testing Strategy	48
Testing outline	48
Input Testing Strategy.....	48
input testing datasets	48
Input testing table.....	49

Flow of Control Testing Strategy.....	50
Flow control within simulation	50
Flow control within Account use	50
Process Testing Strategy	52
Storage Testing Strategy	57
Database creation	57
Inserting into the database	59
Beta Testing Strategy	61
Questionnaire For students	61
Questionnaire For teachers	62
Technical Solution	63
Home Form	63
Layout Creation	63
Navigation	67
Simulation	69
Login System	83
Register System.....	85
Account	86
Database	87
Results.....	96
Tests.....	98
Alpha Testing.....	99
Input Testing	99
Flow of Control Testing.....	100
Variable passing test for student user	100
Variable passing test for Teacher user.....	100
Process Testing	100
Calculation of Gravitational Potential.....	100
Calculation Of Gravitational Force	100
Summation of Forces	101
Other Algorithmic Functions.....	101
Visual Testing	101
Storage Testing	102
Location Testing for student registering.....	102
Location Testing for teacher registering.....	103

Requirements Testing	104
Simulation/Model Requirements	104
Learning Resource Requirements	105
Database Requirements.....	105
Beta Testing	107
Results from Beta Testing Questionnaires.....	107
User feedback	108
Evaluation	109
Appendix	111
Testing Video	111
Input testing.....	111
Process Testing	114
Storage Testing	119

Introduction

The aim of this project is to create a visualisation and classroom aid for teaching or independently learning about gravitational potential. With set questions for testing a students comprehension, the results from such questions would be accessible to a teacher through a database.

The proposed audience for this project is Advanced Level Physics Students aged 16 to 18, or similarly able students abroad. However these foreign students would have to understand English as the resource will not be multi-lingual.

While this product will be useable as a learning and educational tool it should also be accurate enough for use in scientific fields and should be of aid in an investigation. This will require the inclusion of tools for examining and potentially modifying the program.

Investigative Methods

I will determine the specification and requirements for the programme mainly through questionnaires, which will allow me to receive responses from large numbers of people. The advantages of a large data set are the improved accuracy of my results and they will be representative of more people as a result. The recording of all responses reduces the chance of answers being lost or misinterpreted. However, there is a chance that some people will not return the surveys, or they will be lost.

The requirements for an educational end user will be discovered through face to face communication with physics teachers and students. The advantage of this method is it allows more questions to be asked which means that a greater understanding of the users' needs can be gained, lowering the chance of a misunderstanding skewing the needs. While time consuming this is necessary to create a comprehensive list of the end user's requirements.

The requirements for the system will be developed through the observation of other open-source utilities, which will enable me to find the strengths and shortcomings of each resource. This process will allow me to ensure that my system is effective at solving the issues discovered within other resources. While time consuming to conduct this process is not only beneficial to overall effectiveness it will also provide inspiration for the design of my system.

I will also inspect existing educational tools such as textbooks and websites, while also gaining experience in actual instructed lessons for comparison. This will show me the details that are essential for students to understand this concept, as well as the possible methods of tuition that should be implemented into the system so it is reflective of real lessons.

Questionnaires

The following questionnaires will be given to potential users of the system to determine the product's necessity. This will allow me to narrow the field of people for which a face to face discussion would be more appropriate. They will be distributed via email and social media for ease.

FOR STUDENTS

1. What is the highest level of physics you have studied?

Below GCSE *GCSE* *AS physics* *A2 physics* *Degree*

Means that the participants can be categorized allowing me to gain an understanding of my target audience. And the results from people who have not studied the topics in question can be ignored

2. What would you rate your understanding of work done and potential energy?

Poor *Below Average* *Average* *Above Average* *Strong*

Shows the need of the product and how much background information the resource requires.

3. Have you ever struggled with understanding gravitational potential?

Yes *No*

Asks again if the resource would be required

4. Would a clear visualisation of gravitational potential assist in your understanding?

Yes *No*

FOR TEACHERS

1. What level of physics do you teach?

Below GCSE *GCSE* *AS physics* *A2 physics* *Degree*

2. What would you rate the understanding of work done and potential energy in your classes?

Poor *Below Average* *Average* *Above Average* *Strong*

3. Have you ever struggled with getting students to understand gravitational potential?

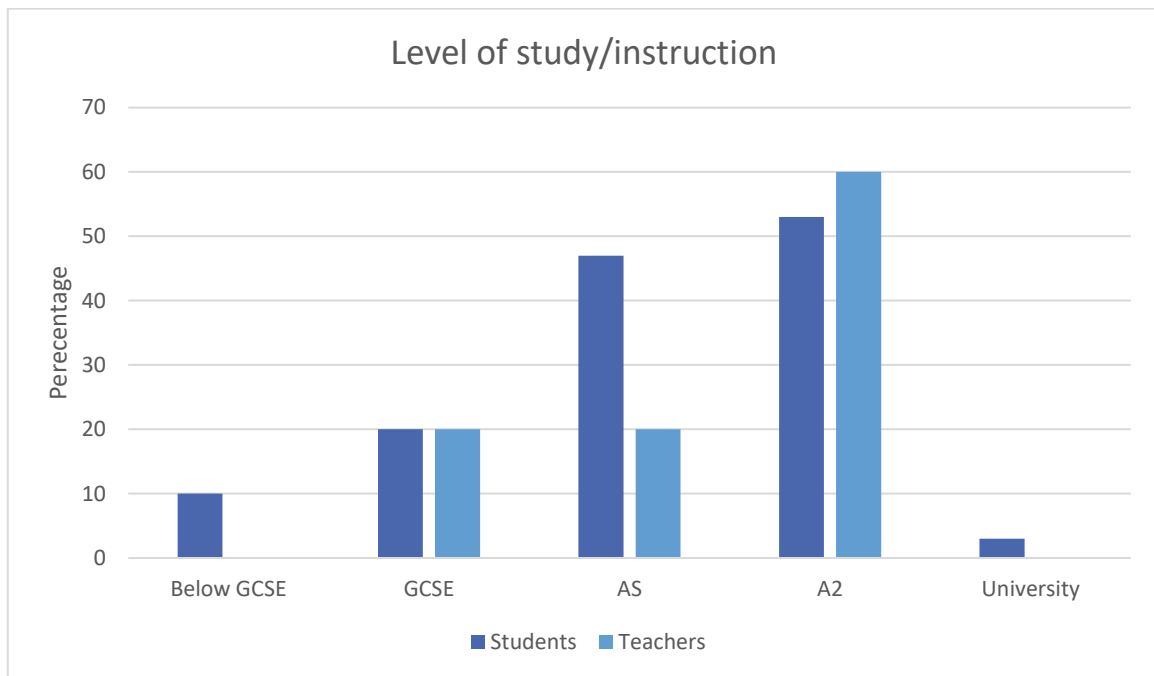
Yes *No*

4. Would a clear visualisation of gravitational potential assist in your teaching?

Yes *No*

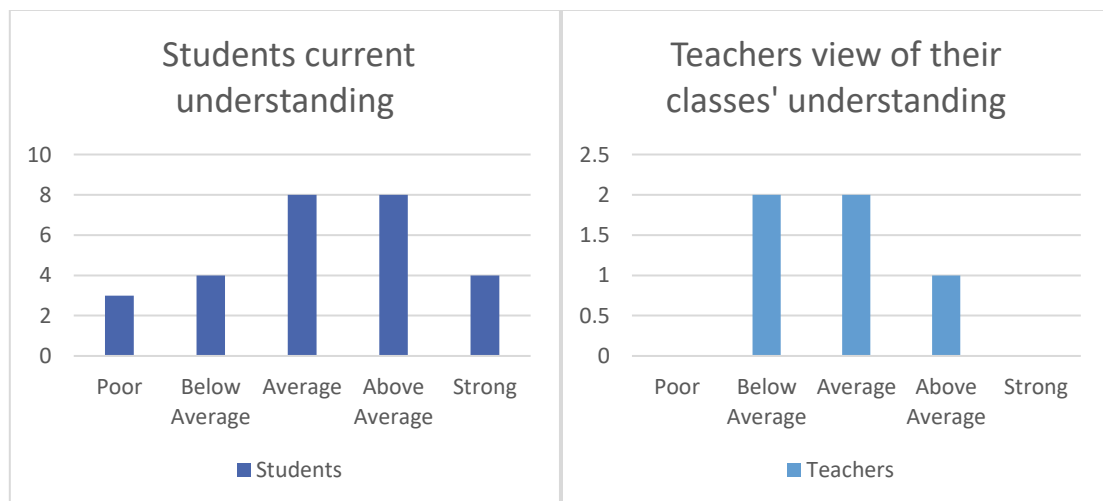
Questionnaire Analysis

A sample size of 30 physics students and 5 physics teachers was used in the questionnaire.



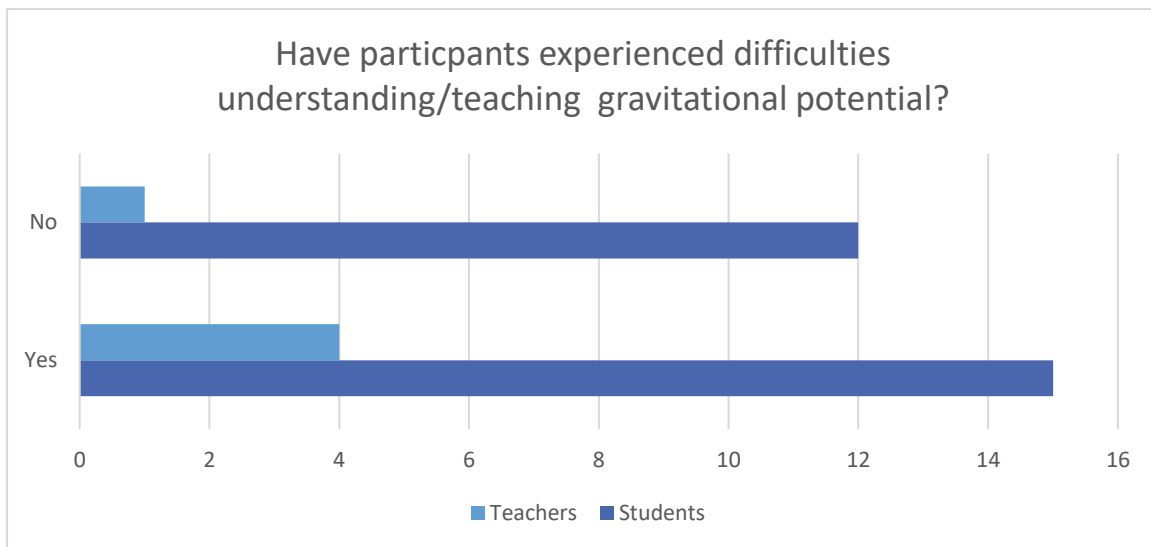
While not immediately beneficial, this enables the removal of irrelevant results as GCSE and below students will have little to no knowledge of the subject area and their results are not applicable as a consequence. Conversely a university student would be expected to have a wide understanding of the subject area and the result may not be of use to them. The majority of results are based in the AS and A2 range as these include the target audience for the product.

While the GCSE teacher may seem irrelevant they may also have experience teaching at A level standard and therefore the result cannot be discarded.

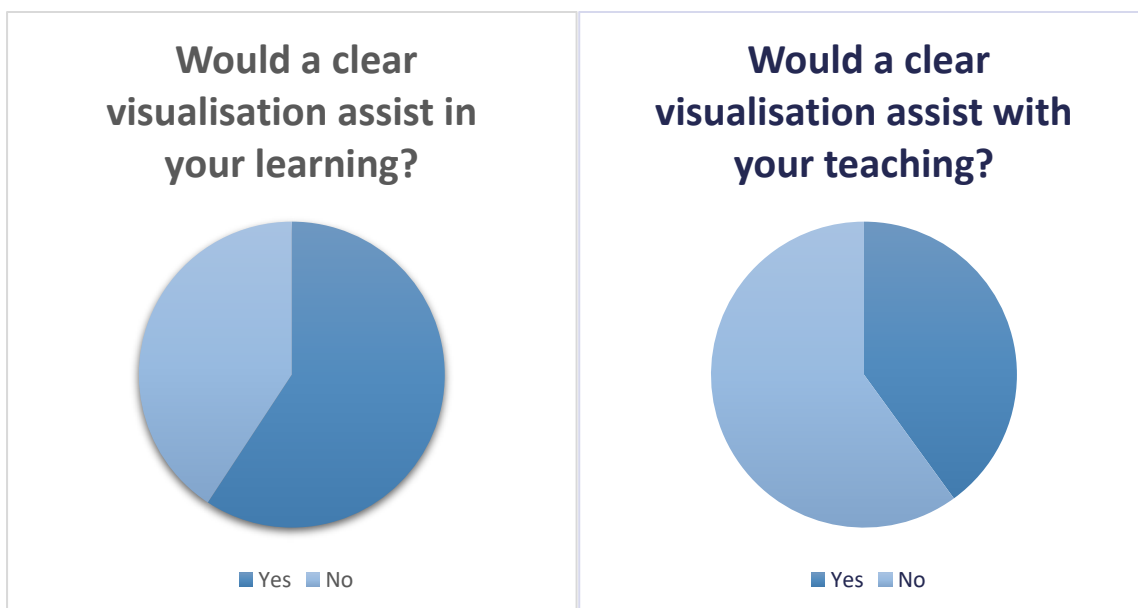


The opposing skews of these two graphs show imposing views of the understanding of gravitational potential. A possible reason for this could be the time at which the questionnaire was asked, before the students had started learning gravitational potential.

This would mean that students are confident in their knowledge of potentials before college and are perhaps unaware of the complexities of the subject. Therefore, it may be wiser to focus on the results from the teacher’s questionnaire – as they have the ability to say how the understanding has been for previous classes over past years. The results clearly show the teachers believe that understanding has been slightly below average.



The results from this question were clear: both students and teaching have encountered difficulties learning/getting students to understand gravitational potential. This shows the need for a clear resource for assisting in improving the comprehension of gravitational potential.



The student’s results suggest that they would benefit from a visual representation of the problem, and that they would use a resource if there was one. However, the consensus from the teacher’s results is that one

is unnecessary. Could this be due to the small sample size? Should this just be used as an extra-curricular learning resource?

Discussion with a teacher

To assist in designing and recognising the successfulness of the product I approached Mr G. Weston, a physics teacher at a local college, who agreed some preliminary requirements with the notion of utilising such a product within lesson time.

Here is a summary of the conversation exchanged:

After explaining my intentions to produce a product to aid in visualising gravitational fields Mr Weston was quick to suggest the different aspects that my system should include. He stated that a visualisation of gravitational potentials would be especially useful and that any system must be able to show the gravitational force at a point within a system.

When queried further about how such a visualisation would look Mr Weston was able to refer me to other similar programs that were used for alternate similar purposes, such as the electric field equivalent, where potentials were shown by equipotential lines of in growing rings around the system of charges, and the electric force was displayed by a combination of vector arrows. He also thought that a representation that showed the potential wells could also be useful.

Mr Weston was also quick to point out the need for such a product as many of his students had struggled to understand the concepts involved in the first time and that his quick 'scribbles' on the whiteboard were crude images that hardly aided in his demonstrations. While he occasionally used the electric field equivalent to model the gravitational system, he had generally found that this was detrimental and only served to confuse his students, who had not yet learnt about electric fields.

Moving on it was noted that the majority of his lessons were a combination of lecture followed by textbook questions, and that this repetitive nature was believed to be a little dull and monotonous for the students. Perhaps a variation in this method could be refreshing for the students. The conversation then progressed to online learning, where he mentioned that his physics department was making a definitive effort to move to more online learning and improving the online resources available. The reasoning for this was that an increased quality of resources would encourage more online learning and more independence from the students. With the current facilities available there is a lot of frustration stemming from the ineffectiveness of the products, and that it was not uncommon that a student would use this as an excuse for not completing the set work.

What utilities are in existence already?

By searching online for gravitational potential visualisation some of the easily found tools are:

Isaac Physics – Sound explanations on both gravity and gravitational potential with the occasional diagram. Explanations are generally in the structure of brief introduction, equations given, aspects of equations explained. Information is ordered in a manner to improve understanding with gradual increments into the depth of the subject covering a wide range of information. There are the generally 1-2 questions at the end of each chapter. However, if the questions are not understood there are no hints in this section and no easy way to see the method through which the answer has been obtained.

Isaac Physics was a learning tool used during physics lessons in my previous year, and the consensus was that it was not user-friendly, the hints were unhelpful, and the vague message received through an incorrect answer did not benefit questions. In summary this is a very useful revision and practice resource but not suited for learning or teaching, and only beneficial once an understanding of the topic has been gained.

On the other hand Isaac Physics was very helpful for teachers as it allowed for work to be set over a large set of topics, and returned a summary of the classes answers showing which questions were perceived to be hard and many students struggled with, Allowing for corrections to be made during lesson time.

The gravitational potential energy (GPE) of a body of mass m in a gravitational field is the energy stored as a result of the body's position in the field. If the field is due to a uniform spherical mass M , and the particle is outside of M at a distance r from the centre of M , then the particle's gravitational potential energy is conventionally given by

$$U_{\text{grav}} = -\frac{GMm}{r},$$

Figure 1: Isaac Physics

Khan Academy – With each subject and topic broken down into individual sections Khan academy provides a useful teaching, learning and subject reinforcement tool. Combined with useful diagrams and YouTube videos this creates a helpful learning tool. However, on this particular topic the information regarding potentials further away from a planet’s surface was lacking and the sole diagram is not perfectly clear.

What if the gravitational field is not uniform?

If the problem involves large distances, we can no longer assume that the gravitational field is uniform. If we recall [Newton's law of gravitation](#), the attractive force between two masses, m_1 and m_2 , decreases with separation distance r squared. If G is the [gravitational constant](#),

$$F = \frac{Gm_1m_2}{r^2}.$$

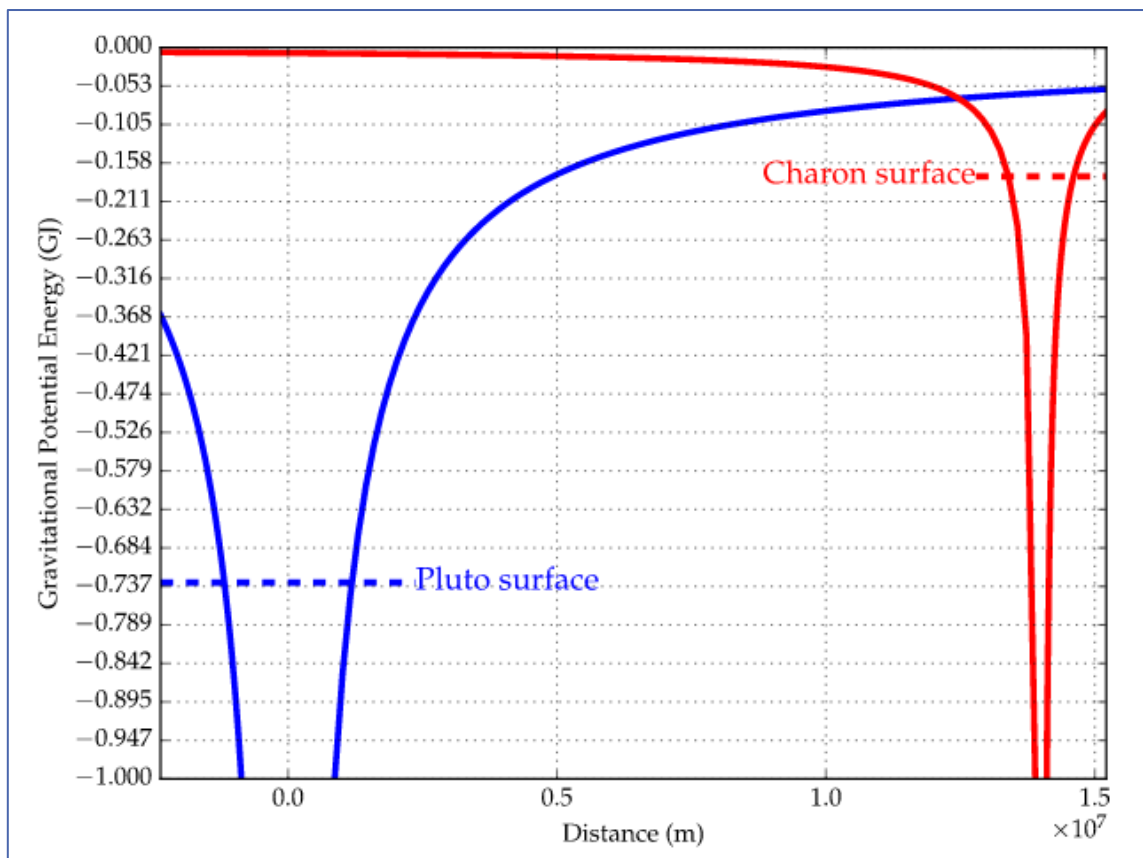


Figure 2: Khan Academy

Phys.Libretexts.org – Coherent yet simple explanations, derived from a different point of view to other resources that makes sense and benefit the reader. Plenty of equations are laid out in a sensible manner, however much information seems to be missing.

There are some graphs and visualisations, yet these are not clear, and frankly not aesthetically pleasing, and therefore unlikely to benefit a reader to a sufficient level. There are brief explanations on two body systems, yet this section completely lacks any images and feels poor in comparison to earlier information.

To see what this potential function looks like on a larger scale, going far from the Earth, it is necessary first to decide where it is most natural to set it equal to zero. The standard convention is to set the potential energy equal to zero at $r = \text{infinity}$! The reason is that if two bodies are very far from each other, they have no influence on each other's movements, so it is pointless to include a term in their total energy which depends on their mutual interaction.

Taking the potential energy zero at infinity gives the simple form

$$U(r) = -\frac{GMm}{r} \quad (1.3.7)$$

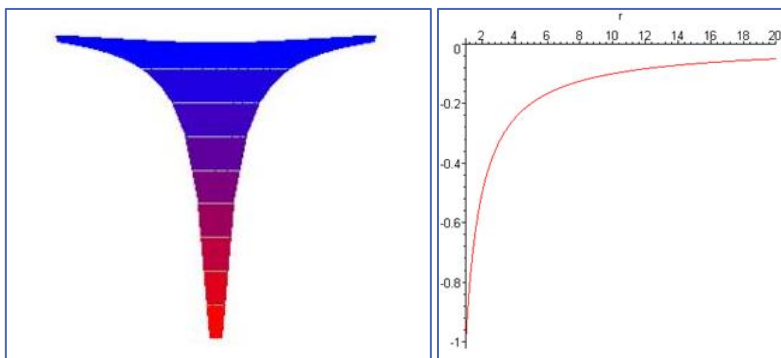
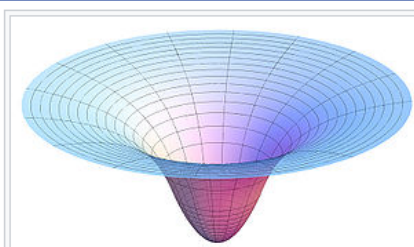


Figure 3: phys.libretexts.org

Wikipedia – Detailed and expansive resource, with beneficial diagrams. Yet unspecific target audience could lead to confusion for readers of different abilities. Concise descriptions transfer large amounts of knowledge with links to further reading. However, the discouragement of the use of Wikipedia due to the inaccurate information the site can hold – as a result of its editable web pages – means that Wikipedia is unsuitable for educational needs.

In classical mechanics, the **gravitational potential** at a location is equal to the work (energy transferred) per unit mass that would be needed to move the object from a fixed reference location to the location of the object. It is analogous to the electric potential with mass playing the role of charge. The reference location, where the potential is zero, is by convention infinitely far away from any mass, resulting in a negative potential at any finite distance.



Plot of a two-dimensional slice of the gravitational potential in and around a uniform spherical body. The inflection points of the cross-section are at the surface of the body.

Figure 4: Wikipedia

TexasGateway.org – Designed as a lesson with clear learning objectives, formulated from well dictated sections that link together ideas. Some diagrams and models but nothing particularly helpful with some worked question examples. Educates gravitational potential at short and long ranges covering multiple formulae. Though paragraphs tend to be excessively wordy which hinders the teaching capabilities of the piece as it is harder to read.

Newton’s Universal Law of Gravitation and Gravitational Potential Energy

Near the surface of Earth, where the gravitational force on an object of mass m is given by $F = mg$, there is an associated gravitational potential energy, $\Delta PE_g = mgh$, where h is the height above some reference value (e.g., sea level), and the potential is defined to be zero at that reference height ($h = 0$). In chapter 6 we learned that the magnitude of the gravitational force between two bodies having masses m and M , with a distance r between their centers of mass, is given by the equation $F = G \frac{mM}{r^2}$. Again, in this case, an associated gravitational potential energy can be determined by

$$\Delta PE_g = -G \frac{mM}{r}$$

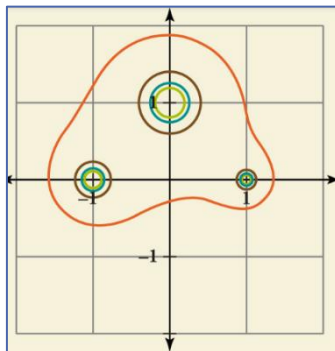


Figure 5: Texasgateway

During a brief researching session on the internet there were no clear models or visualisations of gravitational potential.

Notable Simulations

PhET Charges and Fields – While a different topic to gravitational potential this model is still a valid example. It is simple and easy to understand without prior knowledge, this is something that should be included in any educational model as a student should be able to use this by themselves. The toggleable options for voltage and values add layers of depth to the model that allow a user to start from a simpler model and build up. The inclusion of a measuring tool to calculate the electric field strength at any point is also a noteworthy addition as it means that a user can see how the field strength varies.

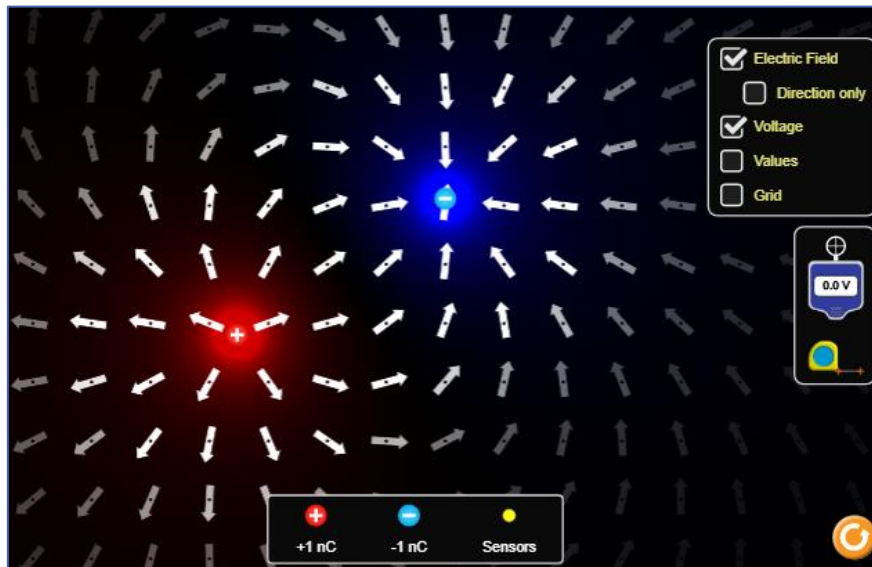


Figure 6: PhET Charges and Fields example 1

The user can also use the measuring tool to visualise equipotential lines at any point before combining with the values feature to see the calculated field strength of that equipotential as seen in example 2. Additionally the user can use a sensor to see how the field will affect a particle at any position, simply displaying a resultant force arrow with length proportional to the force experienced.

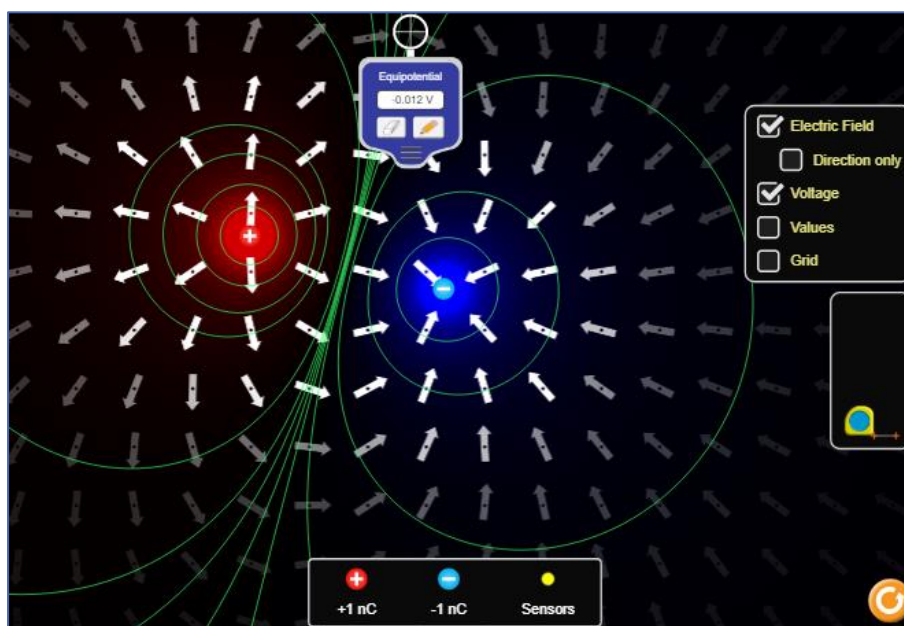


Figure 7: PhET Charges and Fields example 2

FlashPhysics Electric Field – A simple to use model with multiple pre-sets and clearly labelled options. Options are divided into auto-plotting functions, mouse controls, charge addition/modification/removal and sample charges. This allows simple navigation and use of the product.

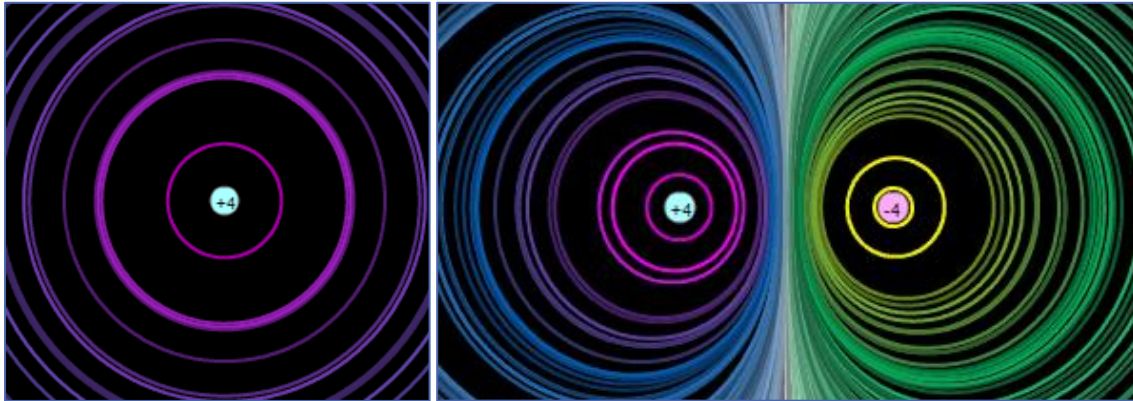


Figure 8: FlashPhysics 1

The equipotential auto-plotting facility is simple to use and easy to tell what is happening with the colour code. However, there is no key for the colour code so the magnitude of the potential must be inferred. It is also difficult to get a clear spread due to the random nature of its operation which leads to solid rings of colour. This can be accomplished through the mouse control version and clicking where you want an equipotential, however there are no instructions for this, and it is not immediately obvious.

Other options include showing field vectors, field lines and a test charge. The field vectors are an interesting inclusion as it displays not only the resultant vector from every charge present, but the vectors caused by each charge present in the system.

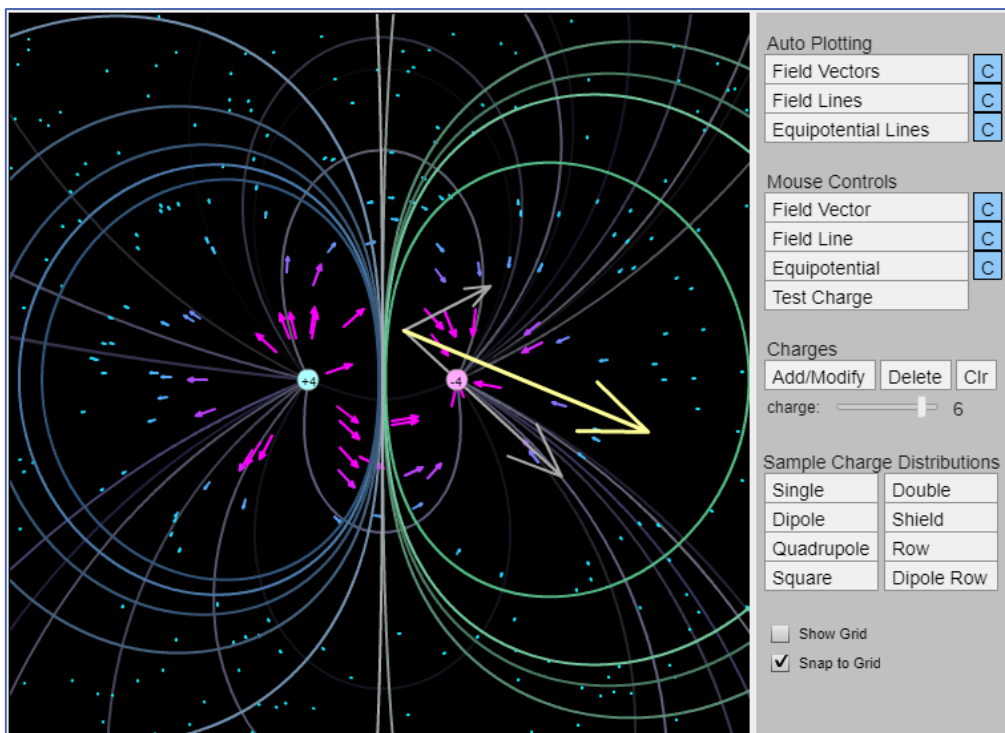


Figure 9: FlashPhysics 2

Textbooks

Within the textbook there were limited diagrams helping to explain the concept with a clear favouritism towards graphs. A few of the diagrams that have been selected are here as follows:



Figure 10: Textbook example diagram 1

The preceding diagram is used within the textbook to help show **Newton's Law of Gravitation** showing how each mass feels a force direction towards the other. There are a few other diagrams along with this showing that no matter the masses of the two bodies the force between them will always be equal.

The next diagram shows the direction of the field lines going into a single mass, the dotted line around the outside is an equipotential line. This shows that the radial field decreases with the distance away from the centre of mass.

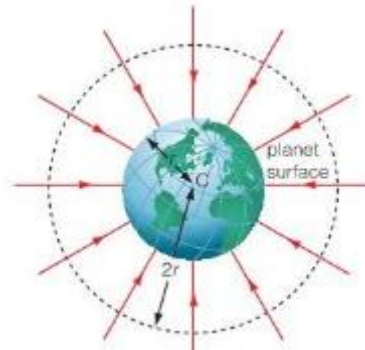


Figure 11: Textbook example diagram 2

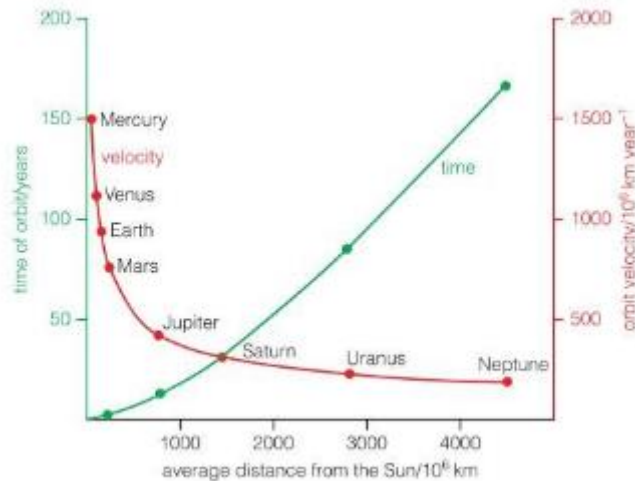


Figure 12: Textbook example graph

Figure 12: Textbook example graph

The graph above shows how the time period of an orbit relates to the average distance from the sun and compares it to a graph of orbital velocity against orbital distance. The textbook then goes on to link into Kepler's equations which describe the relationship between time period and orbital radius.

The following is an example of an explanation given by the textbook that should be sufficient to give a student a solid understanding of escape velocities, the user should also be aware of the formulae used to calculate the escape velocity.

Escape velocity

The Earth has its atmosphere because the molecules of gas, moving in our atmosphere, do not have enough kinetic energy to escape from the pull of gravity at the Earth's surface. So how fast does something have to move to escape from the Earth's surface?

When a fast-moving object leaves the surface of a planet, we can write that:

decrease in kinetic energy = increase in gravitational potential energy

$$\Delta E_k = \Delta E_p$$

This assumes that the object is not affected by an atmosphere, and is in free fall – this is not a spacecraft with a rocket.

The equation above can be written as

$$\frac{1}{2}mv_1^2 - \frac{1}{2}mv_2^2 = m\Delta V$$

If the object is to just escape the pull of the planet, its speed, v_2 , will just reach zero at an infinite distance from the planet. This leads to the idea of **escape velocity**, which is the minimum velocity that an object must have at the surface of a planet in order to escape the pull of gravity of the planet using its own kinetic energy.

The gravitational potential at the surface of a planet is given by

$$V = -\frac{GM}{r}$$

so the change in potential is

$$\Delta V = \frac{GM}{r}$$

So the escape velocity for a planet can be calculated using

$$\frac{1}{2}mv^2 = m\Delta V = \frac{GMm}{r}$$

Figure 13: Textbook example explanation

The textbook has a large range of questions in each topic, the following is a screenshot of a section of these questions. There are a variety of written response and calculation based questions designed to aid a users development as they progress through the chapter.

TEST YOURSELF

14 Explain why we choose to define the zero point of gravitational potential at an infinite distance from any planet.

15 Calculate the gravitational potential at the surface of Jupiter. The mass of the planet is 1.9×10^{27} kg and its radius is 70 000 km.

16 This question is based on the information in Figure 3.8.

a) Calculate the work done in taking a 1200 kg spacecraft from

- i)** point 1 to point 2
- ii)** point 2 to point 3
- iii)** point 3 to point 4
- iv)** point 4 to point 5.

b) Use your answer to part (a) to explain why a spacecraft can stay in a circular orbit round a planet indefinitely.

c) The spacecraft returns to the planet. It passes point 5 travelling at a speed of 5200 m s^{-1} , and falls freely to point 2. How fast is it travelling as it passes point 2?

17 The radius of the planet shown in Figure 3.8 is 10 000 km. Either by estimating the distance between the equipotentials near the planet's surface, or otherwise, calculate the value of g near the planet's surface.

18 When a large star collapses at the end of its life, it can collapse into a black hole. The pull of gravity at its surface is so strong that not even light can escape. Einstein's theory of general relativity predicts that black holes are 'singularities', which means they have collapsed into a tiny space. However, we can use Newton's theory to calculate the maximum size of such a hole.

a) Use the equation for escape velocity (marked [iv] in the text) to calculate the maximum radius for a black hole formed by a star of mass 10^{31} kg. The speed of light is $3 \times 10^8 \text{ m s}^{-1}$.

b) Calculate:

- i)** the gravitational field at this surface
- ii)** the gravitational potential at this surface.

Figure 14: Textbook example questions

Experiences in direct tuition

My experiences in direct tuition could be best described as mixed. Within the first 5 minutes of starting the topic we were told most students in our class would not understand the concept this time. The inverse square law fields were taught separately in the hopes that we (the students) would understand the second time and be able to retroactively learn gravitational fields. Giving the impression that in order to achieve good marks we would have to put in large amounts of hours to teach ourselves.

Alongside a few hastily drawn (and redrawn) diagrams on the whiteboard we were taught a simple derivation of the gravitational force exerted on a mass. When teaching the connection between potential energy and force we were taught a simpler method of combining equations rather than the real calculus methods that are technically correct. While not required for our course the calculus is not beyond A level standards and it would have been nice to see a proper derivation as in some cases it may further understanding of a subject.

All definitions and laws were correctly stated and ingrained throughout the lesson as would be expected.

In the successive lessons potential gradients and escape velocities were covered before leading on to the link between gravitational force and centripetal motion, an important part of astronomy. In my opinion the coverage of potential gradients and escape velocities were too brief and this left multiple students confused and without a proper understanding. That particular lesson comprised a lot of textbook based work where details are easy to gloss over and students get distracted. On the other hand the link to centripetal motion was thoroughly explained and in a detailed manner which had the opposite, positive effect as one would expect.

In summary the tuition could have been more detailed in parts and would have benefitted from a few more accurate diagrams.

What are Energy and Potential Energy?

Energy is the capacity to do work, it may exist in various forms such as potential, kinetic, thermal, electrical, chemical and nuclear. The **law of the conservation of energy** states that '*the total energy of an isolated system remains constant*'; this law means that energy can neither be created nor destroyed, only transformed or transferred from one form to another.

Work is done on an object when a force acting on it makes it move, and as a result energy is transferred to the object.

Energy as a result, is inextricably linked to work which leads to the **Work-Energy Principle**. This states that an increase in the kinetic energy of a rigid body is caused by an equal amount of positive work done by the resultant force acting on the body. And conversely, a decrease in kinetic energy is caused by an equal amount of negative work done by the resultant force.

$$\Delta E = w$$

Potential energy is the energy of an object due to its position.

The **total mechanical energy** of an object is the sum of its potential energies, and kinetic energy.

WHAT IS GRAVITATION AND GRAVITATIONAL POTENTIAL ENERGY?

Gravitation is the movement, or a tendency to move, towards a centre of gravity. The **gravitational force** is a force that attracts any two massive objects. **Newton's law of gravitation** proposes 'every particle attracts any other particle with a gravitational force whose magnitude is given by:

$$F = G \frac{m_1 m_2}{r^2}$$

Where m_1 and m_2 are the masses of the particles and r is the distance between them. G is the **gravitational constant** $6.67 \times 10^{-11} \text{ Nm}^2/\text{kg}^2$

The **principle of gravitational superposition** says a net effect is the sum of the individual effects. For a given group of particles the net gravitational force exerted on any one of them is:

$$F_1 = F_{1-2} + F_{1-3} + F_{1-4} + \dots + F_{1-n}$$

This can be expressed as the vector sum: $\sum_{i=2}^n (F_{1 \rightarrow i})$

A **Gravitational Field** is a region of space around a mass in which another body experiences a force of gravitational attraction.

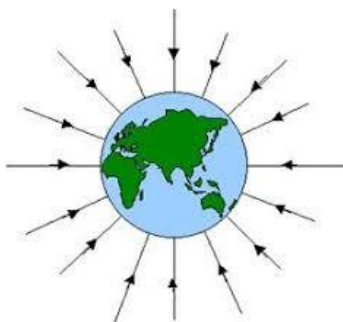
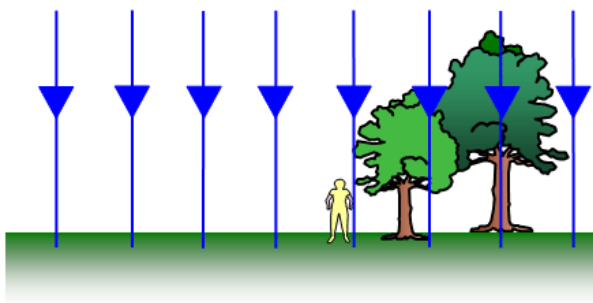


Figure 15: Radial Field

The gravitational field around a mass is radial, meaning it always acts towards the centre of a mass.

The gravitational field strength = Force Due to Gravity / Mass

$$g = F/m = GM/r^2$$



At the earth's surface we can assume a uniform field where the force due to gravity has a value of 9.8 ms^{-2} . However this value is only accurate near the earth so this approximation has limits.

Figure 16: Uniform Field

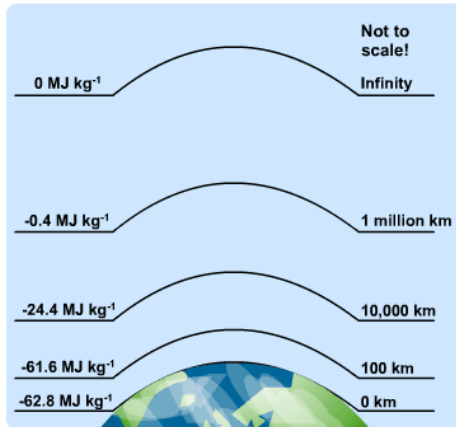
Gravitational Potential Energy is the work done against gravity to raise a mass to a given height. In a uniform field the gravitational potential energy, U_g , is simply equal to the mass \times force of gravity \times height.

$$\Delta U = mg\Delta h$$

Where h is the height above the point taken to be at a potential energy of 0J.

Potential is defined as the potential energy / mass.

Gravitational Potential is the work done to move a mass from infinity to a given point in a gravitational field per unit mass.

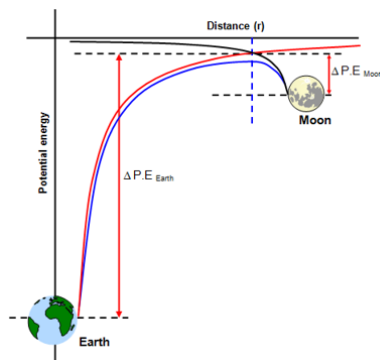


As distance from the centre of mass increases the gravitational potential approaches 0J, as the mass is moved closer to infinity.

Lines of equal gravitational potential can be shown radiating outward from the centre of mass. These equipotential lines will become more spaced for the same change in potential energy due to the inverse square relationship.

Potential gradients

Figure 17: Equipotential lines surrounding the earth



Another way of showing this is through potential wells where a mass is shown at the bottom of each 'well' and the size of the well is directly proportional to the potential energy of a test mass at set distances from each mass.

Figure 18: Planetary Potential Wells

To calculate the gravitational potential energy at a point in a gravitational field you need to integrate the gravitational force, F_g , with respect to distance.

$$U_g = \int_0^{\infty} F_g \partial r : \quad -\infty \quad \frac{GMm}{r}$$

$$U_g = -\frac{GMm}{r}$$

As gravitational potential is equal to the gravitational potential energy per unit mass:

$$U_g = \frac{V_g}{m} = -\frac{GM}{r}$$

So we get our final equation:

$$V_g = -\frac{GM}{r}$$

Research Sources

Type	Name of Source	Summary of information	Relevant sections
Book	<u>AQA Physics A Level 2nd Edition</u> Author: Jim Breithaupt ISBN: 978-0-19-835187	General subject knowledge and definitions.	Section 7, 21.1-21.5, Pages 336-353
Book	<u>Extended Fundamentals of Physics Fourth Edition</u> Authors: David Halliday, Robert Resnick, Jearl Walker	More in depth subject knowledge and derivations.	Pages 192-194, 420-423
Book	<i>Hodder AQA A level year 2 textbook</i>	---	---
Website	http://hyperphysics.phy-astr.gsu.edu/hbase/Mechanics/lagpt.html	Lagrange Points, Three body Equipotential Surfaces and Equipotential Contours.	n/a
Website	https://isaacphysics.org/concepts/cp_gravitational_field https://isaacphysics.org/concepts/cp_potential#acc_cp_potential_relatng_field_potential	Great database of information and equations laid out in comprehensible manner.	n/a
Website	https://phys.libretexts.org/Bookshelves/Astronomy_and_Cosmology/TextMaps/Supplemental_Modules_(Astronomy_and_Cosmology)/Astronomy/Gravity/1.3%3A_Working_with_Gravity%3A_Potential_Energy	Explanation of gravitation and potentials from a university physics professor.	n/a

Analysis

Previous System Questions

Questions typically listed in small exercises, containing 4-6 questions; generally split into 3 parts. These questions generally increase in difficulty, refining the techniques and solidifying the knowledge learnt. The last questions should challenge the students.

Q: Kepler's Third Law



The magnitude of the gravitational force between two masses M and m is given by Newton's Law of Gravitation:

$$F = \frac{GMm}{r^2}$$

where G is the gravitational constant ($6.67 \times 10^{-11} \text{ m}^3 \text{ kg}^{-1} \text{ s}^{-2}$).

Kepler's Third Law, for planets in circular orbits, states that the square of time period T is proportional to the cube of the radius of the orbit R i.e.

$$T^2 = kR^3$$

Answer Now Hint 1 Hint 2 Hint 3 Hint 4 Hint 5

Given that the mass of the sun is $1.99 \times 10^{30} \text{ kg}$ prove that this is true in the special case of a circular planetary orbit and find the constant of proportionality, k and its units.

Value Units

Please answer to an appropriate number of significant figures. Please choose an appropriate unit of measurement.

Correct!

Well done!

Kepler's third law is also true for elliptical orbits if the radius of the orbit in the formula is replaced by the length of the semi-major axis of the orbit.

Figure 19: Isaac Physics Model Question

Textbook previous questions

Current System Analysis

A lesson can typically be divided into 4 parts. Firstly at the start of the lesson a student receives their marked work from the previous week, and a brief discussion of the questions that were deemed to more challenging by the classes results, addressing typical sources of errors and poor methods.

Then the teacher would introduce the next aspect of the topic and explain the key principles, generally with the aid of a PowerPoint. This informs the student of the knowledge they need to learn and practice. In order to teach this the teacher generally works through the PowerPoint, sometimes drawing additional diagrams and showing the origination of certain formula to improve student comprehension.

However, it is not uncommon to see teachers skipping multiple 'unnecessary' slides in the PowerPoint which are supposedly unimportant. This suggests that the PowerPoints are out of date.

Following this a teacher will work through some simpler problems to show how the formulas learnt can be applied to the issues. These problems are generally very simple and require little to no rearranging of formulas.

Finally, to finish a lesson the students will work through the example questions, generally from the textbook, which gradually increase in difficulty. Within a section there is a small variety of questions which are then expanded on during the topics summary questions.

To cement learning a teacher will generally set homework, ranging between 5 and 20 questions which is expected to take about 45 minutes to an hour. Covering a variety of question types and difficulties. Homework is generally set every lesson and the work from the previous week is marked manually by the teacher and returned to the student.

Each teacher keeps a record of which the homework results in each class, which stores the result the student has attained and the week the homework was set. If the student has not done that homework then that week is simply left blank allowing the teacher to see how many homework's the student has not completed.

It is then up to the student to maintain each topic in preparation for a test or the final exam, with a possibility of revision lessons for some topics depending on the rate at which work had been covered.

IPSO Diagrams

IPSO FOR A TEXTBOOK BASED SYSTEM

Input	Process
Student completes the questions, presumably with detailed workings that outline the processes used to reach their answer.	In order to mark the homework, the student will generally bring their work in to be marked by a teacher. In some cases answers are contained at the end of the textbook so the students are able to confirm their answers independently
Storage	Output
The textbook stores the questions and in some cases the answers. The student has to store their work. If answers are not available it is likely the teachers will also have a copy of work outlining the correct answers	After marking the students results are calculated and a judgement on their comprehension can be made. Leading to actions.

IPSO FOR A COMPUTERISED QUESTION SYSTEM

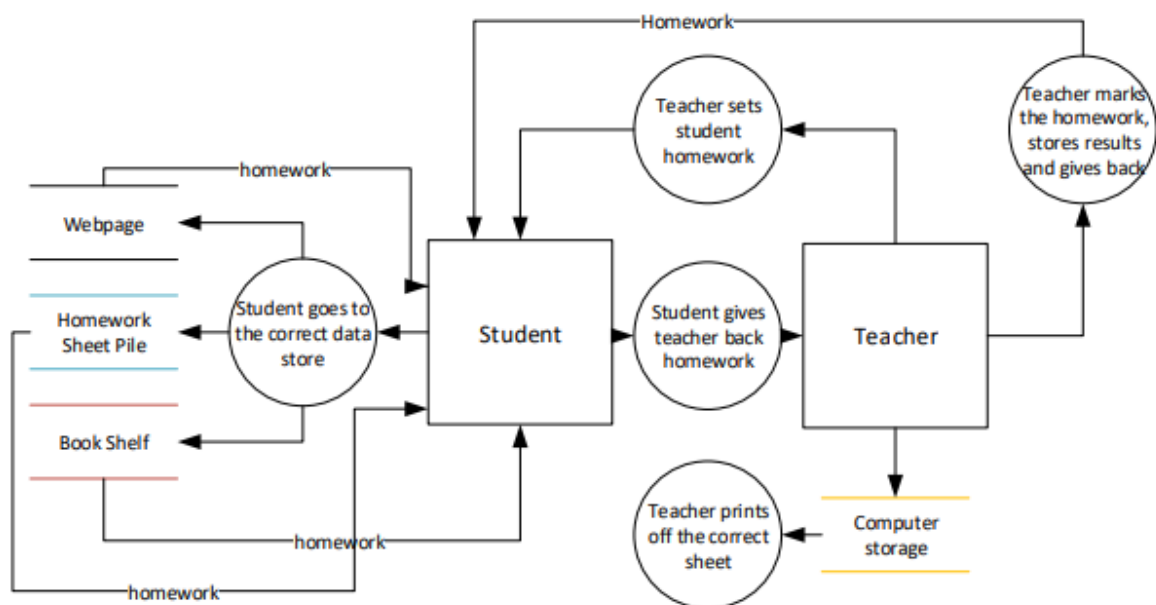
Input	Process
Student completes the questions on their account, and enters their answer into the system, until a correct answer is entered	The system compares the answer entered with the correct answer and returns whether it is correct or not.
Storage	Output
The system stores: <ul style="list-style-type: none"> • The number of answers entered (attempts) for each question. • Whether the student answered each question correctly. • Which student each set of results corresponds to. 	All results are outputted in a tabulated format that is easy to read, and colour coded for how well each student did. This is only visible to the teacher. For each question the student receives feedback on whether their answer is correct, and where the error(s) may lie.

DOCUMENT SPECIFICATION

Volumetric For a Computerised based system (Isaac Physics)				
Document Description	System	Document	Name	Sheet
Isaac Physics Question	Current System	1	n/a	1
Stationery ref.	Size	No. of parts	Method of preparation	
Physics Question	n/a		Online Webpage	
Filing Sequence		Medium		Prepared by
		Computer		Cambridge University
Frequency of Preparation		Retention Period		Location of File
n/a				Isaac Physics Link
	Minimum	Maximum	Average	Growth Rate/fluctuations
	n/a	Unlimited	1 per person	Dependent on use.
Users/Receipts		Purpose		Frequency of use
students		Homework / Exam Preparation		Very Frequent
Data Dictionary				
Ref	Name	Data Type	Occurrence	Source of Data
1	Question	String	Once per question	Isaac Physics example question
2	Gravitational Constant	Single	Once per question sheet	Isaac Physics example question
3	Solar Mass	Single	Question Specific	Isaac Physics example question
4	Hint 1 (topics)	URL(s)	Question Specific	Isaac Physics example question
5	Hint 2 (goal)	String	Question Specific	Isaac Physics example question
6	Hint 3 (Useful equations)	String	Question Specific	Isaac Physics example question
7	Hint 4 (Diagram)	Image	Question Specific	Isaac Physics example question
8	Hint 5 (Video)	URL	Question Specific	Isaac Physics example question

Volumetric For a textbook based system				
Document Description	System	Document	Name	Sheet
Textbook questions	Current System	1	n/a	1
Stationery ref.	Size	No. of parts	Method of preparation	
Physics Question	< A4	4	Textbook	
Filing Sequence		Medium	Prepared by	
Every time set		Paper/Digitised Book	Hodder Education	
Frequency of Preparation		Retention Period	Location of File	
Once a year		0. Copies are generally scanned in and posted online.	Classroom, eBook	
	Minimum	Maximum	Average	Growth Rate/fluctuations
	1	~60	1 per person	Depends on the number of students.
Users/Receipts	Purpose			Frequency of use
Students	Homework			Once for homework, more for revision is required.
Data Dictionary				
Ref	Name	Data Type	Occurrence	Source of Data
1	Question	String	Once per question	Hodder Education Physics year 2 textbook
2	Gravitational Constant	Single	Once per question sheet	Hodder Education Physics year 2 textbook

Data Flow Diagrams



In the current system each homework must be manually set by the teacher during lesson time. This involves the teacher finding the location of the homework sheet, which is stored in either a book or on the teachers computer.

In the case of the homework being located in a book the teacher will tell the student the book title, page numbers and questions. Expecting the student to look up the questions and return the following week with the answers. In rare occasions a teacher may scan a copy of the homework page and upload it to the college website, where the students can access the image, which generally stored in a PDF format.

If the homework sheet is located in the teacher's computer files, then the teacher must first print off around 20 copies of the sheet before handing them out to each individual student. This is more convenient for the student but at the expense of the teacher's time.

Summary of Research and Analysis

The results of my Surveys, Experiences and Conversations with relevant people in the subject field combined with research into the current state of learning resources have shown there is a clear need for a resource that displays and helps students understand the concept of gravitational potentials.

From my analysis it is also clear that a system that can help explain parts of the physics specification whilst also digitising the system would be beneficial to both students and teachers of the subject. Enabling teachers to reduce time wasted and spend longer on the more challenging areas of the syllabus, deepening the students understanding to both the course and explaining the finer details that would may improve the students grade.

However due to the complexity of the problem the visualisation element of the simulation must be clear and clearly explained or self explanatory for it to be useable, in both a class environment and outside in independent study.

Requirements

SIMULATION/MODEL REQUIREMENTS

1. The model should be easily to use:
 - 1.1 The simulation interface should be well designed and organised in a logical manner.
 - 1.2 The interface controls should be named so that their purpose is immediately understandable without prior knowledge.
 - 1.3 Each control should have a brief description with instructions on how to use it.
 - 1.4 Any inputs should be validated or the available values should be limited to prevent a system malfunction.
2. The visualisation should be immediately understandable with the information available within the system:
 - 2.1 The model must be clear and obvious what it is supposed to be visualising.
 - 2.2 The gravitational field strength must be correctly calculated for any point within the gravitational field.
 - 2.3 The gravitational potential must be correctly calculated for any point within the gravitational field.
 - 2.4 The Gravitational potential function should produce concentric equipotential lines that are appropriate for the system that has been created.
 - 2.5 The Create Field Line function should produce a set of gravitational field lines that correctly demonstrate the field of the created system.
 - 2.6 The test mass function should correctly show the movement of masses in the field due to the gravitational field.
 - 2.7 The model should be beneficial in the classroom and at home.

LEARNING RESOURCE REQUIREMENTS

1. Should be educational, useable in a classroom and beneficial to students.
 - 1.1 The information should cover the entirety of the subjects specification.
 - 1.2 The detail of the information must be sufficient to leave the students with a strong understanding of the subject.
2. There should be suitably challenging questions for the students to answer
 - 2.1 The Questions should have a range of difficulties to develop the users skills.
 - 2.1.1 All students should be able to answer the easiest questions
 - 2.1.2 The most challenging questions should be testing for the majority of students, and should require the adaptation of principles along with a depth of subject knowledge to answer.
 - 2.1.3 Over a range of questions students should be prepared for all common exam question scenarios.
3. There should be a form of feedback available so that the students can view their progress, and what aspects they need to revise.
 - 3.1 The students should be able to see the percentage they have achieved on each test.
 - 3.2 The students should receive a feedback message from their teacher.

DATABASE REQUIREMENTS

1. Database should be designed logically:
 - 1.1 All tables within the database should be fully normalised:
 - 1.1.1 Tables should only contain atomized values.
 - 1.1.2 Values stored within the same column should be of the same domain and data type.
 - 1.1.3 All columns should be appropriately named so that they are self explanatory and any inter-tabular links are evident without using the design.
 - 1.1.4 The order in which data is stored should not matter.
 - 1.2 All data stored within the database should be relevant in some respect.
2. The system must be able to interact with the database.
 - 2.1 The database must be created correctly:
 - 2.1.1 Each table must be created with the correct name.
 - 2.1.2 Each table must have the correct number of fields.
 - 2.1.3 Primary keys and partial keys must be defined correctly for each table
 - 2.1.4 Each field must be correctly named.
 - 2.1.5 Each field must have the correct data type.
 - 2.2 The database should be able to store all necessary values:
 - 2.2.1 When an account is created the correct details should be correctly stored within the students or teacher table.
 - 2.2.2 When a students class code is changed this alteration should be reflected in the database.
 - 2.2.3 When a teacher adds a students homework the results table should be correctly filled in.
 - 2.2.4 When a teacher adjusts a students feedback the changes must be altered correctly in the feedback.
 - 2.3 The database should be able to answer a variety of queries:
 - 2.3.1 All and any data required to fulfil data fields within the system should be gathered by the system.
 - 2.3.2 Teachers must be able to view the feedback they have left the students in their class.
 - 2.3.3 Teachers should be able to view and alter the data involving the students within their class.
 - 2.3.4 Any changes made to the data within the system should be reflected in the database.

Justification Of Requirements

SIMULATION/MODEL REQUIREMENTS

As the model/simulation is mainly purposed to be a learning aid it must be simple for a student to understand without any prior knowledge of the subject or program. To achieve this each inputs and options should be distinctly named and categorized into like functions. These inputs must be validated and controlled as the model should always work independent of any mis-input.

All of the functions in the simulation must be able to produce the correct output image and calculate the correct values independent of the scenario created by the user. The produced output images must be accurate and easily understandable to the user.

To ensure the tool is usable in a classroom environment all the processing must be completed in real time so that there is little to no delay before an output is produced. This ensures that the product can be used during a demonstration. To aid its ease of use each control should be responsive and sensical so that multiple scenarios can be quickly established.

To be beneficial and therefore worth using the model must be clear and quickly understandable. Clear consideration should be taken to optimise the model and design for learning.

LEARNING RESOURCE REQUIREMENTS

The learning resource should be split into 2 distinct aspects: a relevant information source for educating and furthering understanding of the subject content, along with a large variety of appropriate practice questions.

The information lesson source should be designed so that it is compatible with normal educational use. For this reason it must be easy for students to use independent of their teachers. The source must be thorough enough so that the students have a complete understanding of the subject, yet easy and simple to understand so that it is beneficial to all students. This source should also include worked exemplars of common questions to demonstrate a strong method of answering calculation questions.

The practice questions should be reminiscent of the questions available in textbooks and exams with an emphasis on exam style. There should be a variety of questions so that a student can reinforce their understanding of the learnt concept, developing deeper links and improving their skills so that they are able to answer the majority of questions on the topic confidently.

For the students to adequately understand their progress it is essential that a form of feedback be available to them, so that they can gauge their understanding and know which areas need improvement.

DATABASE REQUIREMENTS

The database must be logically and efficiently designed to improve the effectiveness of the system and to make it easier to alter and improve in the future, as the system updates and evolves. To do this the data base should be fully normalised, and effectively labelled tables should be used to store relevant data only. The database must be robust and reliable as the system relies on it. As well as being able to quickly answer a small number of specific pre-defined queries to improve the usability of the system.

These pre-defined queries should return any data requested by the system. If at any point the system needs to examine the database and query for an array of information, for example if the system needs to know all the students in a single class, then a suitable query should be selected and run. It is required that this query returns the correct results.

During parts of the program values may be changed and these changes must be reflected in the database so that it is kept up to data and works as it should. This may involve classes being changed or new accounts being created. Or if feedback is being added or altered then it should only affect the student who's feedback it is, and no other data should be affected accidentally.

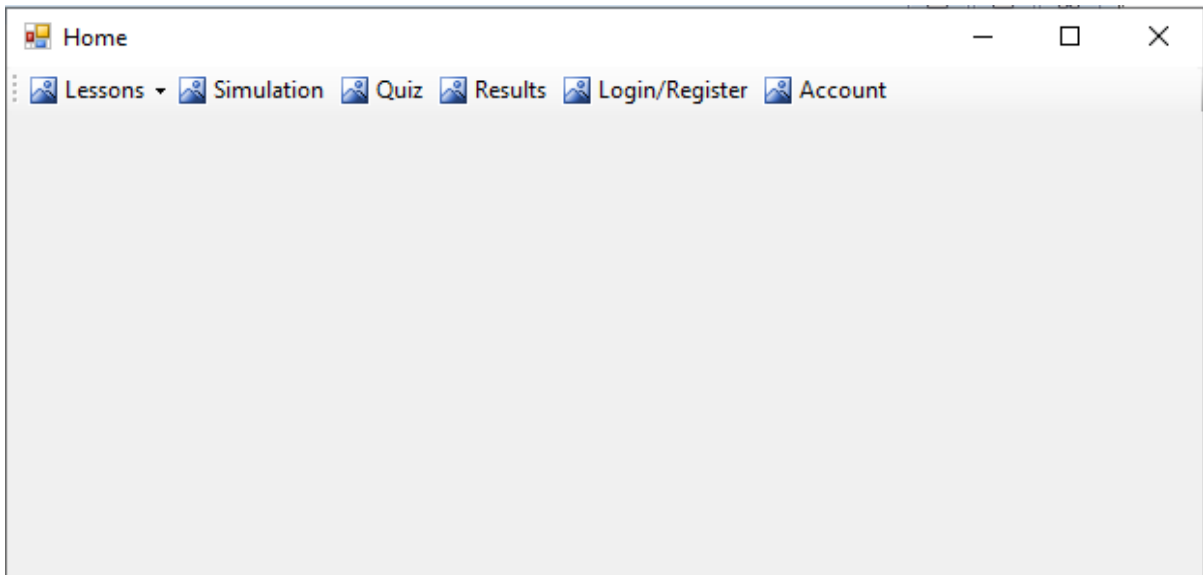
Data gathered from the database should also be displayed in a sensible and visually effective manner so that it is evident what data is on show and what its purpose is.

Design

Form Design

The form will be coded using windows basic in visual studio making use of the windows form application environment, enabling the use of multiple forms and easy design applications.

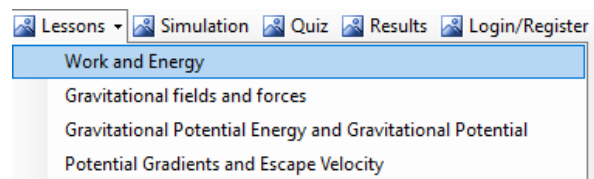
HOME FORM



The home form should be simple and lay out all the options a user can use. To do this I will make use of the ToolStrip tool available. By populating this toolbar with a combination of buttons and 'comboboxes' I will be able to create an accessible toolbar which is easy and simple to use.

FORM NAVIGATION

The **Lessons** tab will be created using a combobox that displays all the available lessons split into their respective topics, making it easy for a user to access specific areas of the subject. Each of the drop-down items will take you to the relevant information.



For the rest of the tabs a button will provide the appropriate service as a single click will open an alternate form which contains the information for that particular section. For example clicking on the quiz section will take the user to where they can complete the quiz.

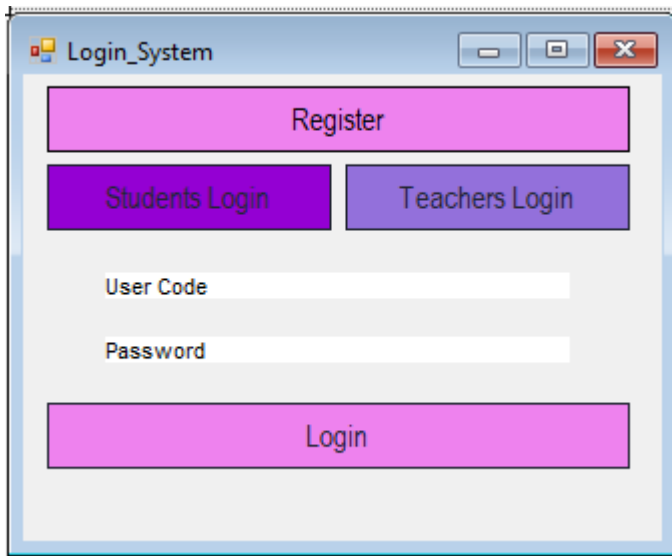
While not logged in the **Results** and **Account** tab will both take you to the **Login/Register** Form, as both of them require the user to be logged in on order to access the available information.

These tabs will be available in every form allowing the quick movement between pages.

LOGIN FORM

The login system must be clear and self explanatory. To make the use of the system clear the display will have a toggle to switch between Logging in and registering.

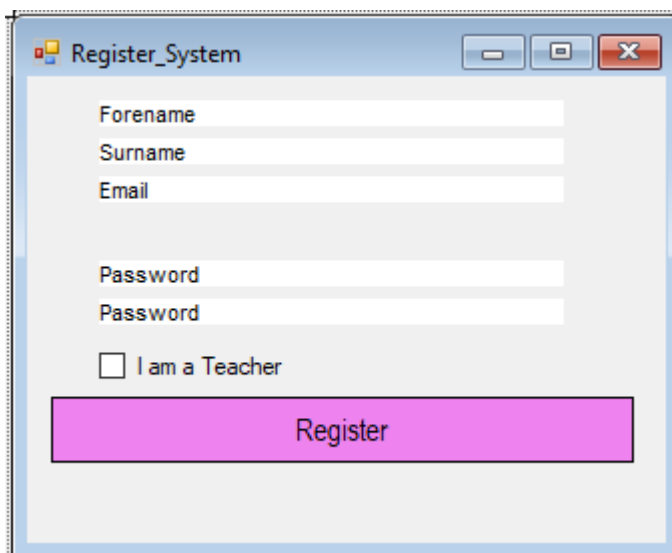
The login page should be simple and obvious, it should contain fields asking for the users Username and Password. This must be laid out simply and in a sensible manner. In the registering option it must contain the same as the login page but the user should be asked to confirm their chosen password and extra fields such as their name, surname, email, and an optional field containing the students Teacher (or class code for simplicity)



The screenshot shows a window titled "Login_System" with standard Windows window controls (minimize, maximize, close). The interface contains a large pink "Register" button at the top. Below it are two smaller buttons: a purple "Students Login" button and a blue "Teachers Login" button. Underneath these are two white text input fields labeled "User Code" and "Password". At the bottom of the window is a large pink "Login" button.

It should also be evident whether the user is electing to log in as a student or as a teacher.

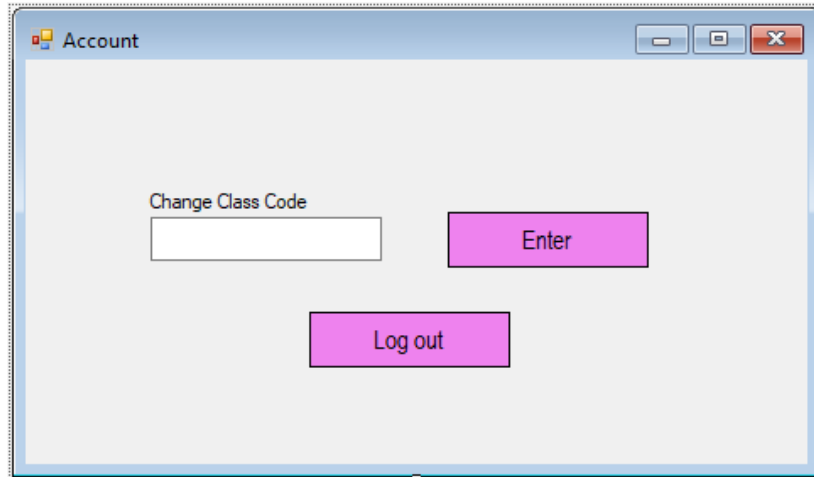
The registry form needs to have fields for all required data variables, such as forename and surname. It needs to be possible to register as both a student and a teacher.



The screenshot shows a window titled "Register_System" with standard Windows window controls (minimize, maximize, close). The interface contains four white text input fields: "Forename", "Surname", "Email", and "Password". Below the "Password" field is a second "Password" field for confirmation. There is a checkbox labeled "I am a Teacher" which is currently unchecked. At the bottom of the window is a large pink "Register" button.

ACCOUNT AND RESULTS

In the account page it should display the users information, and offer the ability amend the class field to enable a student user to add themselves to the relevant class. These changes should be mirrored in the database to keep the system up to date.



The screenshot shows a window titled "Account" with a light blue border and standard window controls (minimize, maximize, close) in the top right corner. The main content area is light gray and contains the following elements:

- A label "Change Class Code" positioned above a white text input field.
- A pink rectangular button labeled "Enter" to the right of the input field.
- A pink rectangular button labeled "Log out" centered below the "Enter" button.

Once logged in all users will be able to access the results page. For a student user the results page should automatically bring up all the quizzes they have completed, along with the percentage of marks they gained and the feedback left for them if there is any.

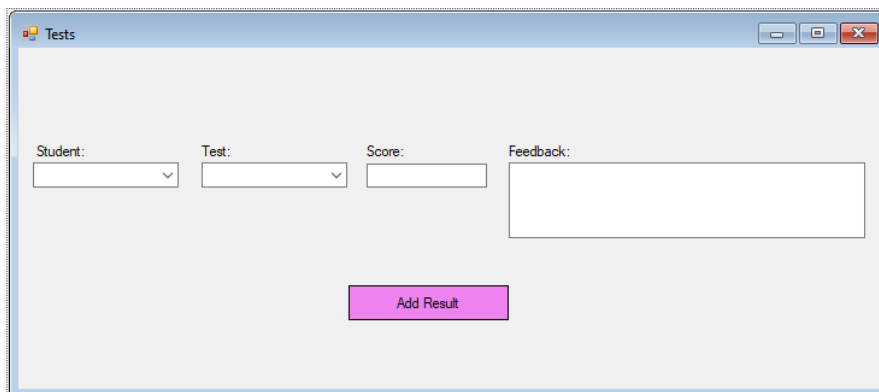
For a teacher the results form will first ask the teacher to fill out an SQL query in the form of selecting items from drop down lists, providing infallible validation of the data entry. The teacher should be able to filter to select specific classes, homework or students; and should be able to select as many items as required.

Once the search button is used the program should automatically run the desired SQL search and return all the wanted data in a presentable format.

TEST RESULT UPLOAD

The test upload should offer teachers the ability to select a student from their class and a test from the available tests. And to input the students score and feedback to save it on the system. This will enable the teacher to create multiple different SQL queries to add data to the results page of the database.

Once the user selects the add result button the entered data should be added if it is valid.



The screenshot shows a window titled "Tests" with a light blue border and standard window controls (minimize, maximize, close) in the top right corner. The main content area is light gray and contains the following elements:

- Four input fields arranged horizontally: "Student:" (a dropdown menu), "Test:" (a dropdown menu), "Score:" (a text input field), and "Feedback:" (a larger text input field).
- A pink rectangular button labeled "Add Result" centered below the input fields.

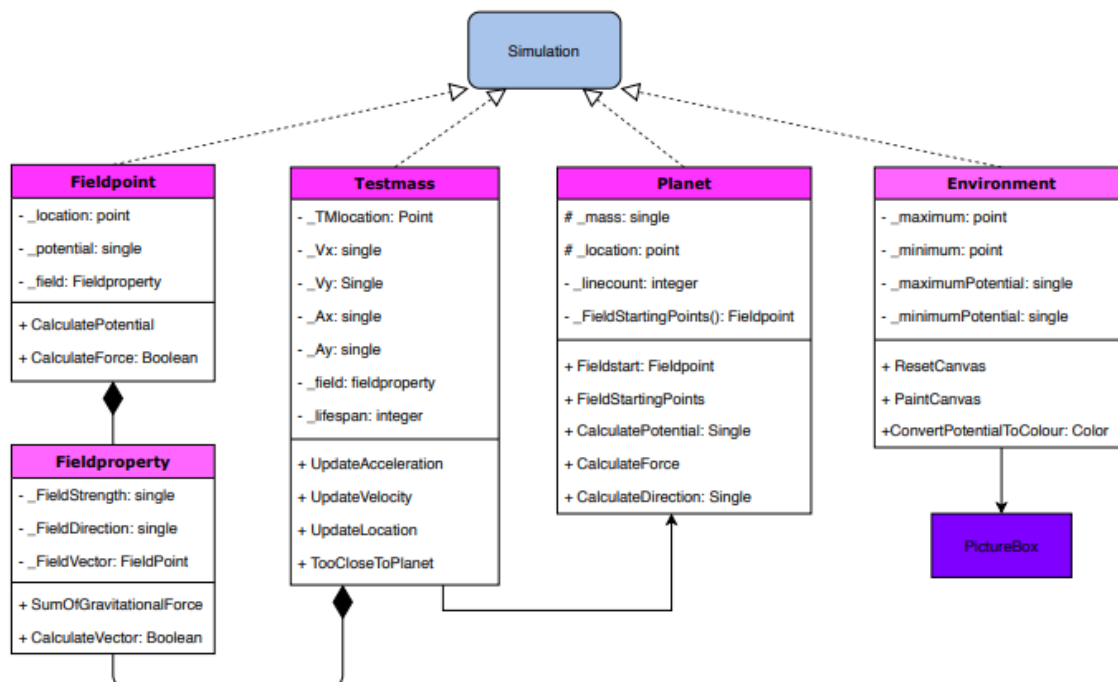
UML Class Diagram For the simulation

The simulation is a large aspect of the system as a whole and to ensure that it is operational and effective it must be programmed in a structured manner. The object oriented paradigm is ideal for this type of system as it enables multiple 'objects' to be created and destroyed which each independently store their variables.

In the situation it is logical that each mass (planet) placed in the system is its own object; as this enables multiple masses to be easily created each with the same methods which would simplify calculations later on.

A testmass is inherently similar to a mass, it being of constant mass 1 kg, while a placed mass should be able to take a variety of values. However a lot of the same methods will be required so the testmass class should inherit these from the planet class.

To aid and benefit calculations the fieldpoint and fieldproperty classes are required to simplify the calculations and processes needed to produce an accurate model.



The environment class is shown to inherit from the picturebox object. This is a included object within visual basic windows forms and enables visual representation. Here I am adapting the picturebox class to fulfil my needs.

The Fieldproperty class should never be able to exist by itself as it should always be linked to a point in a field. Here it can be seen that the fieldpoint class and the testmass class are composed of multiple fieldproperty so that the fieldproperty cannot exist without its 'parent' class.

Each class is implemented within the simulation to produce the desired outcome. While the simulation could be another class in the system it will be defined as a module where each of the separate classes are defined as it was deemed more appropriate for the system.

Algorithms

SUM OF GRAVITATIONAL FORCES

```
Procedure                               SumOfGravitationalForce(Force1Magnitude:Integer,
Force1Direction:Single, Force2Magnitude:Integer, Force2Direction:Single)
    HorizontalForce ← Force1Magnitude*cos(Force1Direction) +
Force2Magnitude*cos(Force2Direction)
    VerticalForce ← Force1Magnitude*sin(Force1Direction) +
Force2Magnitude*sin(Force2Direction)

    NewForceMagnitude ← sqrt(HorizontalForce^2 + verticalforce^2)
    Newforcedirection ← CalculateDirection
End Procedure
```

To sum two vector forces they first need to be broken down into their separate x and y components. Once broken down the different components can be summed together to find the resultant components, then using Pythagoras' theorem the magnitude of the resultant force can be found.

Finally to calculate the direction of the force the arctan of the vertical component over the horizontal component will return the angle from the horizontal. Once a correction has been applied; otherwise a force down and left will appear to have the same angle as a force heading up and right.

RECURSIVELY GENERATING FIELD LINES

```
Procedure CreateForceLine(Planets:Planet, CurrentPoint:Point)
    NewPoint ← CalculateNewPoint(Planets)
    If NewPoint <> CurrentPoint Then
        CreateForceLine(Planets, NewPoint)
    End If
End Procedure
```

The aim of the procedure is to create a field line by gradually stepping out from the mass. A set of starting points will be generated and stored as a variable within each mass. Then one at a time these points will be passed into the 'CreateForceLine' procedure.

Then the gravitational field at that point will be calculated. By keeping the magnitude of the field the same and reversing the direction a 'inverse force can be calculated'. If by using this force we step out to a new point, which if treated as a vector the force would have pointed to, the process can be repeated using this new point.

This will continue until the gravitational field strength is reduced to a point where the point passed in points to itself, meaning the cycle has ended.

COMPRESSION OF THE TEST MASS LIST

During the runtime of the program the test mass list will be populated with test masses, however test masses will be deleted if certain conditions are met which would leave blank values in the middle of the list.

So that it is clear where the next value should be added it is important to compress the list so that all the values are at the start of the list, and to keep note of where the next free index is.

```
Procedure    CompressList(MyList:List(of something),    NextIndex:Integer,
RemoveCount:Integer)
    If RemoveCount > 0 Then
        ShiftTo ← 0
        ShiftFrom ← 0
        While ShiftTo < NextIndex - RemoveCount
            While MyList(ShiftFrom) Is Nothing And ShiftFrom < MyList.Length
                ShiftFrom ← ShiftFrom + 1
            End While
            If ShiftTo <> ShiftFrom Then
                MyList(ShiftTo) = MyList(ShiftFrom)
                NextIndex ← NextIndex + 1
            End If
            ShiftTo ← ShiftTo + 1
            ShiftFrom ← ShiftFrom + 1
        End While
    End If
End Procedure
```

Each time a value is deleted within in the list then the remove count is incremented by 1 externally to this procedure. Therefore the system knows it needs to perform a compression.

By starting at the beginning of the list the program cycles through each index until it finds an 'empty' index incrementing both shift to and shift from. Once an 'empty' index has been found the program increments shift from until a 'full' index is found.

Then using the values stored in shift to and shift from it switches the full index into the position of the empty index. The list cycles through until all full indexes are consecutively at the start.

External Libraries and modules

1. **Imports** ADOX
2. **Imports** System.Data.OleDb

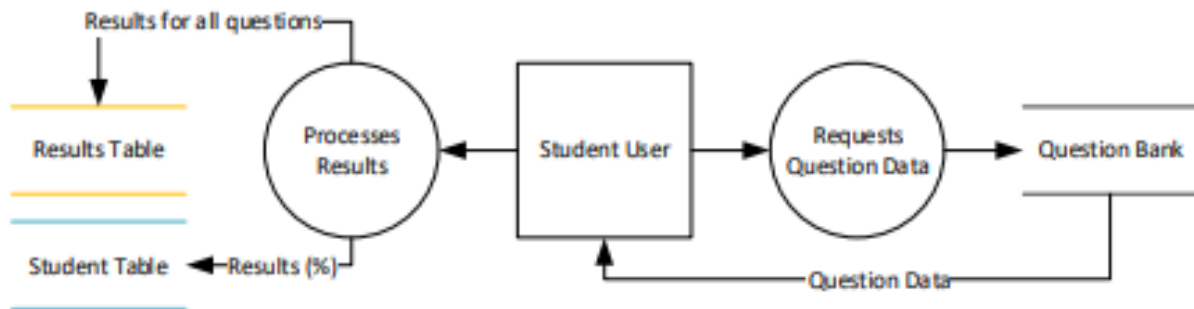
The ADOX and OLEDB libraries are necessary to enable the database system to work. The function of the ADOX library is expose additional objects for creating, modifying, and deleting schema objects; such as tables and procedures. It also includes security objects to maintain users and groups and to grant and revoke permissions on objects.

The OLEDB library allows me to interact with the SQL database, for my specific uses I am using the OleDbCommand, OleDbConnection, OleDbDataAdapter and OleDbDataReader classes which enable me to establish a connection to a SQL database, before writing, modifying or selecting data using a SQL command.

The data adapter and data reader classes respectively enable me to format tables of data or read specific rows of data.

Data Flow Diagrams

DATA FLOW WITH A STUDENT USER



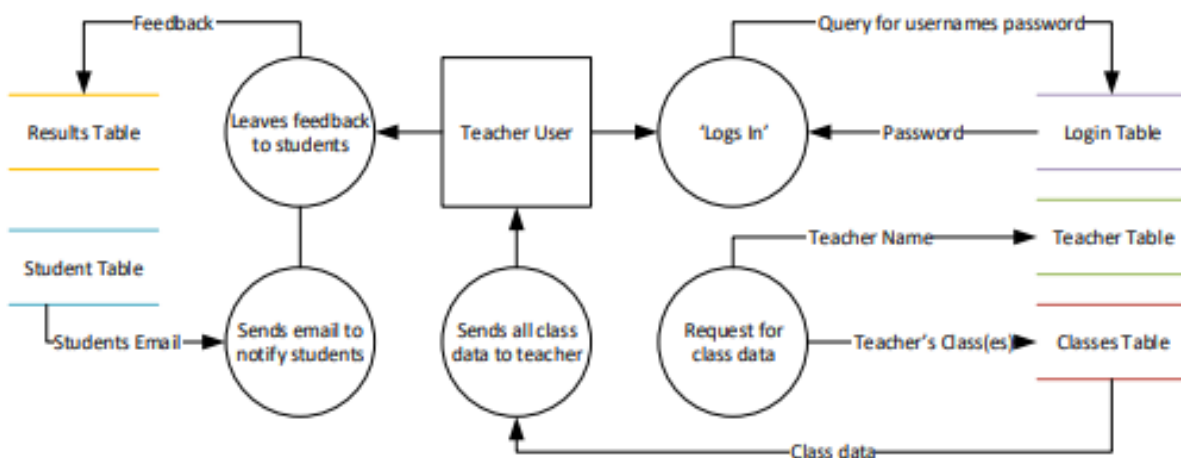
A student user will have access to the question and simulation aspects of the system. For a student's results to be stored they will have to have an account in the database. Therefore data will pass between the Username and Password storage and the user.

In order to answer the questions the student must have access to them and so access to the question bank will be necessary.

The student also needs to be linked to the class table, in order for this to occur the student should have to enter the unique class code. This requires that data flows between the classes table and the students table.

Finally once the student has finished the questions/quiz the full results will be stored in the results table, accessible by the student and their teacher.

DATA FLOW WITH A TEACHER USER



A teacher will also have an account in the system that will be linked to a number of classes, a one-to-many relationship. This means that the teacher will receive and send data to the username and password storage. So that the login system can work.

The teacher also needs access to their entire class data. In order for this to occur data will pass from the teacher table to the system so the teacher knows which classes the teacher is linked to. Then all the relevant data from the class table must be delivered to the teacher user.

A teacher user should be able to leave feedback for the students. This feedback will be stored in the results table and a student should be notified when feedback has been left. To notify a student access to the students table – where the student’s email is stored – will be important.

Data Dictionaries

STUDENTS TABLE

Data Name	Data Type
<u>Student Code</u>	Integer
Forename	String
Surname	String
Email	String
Password	String
Date Joined	Date

TEACHER TABLE

Data Name	Data Type
<u>Teacher Code</u>	Integer
Forename	String
Surname	String
Email	String
Password	String

CLASSES TABLE

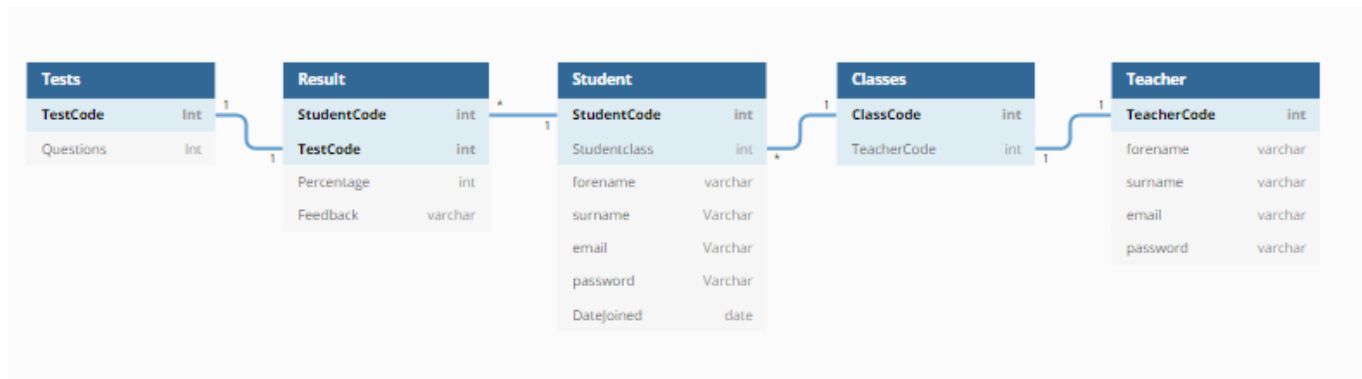
Data Name	Data Type
<u>Class Code</u>	Integer
TeacherCode	String

RESULTS TABLE

Data Name	Data Type
<u>Homework Code</u>	Integer
<u>Student Code</u>	Integer
Percentage	Integer
Feedback	String

Database Design

ENTITY RELATIONSHIP DIAGRAM



DDL STRING FOR DATABASE

Firstly the DDL string to create the Student table must create fields for student's code, class, forename and surname, along with their email, a password and the date they joined the system. The primary key must be the student code and each field should have the correct data type. Therefore the DDL string will look like this:

```
"CREATE TABLE [Students] ([StudentCode] INT CONSTRAINT PrimaryKey PRIMARY KEY, [StudentClass] INT, [Forename] Varchar(25), [Surname] Varchar(25), [Email] Varchar(25), [Password] Varchar(25), [DateJoined] DATE)"
```

Then the teacher table must have the primary key TeacherCode, and fields to include a user's forename, surname, email and password. The teacher table DDL will be:

```
"CREATE TABLE [Teachers] ([TeacherCode] INT CONSTRAINT PrimaryKey PRIMARY KEY, [Forename] Varchar(25), [Surname] Varchar(25), [Email] Varchar(25), [Password] Varchar(25))"
```

Then the Classes link table should be created with the classcode as the primary key and also contain a field for the teacher code so that each class links to 1 teacher.

```
"CREATE TABLE [Classes] ([Classcode] INT CONSTRAINT PrimaryKey PRIMARY KEY, [TeacherCode] INT)"
```

The results table must have a composite key to link the students and their results for each homework. This will be a combination of the student's code and the Test code.

```
"CREATE TABLE [Results] ([StudentCode] INT CONSTRAINT PrimaryKey PRIMARY KEY, [TestCode] INT CONSTRAINT PrimaryKey PRIMARY KEY, [Percentage] INT, [Feedback] Varchar)"
```

Finally the tests table must have an entry for each test, and hold the number of questions a test contains:

```
"CREATE TABLE [Tests] ([TestCode] INT CONSTRAINT PrimaryKey PRIMARY KEY, [Questions] INT)"
```

SQL STRING FOR QUERIES

In order for the login system to work the program must be able to gather all username and password pairs from the database; the SQL query for this will look like:

```
"SELECT UserCode, Password FROM Students"
```

```
"SELECT TeacherCode, Password FROM Teachers"
```

When a student user accesses their account page then they should be able to change their Class code, however they should first be able to view their current class code, the SQL query for this should be:

```
"SELECT StudentClass FROM Students WHERE Students.StudentCode = @condition"
```

When the user inputs the Class they want to be a part of it needs no be included in the database Student table under StudentClass, to accomplish this a suitable SQL query would be:

```
"UPDATE Students SET [StudentClass] = @data WHERE [StudentCode] = @condition"
```

The results system shall require the Testcode, Percentage and Feedback fields from the results table, however for a student it should only return the particular students results. While a teacher should be able to access their relevant classes results. Therefore the SQL Query would look like:

For the students:

```
"SELECT HomeworkCode, Percentage, Feedback FROM Results WHERE Results.StudentCode = @condition"
```

For the teachers the system should present a dropdown menu with all the teachers classes, allowing them to select one, and then select an individual student or an individual test to review the students individual scores or the classes scores on the test. This should be straightforward and easy to use. To retrieve these lists of data the following queries are appropriate:

```
"SELECT Students.Forename, Students.Surname FROM ((Students INNER JOIN Classes ON Students.StudentClass = Classes.ClassCode) INNER JOIN Teachers ON Classes.TeacherCode = Teachers.TeacherCode)"
```

or to gather the testcodes of each test in the database:

```
"SELECT TestCode FROM Tests"
```

Once these fields have been gathered the user is required to check a single checkbox and then press the query button, and the system should gather all the data relevant to the selected case.

Case 1: Viewing a single students results

Here to gather a single students test results the system must have the students name to apply the condition to the SQL:

```
"SELECT Results.HomeworkCode, Results.Percentage, Results.Feedback FROM Students INNER JOIN Results ON Students.StudentCode = Results.StudentCode WHERE Students.Forename = @forename AND Students.Surname = @surname;"
```

The students name may be gathered from the selected dropdown list item as that is the student whose results the teacher has asked for, the above query will return the desired result when a students forename is passed into the @forename and their surname into the @surname.

Case 2: Viewing a single tests results

To view all the results of a single test completed by everyone in a teachers class the testcode and the classcode of the teacher must be known.

```
"SELECT Students.Forename, Students.Surname, Results.Percentage, Results.Feedback
FROM ((Students INNER JOIN Results ON Students.StudentCode = Results.StudentCode) INNER
JOIN Tests ON Results.HomeworkCode = Tests.TestCode) INNER JOIN (Classes INNER JOIN
Teachers ON Classes.TeacherCode = Teachers.TeacherCode) ON Students.StudentClass =
Classes.Classcode
WHERE Teachers.TeacherCode = @teachercode AND Tests.TestCode = @testcode"
```

The students forename, surname, percentage on the test and their individual feedback for a single test will be retrieved by the Select SQL statement, which required that the student is part of the teachers class. The teachers code is passed into the @teachercode and the desired testcode is passed into @testcode

A teacher should also be able to add feedback onto a students results. By using a event handler the system can tell when a cell in the results grid has experienced a change in value, and by filtering through the columns it is easy to validate that the feedback has been changed.

Then by gathering the testcode and the students name from either the dropdown lists or the same row of the results grid the following SQL condition can be met.

```
"UPDATE ((Tests INNER JOIN Results On Tests.TestCode = Results.HomeworkCode) INNER JOIN
Students On Results.StudentCode = Students.StudentCode) Set Results.Feedback = '' &
feedback &'' WHERE Tests.TestCode = @testcode And Students.Forename = @forename And
Students.Surname = @surname"
```

REGEX EXPRESSIONS FOR VALIDATING STRINGS

When registering an account the user is asked to input their forename, surname, email and two matching passwords. A simple way to validate the entrees is using regular expressions to check the entered characters.

The entered forename and surname should be strings that only contain upper and lower case letters. The regular expression to validate this would be:

```
"^[A-Za-z]+$".
```

Which returns true when passed through the Regex.IsMatch function (with the string to be tested) if all the characters in the string are upper or lower case letters.

The email is must contain a domain name and address, and should be in the format: [first@second.third](#). In order to test for this:

```
"^([0-9a-zA-Z]([-\.\\w]*[0-9a-zA-Z])*)@([0-9a-zA-Z]([-\\w]*[0-9a-zA-Z]\.)+[a-zA-Z]{2,9})$"
```

Which returns false if the string does not contain a combination of letters or dots, followed by an @ sign and then followed by a combination of letters and dots. Enabling the system to quickly test the validity of an email address.

The password does not need a validation using regex as it can contain any character, and must simply match the other entered password. It could easily be appended so that there is a minimum length that the password can be.

Finally when the teacher is adding a test results the system should only accept a string which contains only numbers.

Which is simply:

```
"^[0-9]+$"
```

This will aid in robustness as it will prevent crashes when converting to an integer.

Alpha Testing Strategy

Testing outline

Test	Description	Purpose
1.	Input Testing	Validates inputs into the system using regular expressions and data validation (bottom-up testing)
2.	Flow of control testing	Checks the users path through the system, they should be forced to log in to access certain elements. (Top-down testing)
3.	Process Testing	Ensures the used algorithms produce the correct outputs. (black box testing)
4.	Testing saving of data	Ensure all data must be saved in the correct location and in the correct data type.
5.	Specification testing	Used to check whether the system meets the original specification set out in the analysis/requirements

Input Testing Strategy

INPUT TESTING DATASETS

Typical datasets (T), Erroneous Datasets (E), Extreme Datasets (X)

Registering and Login Dataset: (for both students and teachers)

Data set	Usercode	Password	Email	Forename	Surname
Treg1	10000	ABcd1234	LBJ@myemail.com	James	Labpartner
Ereg1	10001	Ab1234	Heresmyemail@myemail.com	ThisIsMyNAME	AwKWArD
Xreg1	Nope	Word	Notanemail	N^M3	WRONG

Classcode Dataset

Data set	Classcode
Treg1	1
Ereg1	124124
Xreg1	1a

INPUT TESTING TABLE

No.	Purpose	Description	Data Typical, Erroneous, Extreme	Expected outcome	Actual outcome	Evidence in appendix
1.1	Validate Password	Any combination of characters longer than 6 should be accepted	Treg1 – "Abcd1234" Ereg1 – "Ab1234" Xreg1 – "Word"	Accept Accept Error		
1.2	Validate Email	Email should be accepted if it contains an @ sign and a domain	Treg1 – "LBJ@myemail.com" Ereg1 – "Heresmyemail@email.com" Xreg1 – " Notanemail"	Accept Accept Error		
1.3	Validate Forename	Forename should be accepted if it only contains letters.	Treg1 – "James" Ereg1 – "James" Xreg1 – "James12"	Accept Accept Error		
1.4	Validate Surname	Surname should be accepted if it only contains letters	Treg1 – "Hancock" Ereg1 – "Hancock" Xreg1 – "Hancock12"	Accept Accept Error		
2.1	Validate Classcode	Classcode must be of the correct format i.e an integer	Treg2 – 1 Ereg2 – 124124 Xreg2 – 1a	Accept Accept Error		

Flow of Control Testing Strategy

FLOW CONTROL WITHIN SIMULATION

- In simulation, disabling of buttons

In order to be a robust simulation that is difficult to cause errors in there must be some controlling of flow to prevent certain actions.

While most key processes will still function regardless of the order of operation a desired route through the system would be:

1. The user chooses the locations and relative masses of the systems masses.
2. The user then chooses one of the options from the list of functions.
3. The function then executes and control is brought back to the user after the process has been completed.

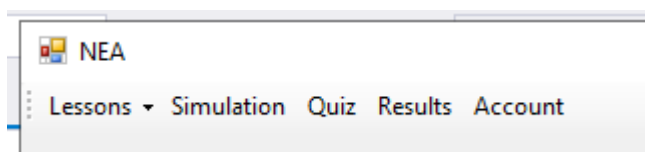
To control flow here and improve robustness the technique of limiting the users actions has been used. For example once the add mass button has been clicked all other controls on the interface excluding clear are disabled so that the user cannot continue to add masses while a process is running. This aids in preventing system crashed and creating false results where the environment was changed after the defining variables were calculated.

This should be evident within the [testing video](#) as windows forms changes the view of controls when they are disabled.

FLOW CONTROL WITHIN ACCOUNT USE

Pre-login

On the home form there are multiple options available to the user, using the taskbar class the user can select from several options of where to navigate the system too.



While a user is not logged in they should be prohibited from accessing the results page, so an error message should be displayed asking the user to log in.

By selecting the account page it should bring up a separate form which enables the user to log in. In this second form the user will either be able to log in (as a student or a teacher depending on the selected option, and the login details are validated), or access the register page.

Once the user has elected to log in to their account or to create a new account the system should return them to the home page. However the user should still be logged in so that they can now access the results page.

Therefore variables containing the account details as well as a Boolean stating that there is a user logged in must be passed variables to retain the data whilst changing form.

Case 1: a student with the username 10000 has logged in the following variables are expected to contain the specified values:

No.	Variable	Description	Expected value	Actual value	Evidence in appendix
1.0	LoggedIn	Boolean variable that states whether a user is currently logged in	True		
1.1	Usercode	String that contains the users code, not dependent on whether or not the user is a student or teacher	10000		
1.2	PriveledgeLevel	Innumerable which defines what level of access the user has	Userlevel.Student		

Case 2: a teacher with the username 1000 has logged in:

No.	Variable	Description	Expected value	Actual value	Evidence in appendix
1.0	LoggedIn	Boolean variable that states whether a user is currently logged in	True		
1.1	Usercode	String that contains the users code, not dependent on whether or not the user is a student or teacher	1000		
1.2	PriveledgeLevel	Innumerable which defines what level of access the user has	Userlevel.Teacher		

Post-login

Once a user has logged in the account page should change, and for a student user should display and allow them to alter their Classcode.

The results page should also now be accessible to all levels of user but should display varying interfaces depending on who is logged in.

These changes will be apparent during the [testing video](#).

Process Testing Strategy

Algorithmic processes

The key algorithms responsible for generating the data required to produce the images in the simulation must be working correctly in order to fulfil the requirements. Some of the key algorithms are:

- The calculation of gravitational potential
- The calculation of gravitational force
- The summation of individual forces
- The recursive elements within the generate field line function
- The Compression of the test mass list

The calculation of gravitational potential

Incredibly important for the system as many of the processes within the simulation rely on the basic calculation of gravitational potential.

This is almost impossible to gather during normal runtime so a test will be run using the following variables.

Property	Value	Symbol
Mass	5.97×10^{24} kg	M
Radius	6.37×10^6 m	r
Gravitational Constant	6.67×10^{-11} Nm ² kg ⁻²	G

In order to calculate the gravitational potential, V_g , at a point in a gravitational field the following equation is used:

$$V_g = - \frac{GM}{r}$$

$$V_g = - \frac{(6.67 * 10^{-11})(5.97 * 10^{24})}{(6.37 * 10^6)}$$

$$V_g = - 6.26 * 10^6 \text{ Joules}$$

These values should produce the mean gravitational potential at the earths surface which according to https://en.wikipedia.org/wiki/Gravitational_potential is approx. equal to -6MJ.

These values shall be passed into the calculate potential function and the value return should, if correct, be accurate to the calculated value of V_g .

[View appendix](#)

Gravitational potential is a scalar property which means that when summing them it is a simple case of adding the magnitudes of the potentials from each body in the system. (as discussed in the analysis)

The calculation of gravitational force

Another fundamental process in the system is the calculation of the gravitational force F , at any point within the gravitational field, like before a runtime test within the simulation is challenging so a test of the subroutine with the following values will suffice:

Property	Value	Symbol
Mass 1	5.97×10^{24} kg	m_1
Mass 2	1 kg	m_2
Radius	6.37×10^6 m	r
Gravitational Constant	6.67×10^{-11} Nm ² kg ⁻²	G

As discussed within the analysis the equation for calculating the gravitational force, F , at a point within a field is equal to:

$$F = G \frac{m_1 m_2}{r^2}$$

$$F = (6.67 * 10^{-11}) \frac{(5.97 * 10^{24})(1)}{(6.37 * 10^6)^2}$$

$$F = G \frac{m_1 m_2}{r^2}$$

$$F = 9.81 N$$

These values should produce the mean gravitational force at the earths surface which according to https://en.wikipedia.org/wiki/Gravitational_potential is approx. equal to 9.81N

These values shall be passed into the calculate potential function and the value return should, if correct, be accurate to the calculated value of V_g .

[View appendix](#)

The summation of forces

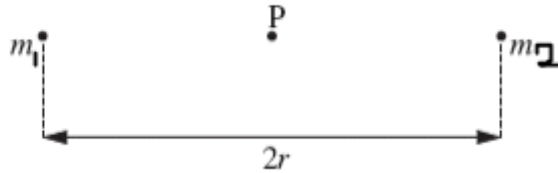
The algorithm responsible for summing together the individual forces acting upon a test mass from each massive body in the system is utilised at multiple points during a simulations lifetime, for example within the show force lines and test mass utilities, therefore it is essential that the correct outputs are received.

Unlike gravitational potential, the gravitational force is a vector property and has direction, which for a single mass system will always point towards the body in question. This simple fact makes the summation of gravitational forces a more complex challenge.

Therefore the resultant force, F_R , of the individual forces will be equal to the vector sum of each force:

$$\vec{F}_R = \sum \vec{F}$$

In order to test this two massive bodies will be passed into the system at specific predefined radius from each mass. For simplicity this will be shown by using two masses that are on the same horizontal line.



In this example point P is in between m_1 and m_2

Property	Value	Symbol
Mass 1	1×10^6 kg	m_1
Mass 2	2×10^6 kg	m_2
Test mass	1 kg	m_t
Radius	1×10^3 m	r
Gravitational Constant	6.67×10^{-11} Nm ² kg ⁻²	G

Therefore to calculate the results gravitational force, F_R , at P:

$$\vec{F}_R = \sum \vec{F}$$

Taking the direction to the right as positive in this example, as P is in the midpoint the denominator remains as r^2 , otherwise the sum of the two r values would equal the distance between the masses.

$$\vec{F}_R = G \frac{(2 * 10^6)(1)}{(1 * 10^3)^2} - G \frac{(1 * 10^6)(1)}{(1 * 10^3)^2}$$

$$\vec{F}_R = G(1) = 6.67 \times 10^{-11} \text{ N}$$

Note the direction of the force should be to the right.

[View appendix](#)

There is also an element of recursion within this algorithm which enables the system to sum individual forces until a stack overflow error occurs. A point which is unlikely to be encountered as it would require the user to place that number of masses. Therefore if the algorithm works for two massive bodies then it should work for N bodies, as the resultant force of two forces can be continuously found until the final resultant force is discovered.

Recursive elements within the generate field line function

The generate field line function works by calculating the resultant gravitational field strength at a point within a gravitational field; by inverting the direction of the force and by scaling its magnitude it can point to a new location where the process can be repeated.

To prevent a stack overflow a termination condition has to be met, where the force has become so weak that applying the process returns the same point. At this point the line will not be extended so there is no need to continue running the program and the stack can be emptied.

An alternate condition where the force 'vector' points to a location that is not contained within the canvas. At this point there is no need to keep running the subroutine and the stack may terminate.

Evidence for the recursive elements and termination of the stack can be found in the [appendix](#)

Compression of the test mass list

The test mass list contains 100 memory locations, and with a new testmass being instantiated every 0.5 seconds (base rate) the list would fill very quickly and cause a fault. However to prevent over populating the canvas with test masses the testmasses have a limited lifespan and will 'decay' after a certain number of life cycles. A testmass will also decay if it comes into contact with a mass.

As a result of this decay process the array of testmasses will slowly be converted until a point where each index contains no value. To remedy this the algorithm replaces all indexes where the value is nothing with the next index that contains a value. Compacting the list to ensure all the existing testmasses are stored at the beginning of the array.

This algorithm will be stepped through within the [testing video](#) to ensure it is fully functional.

Visual processes

Firstly the simulation should be robust and any errors should be encapsulated and handled. This is important as if it to be useable and beneficial it must be robust. This can be testing completely through use and by attempting different combinations of actions.

As the output of the simulation is generally visual the expected outcomes will also be visual. By abstracting the program into its separate parts the expected outcomes can be shown.

Potentials function

The potentials function should produce a colourmap of the potentials surrounding two massive objects. This should be visible by concentric rings of colour surrounding each mass, when multi body systems are produced the program should produce images that are similar to the following:

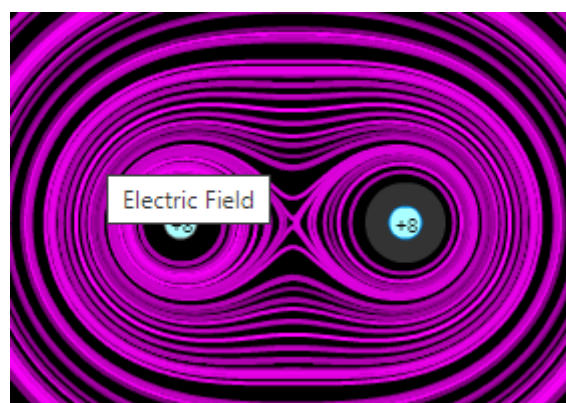


Figure 20: Potentials Example 1

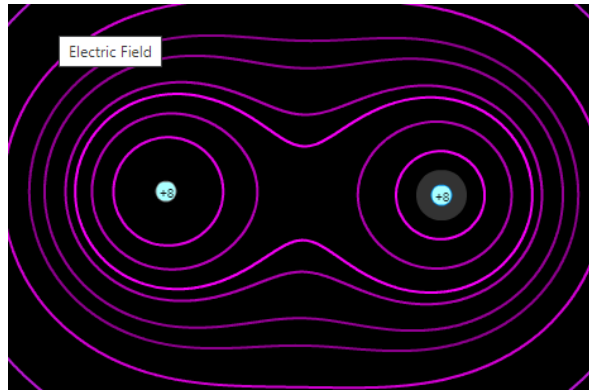


Figure 21: Potentials example 2

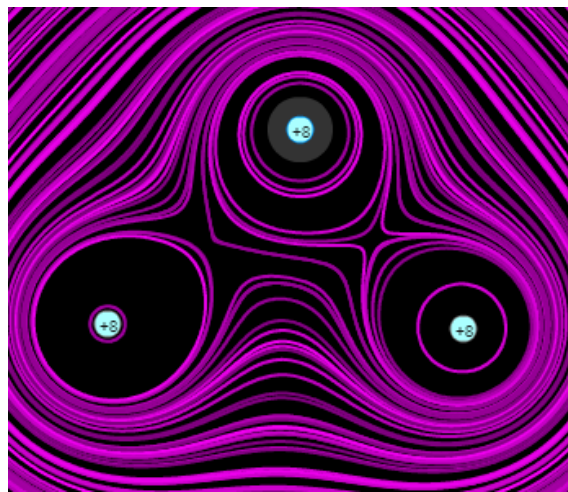


Figure 22: Potentials Example 3

Field Line function

Next the field line function should produce a set of lines that originate at the mass and extend outwards producing a diagram that looks like the following:

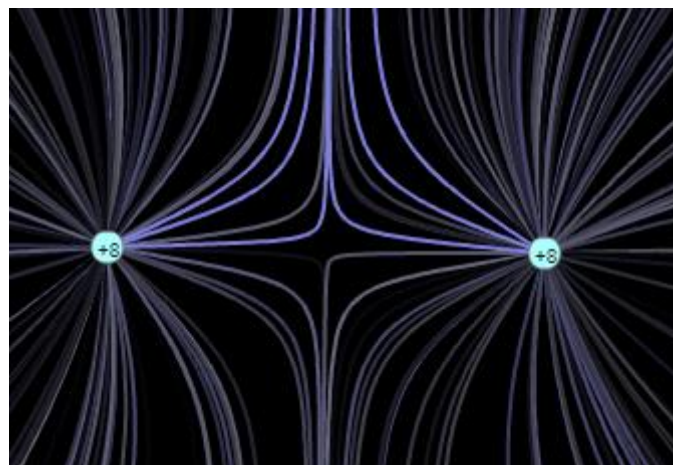


Figure 23: Field Line Example 1

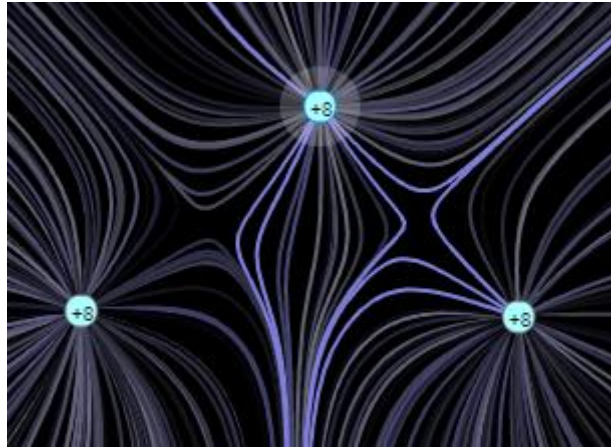


Figure 24: Field Line Example 2

Test mass function

The test mass function should produce a set number of test masses in a set time interval which should show the direction of the gravitational force. They should be affected by all masses in the system but should not affect any of the large masses in the system as their weight is assumed to be negligible in comparison. A test mass should be deleted and no longer shown once it comes into contact with a mass, or it lasts longer than its pre-set lifespan, to prevent multiple test masses becoming permanently stuck in Lagrange points (where the resultant force acting upon the test mass is zero newtons).

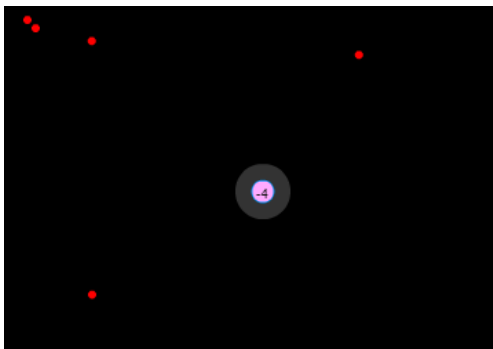


Figure 25: Test mass example

Here the red dots represent the test masses, and will be attracted to each mass in the system.

To prevent overpopulating the system with test masses they should 'decay' after a pre-determined lifespan or if they collide with a mass in the system. This should be visually apparent when running the test mass function during the [testing video](#).

Storage Testing Strategy

DATABASE CREATION

Firstly, it is imperative that the database is created correctly with the correct tables, fields and datatypes, as it has been created by multiple DDL strings contained within the program itself. Each table should be created as per specified in the design.

Students Table

Student	
StudentCode	int
Studentclass	int
forename	varchar
surname	Varchar
email	Varchar
password	Varchar
Datejoined	date

Teachers Table

Teacher	
TeacherCode	int
forename	varchar
surname	varchar
email	varchar
password	varchar

Classes Table

Classes	
ClassCode	int
TeacherCode	int

Results Table

Result	
StudentCode	int
TestCode	int
Percentage	int
Feedback	varchar

Tests Table

Tests	
TestCode	Int
Questions	Int

Table	Evidence in appendix
Students	3.1
Teachers	3.2
Results	3.3
Classes	3.4
Tests	3.5

INSERTING INTO THE DATABASE

There are multiple occurrences when data is inserted into the database:

1. When a new user registers (Teacher or student)
2. When a student adds or changes their class code
3. When a Teacher adds feedback to a students work

User registration

When a student or teacher registers they are required to enter multiple fields such as their forename, surname and email. All this data must be entered into the database correctly

In the case of a student:

A Usercode is generated once valid login details have been entered and is input along with all the other information

Criteria	Expected Field	Expected Data type	Actual Field	Actual Data type	Evidence in appendix
Usercode	Students.StudentCode	Integer			4.1
Forename	Students.forename	Varchar			4.1
Surname	Students.Surname	Varchar			4.1
Email	Students.Email	Varchar			4.1
Password	Students.password	Varchar			4.1
Date Joined	Students.DataJoined	Date			4.1

In the case of a Teacher

Once a teachers account is created a class code is automatically generated however this value is stored within the Classes table.

Criteria	Expected Field	Expected Data type	Actual Field	Actual Data type	Evidence in appendix
Usercode	Teachers.TeacherCode	Integer			4.2
Usercode	Classes.TeacherCode	Integer			4.2
Forename	Teachers.forename	Varchar			4.2
Surname	Teachers.Surname	Varchar			4.2
Email	Teachers.Email	Varchar			4.2
Password	Teachers.password	Varchar			4.2
ClassCode	Classes.Classcode	Integer			4.3

Appending Class Code

A student should be able to add the class code that connects them to their teacher. This value is stored within the student table under the student class field

Criteria	Expected Field	Expected Data type	Actual Field	Actual Data type	Evidence in appendix
Classcode	Students.StudentClass	Integer			

This is also required to be in the correct students table, as otherwise the student would be connecting a different student to his class.

To efficiently test it the following students details will be used:

Property	Value
StudentCode	10000
Forename	"Alfie"
Surname	"Aldwin"

Expected Student (By studentcode)	Actual Student	Evidence in appendix
10000		

Adding/Altering feedback

A teacher is able to add feedback to all their students work. For this test the teacher will be adding the following to the student "Alfie Aldwin" where originally there is no feedback.

Expected StudentCode and testcode	Expected Feedback	Expected Data Type	Actual StudentCode and testcode	Actual Feedback	Actual Data Type	Evidence in appendix
10000, 1	"Great Work"	Varchar				

Evidence for these processes will be demonstrated within a [testing video](#).

Beta Testing Strategy

To gather a response on the overall effectiveness of the developed program it is important to receive an opinion and feedback from the target audience in order to determine its successfulness as a product.

To conduct the beta testing I will give the following questionnaires to students and teachers who have been given a chance to explore the programmes facilities. In order to show of the effectiveness of the data handling aspect of the program both audiences will be given a role specific account to use.

QUESTIONNAIRE FOR STUDENTS

1. Did you encounter any issues with the product?

Yes No

Please outline if yes,

2. To what extent was the product easy to use?

Poor Below Average Average Above Average Strong

3. How beneficial was the simulation to your understanding of gravitational potential?

Not beneficial not useful useful Very beneficial

4. Would you like to see as resource such as this be used by teachers within lessons?

Yes No

5. How easy were the accounts and results page to use?

Hard Below Average Average Above Average Easy

6. How suitable is the range of questions

Not suitable Could be improved Average Suitable

7. What improvements, if any, would you like to see implemented?

QUESTIONNAIRE FOR TEACHERS

1. Did you encounter any issues with the product?

Yes No

Please outline if yes,

2. To what extent was the product easy to use?

Poor Below Average Average Above Average Strong

3. How beneficial was the simulation to your students understanding of gravitational potential?

Not beneficial Semi-useful Very beneficial

4. Would you use a resource such as this within lessons?

Yes No

5. How easy were the accounts and results page to use?

Hard Below Average Average Above Average Easy

6. How easy was it to access a students results and find the desired information?

Hard Below Average Average Above Average Easy

7. How suitable is the range of questions

Not suitable Could be improved Average Suitable

8. What improvements, if any, would you like to see implemented?

Technical Solution

Home Form

The form used for the interface of the system. Everything in the program relies on this form so it is the backbone of the program. However not a lot of code is held here due to the large scope of the system. Note the database is originally created here.

```
1. Option Strict On
2. Public Class Home
3.     Public LoggedIn As Boolean = False
4.     Public PriveledgeLevel As Login_System.UserLevel
5.     Public UserCode As String
6.
7.     Private Sub Home_Load(sender As Object, e As EventArgs) Handles MyBase.Load
8.         'Database.DatabaseMasterCreator(Database.DatabaseName) 'used during the original
           creation of the database
9.         Me.WindowState = FormWindowState.Maximized 'full screen
10.        Me.Text = "NEA"
11.        CreateToolbar(Me) 'creates the toolbar system
12.    End Sub
13. End Class
```

Layout Creation

As multiple forms are reused for different purposes it is necessary that buttons and textboxes, amongst other things, are created programmatically, and can be recreated at any time. For organisation these are all stored within a module, aptly name layout creation. Most forms have a call to a procedure within this module at some point.

```
1. 'Option Strict On
2. Module LayoutCreation
3.     Dim WithEvents Toolbar As New ToolStrip
4.     Dim WithEvents ToolsLessons As New ToolStripDropDownButton
5.     Dim WithEvents ToolsSim As New ToolStripButton
6.     Dim WithEvents ToolsQuiz As New ToolStripButton
7.     Dim WithEvents ToolsResults As New ToolStripButton
8.     Dim WithEvents ToolsAccount As New ToolStripButton
9.
10.
11.    Public Sub CreateToolbar(Domain As Object)
12.        CreateLessonsToolBar(ToolsLessons) 'creates the drop down section of t
           he toolbar, which is a separate type of control
13.        Toolbar.Items.Add(ToolsLessons)
14.
15.        CreateDropDownButton(Toolbar, ToolsSim, "Simulation") 'creating each indi
           vidual toolbar button
16.        CreateDropDownButton(Toolbar, ToolsQuiz, "Tests")
17.        CreateDropDownButton(Toolbar, ToolsResults, "Results")
18.        CreateDropDownButton(Toolbar, ToolsAccount, "Account")
19.        For Each Item As ToolStripItem In Toolbar.Items 'adds the handler to al
           l the separate buttons so that the each fulfil their purpose
20.            If TypeOf Item Is ToolStripButton Then
21.                AddHandler Item.Click, AddressOf Navigation.ButtonDestination
```



```

22.         End If
23.     Next
24.     For Each MenuItem As ToolStripMenuItem In ToolsLessons.DropDownItems
25.         If TypeOf MenuItem Is ToolStripMenuItem Then
26.             AddHandler MenuItem.Click, AddressOf Navigation.ListDestination
27.         End If
28.     Next
29.     Domain.controls.add(Toolbar)
30. End Sub
31. Private Sub CreateLessonsToolBar(ByRef DropDown As ToolStripDropDownButton)
32.     If DropDown.DropDownItems.Count = 0 Then 'prevents items being added mul
multiple times after form remake
33.         DropDown.DisplayStyle = ToolStripItemDisplayStyle.ImageAndText
34.         DropDown.Text = "Lessons"
35.
36.         Dim workandenergy As New ToolStripMenuItem
37.         workandenergy.Text = "1. Work and Energy"
38.         Dim gravfieldsandforces As New ToolStripMenuItem
39.         gravfieldsandforces.Text = "2. Gravitational Fields and Forces"
40.         Dim gravpotential As New ToolStripMenuItem
41.         gravpotential.Text = "3. Gravitational Potential Energy and Gravitation
al Potential"
42.         Dim potentialgradandescapevel As New ToolStripMenuItem
43.         potentialgradandescapevel.Text = "4. Potential gradients and escape vel
ocity"
44.
45.         DropDown.DropDownItems.Add(workandenergy)
46.         DropDown.DropDownItems.Add(gravfieldsandforces)
47.         DropDown.DropDownItems.Add(gravpotential)
48.         DropDown.DropDownItems.Add(potentialgradandescapevel)
49.     End If
50.
51. End Sub
52. Private Sub CreateDropDownButton(ByRef toolbar As Object, ByRef button As Objec
t, ByVal text As String)
53.     button.DisplayStyle = ToolStripItemDisplayStyle.ImageAndText
54.     'ToolsSim.Image = System.Drawing.Bitmap()
55.     button.Text = text
56.     toolbar.Items.Add(button)
57. End Sub
58. Private Sub CreateButton(ByRef but As Button, ByVal location As Point, ByVal si
ze As Size, ByVal text As String)
59.     but.Location = location
60.     but.Size = size
61.     but.Text = text
62. End Sub
63. Public Sub CreateAccountPage()
64.     'create account page
65.     CreateToolBar(Home)
66.     'Home.Text = "Account"
67.     If Home.LoggedIn = False Then 'the user has to log in to access the a
ccounts page
68.         Login_System.Login_Load()
69.         Login_System.Show() 'brings up the login form
70.     Else
71.         If Home.PriveledgeLevel = Login_System.UserLevel.Teacher Then 'if the
user is a teacher
72.             'teacher account page#
73.             Account.LoadTeacherAccount() 'passes the variables through to th
e home form
74.             Account.Show()
75.         Else
76.             Account.LoadStudentAccount() 'passes the variables through to th
e home form
77.             Account.Show()
78.

```

```

79.         End If
80.     End If
81. End Sub
82. Public Sub CreateTestPage()
83.     'create quiz page
84.     CreateToolbar(Home)
85.     Home.Text = "Quiz"
86.     If Home.PriveledgeLevel = Login_System.UserLevel.Teacher Then 'if the use
r is a teacher
87.         Tests.Show()
88.     Else
89.         'get link to tests?
90.         System.Diagnostics.Process.Start("https://drive.google.com/open?id=1cV6
cAqBaLvcIbbn785K8pSI4fZSZA4Vv") 'opens the website for all the saved tests
91.     End If
92. End Sub
93. Public Sub CreateResultsPage()
94.     CreateToolbar(Home)
95.     If Not Home.LoggedIn Then
96.         MsgBox("You must be logged in to access the results page")
97.     Else
98.         Results.PassVariables(Home.UserCode, Home.PriveledgeLevel)
99.         Home.Text = "Results"
100.         If Home.PriveledgeLevel = Login_System.UserLevel.Teacher Then
101.             'MsgBox("Teacher Results")
102.             Results.FormatTeacherPage() 'creates the results page for a
teacher user.
103.         Else
104.             'MsgBox("Student Results")
105.             Results.FormatStudentPage()
106.         End If
107.     End If
108. End Sub
109. Public Sub CreateSimulationPage()
110.     CreateToolbar(Home)
111.     Home.Text = "Simulation"
112.     Simulation.CreateCanvas()
113. End Sub
114. Public Sub Simulationcontrols(ByRef controls() As Button, ByRef address
As Button, ByRef SliderValue As Label, ByRef Slider As TrackBar, ByRef Instructions
As TextBox)
115.     Dim str As String = ""
116.     For index As Integer = 0 To controls.Count - 1
117.         controls(index) = New Button
118.
119.         Select Case index
120.             Case 0
121.                 str = "Clear"
122.                 AddHandler controls(0).Click, AddressOf Simulation.Clear
_click
123.             Case 1
124.                 str = "Calculate Potentials"
125.                 AddHandler controls(1).Click, AddressOf Simulation.Poten
tials_click
126.             Case 2
127.                 str = "Show Force lines"
128.                 AddHandler controls(2).Click, AddressOf Simulation.Force
Lines_click
129.             Case 3
130.                 str = "Add Test mass"
131.                 AddHandler controls(3).Click, AddressOf Simulation.TestM
ass_click
132.             Case 4
133.                 str = "Measure"
134.                 AddHandler controls(4).Click, AddressOf Simulation.Measu
re_click

```

```

135.             Case Else
136.                 str = "<invalid option>"
137.             End Select
138.             CreateButton(controls(index), New Point(850, 50 + 30 * index), N
new Size(400, 25), str)
139.             Home.Controls.Add(controls(index))
140.         Next
141.
142.         'dim slider as new trackbar
143.         Slider.Location = New Point(950, 300)
144.         Slider.Size = New Size(300, 50)
145.         Slider.Minimum = 1
146.         Slider.Maximum = 10
147.         Slider.SmallChange = 1
148.         Home.Controls.Add(Slider)
149.         AddHandler Slider.ValueChanged, AddressOf Slider_Change
150.
151.         addmass = New Button
152.         addmass.Location = New Point(850, 300)
153.         addmass.Size = New Size(100, 25)
154.         addmass.Text = "Add Mass"
155.         Home.Controls.Add(addmass)
156.
157.         AddHandler addmass.Click, AddressOf Addmass_Click
158.         SliderValue.Location = New Point(1255, 301)
159.         SliderValue.Text = Slider.Value
160.         Home.Controls.Add(SliderValue)
161.
162.         Instructions = New TextBox
163.         With Instructions
164.             .Multiline = True
165.             .Size = New Size(400, 130)
166.             .Location = New Point(850, 400)
167.             .ReadOnly = True
168.             .BorderStyle = BorderStyle.None
169.             .Font = New Font(.Font.FontFamily, .Font.Size + 3, .Font.Style)
170.
171.             .Text = "Place masses and select an option:
172. Calculate Potentials: View the equipotential lines about the masses,
173. Show Force Lines: Display gravitational field lines about the masses,
174. Add Test mass: Place a test mass and view how it is affected by the field."
174.         End With
175.         Home.Controls.Add(Instructions)
176.     End Sub
177. End Module

```

Navigation

Coupled with the layout creation module this enables the system to direct itself to where the user wishes. Using the address enumerable, it directs each button on the toolbar to the different sections in the program.

```
1. Module Navigation
2.     Dim Destination As Address
3.     Enum Address      'enumerable for the address to make addressing easier and t
   he program easier to use.
4.         Account
5.         Test
6.         Simulation
7.         Results
8.         Lesson1
9.         Lesson2
10.        Lesson3
11.     End Enum
12.     Public Sub ButtonDestination(sender As Object, e As EventArgs)
13.         Try
14.             Select Case sender.Text      'addresses each case to direct the system a
   fter it has been cleared
15.                 Case "Account"
16.                     MsgBox("Account")
17.                     Destination = Address.Account
18.                 Case "Tests"
19.                     MsgBox("Quiz")
20.                     Destination = Address.Test
21.                 Case "Simulation"
22.                     MsgBox("Simulation")
23.                     Destination = Address.Simulation
24.                 Case "Results"
25.                     Destination = Address.Results
26.             End Select
27.             ResetWindow()      'wipes the window clean
28.         Catch ex As Exception
29.             MsgBox("system error " & ex.Message)
30.         End Try
31.     End Sub
32.     Public Sub ListDestination(sender As Object, e As EventArgs)
33.         Select Case sender.Text
34.             Case "1. Work and Energy"
35.                 System.Diagnostics.Process.Start("https://www.khanacademy.org/scien
   ce/physics/work-And-energy/work-And-energy-tutorial/v/introduction-to-work-And-
   energy")
36.             Case "2. Gravitational Fields"
37.                 System.Diagnostics.Process.Start("https://www.youtube.com/watch?v=z
   dQ54siEfv")
38.             Case "3. Gravitational Potential"
39.                 System.Diagnostics.Process.Start("https://youtu.be/zdQ54siEfv?t=46
   6")
40.             Case "4. Escape velocity"
41.                 System.Diagnostics.Process.Start("https://youtu.be/zdQ54siEfv?t=14
   88")
42.             Case Else
43.                 MsgBox("do list stuff here")
44.                 System.Diagnostics.Process.Start("https://www.youtube.com/watch?v=z
   dQ54siEfv")
```

```

45.     End Select
46. End Sub
47. Public Sub ResetWindow()
48.     Home.Controls.Clear() 'removes all controls from the window
49.     Select Case Destination 'using the destination it creates the next relevant
page.
50.         Case 0
51.             'MsgBox("account")
52.             LayoutCreation.CreateAccountPage()
53.         Case 1
54.             'quiz
55.             LayoutCreation.CreateTestPage()
56.         Case 2
57.             'simulation
58.             LayoutCreation.CreateSimulationPage()
59.         Case 3
60.             'Results
61.             LayoutCreation.CreateResultsPage()
62.         Case 4
63.             'lesson1
64.         Case 5
65.             'lesson2
66.         Case 6
67.             'lesson3
68.     End Select
69. End Sub
70. End Module

```

Simulation

The core of the program where the visualisation of gravitational fields is processed and produced. With a large number of separate procedures and functions occurring it is by far the largest section of the program. Including multiple classes such as the environment, planet, testmass, fieldpoint and field properties; which demonstrate inheritance and composition. In combining these classes with multiple processes and visualising it a demonstration of the gravitational field about a system of masses is created.

```
1. Option Strict On
2.
3. Module Simulation
4.     '// potential wells option?
5.
6.     Private Canvas As New Environment()
7.     Private PaintPotentialshandle As New PaintEventHandler(AddressOf PaintPotential
8.         s)
9.     Private g As Graphics = Canvas.CreateGraphics
10.
11.     Private Controls(4) As Button
12.     Private Instructions As TextBox
13.     Private ControlsEnabled As Boolean = True
14.     Dim Addmass As Button
15.     Dim SliderValue As New Label
16.     Dim Slider As New TrackBar()
17.
18.     Private TimerEnabled As Boolean = False
19.     Private Timer As New Timer
20.     '//encountered an issue with the timer not incrementing by 0.05 and instead becom
21.     ing a recurring decimmlal .9999x
22.     Private time As Single = 0
23.     Private Timesteps As Integer = 0
24.     Private NextIndex As Integer = 0
25.     Private Const TimeStepsPerSecond As Single = 20
26.
27.     Private TestMassPlaceable As Boolean = True
28.
29.     Private Rnd As New Random
30.
31.     Private Numberofsatellites As Integer = 0
32.     Private satellites(Numberofsatellites - 1) As Planet
33.     Private TestMasses(199) As Testmass '//reasonably wont ever be more than 200 tes
34.     t masses
35.
36.     Private FieldPoints() As FieldPoint
37.     Private MaximumPotential As Single
38.
39.     Public Sub CreateCanvas()
40.         AddHandler Canvas.Paint, PaintPotentialshandle
41.         Home.Controls.Add(Canvas)
42.         Canvas.Location = New Point(5, 30)
43.
44.         Dim index As Integer = 0
45.         ReDim FieldPoints(Canvas.Width * Canvas.Height - 1) '//creates a field po
46.         int for each pixel in the canvas
47.         For x As Integer = 0 To Canvas.Width - 1
48.             For y As Integer = 0 To Canvas.Height - 1
49.                 FieldPoints(index) = New FieldPoint(New Point(x, y))
50.                 index += 1
51.             Next
52.         Next
```

```

48.     Next
49.
50.     LayoutCreation.Simulationcontrols(Controls, Addmass, SliderValue, Slider, I
nstructions) 'creates all the controls required for the simulation
51. End Sub
52. Public Sub Clear_click()
53.     Dim paintevent As New PaintEventArgs(g, New Rectangle(Canvas.Origin, Canvas
.Size))
54.
55.     ReDim satellites(0) 'clears the satellites list
56.     Canvas.ResetCanvas(Canvas, paintevent)
57.
58.     time = 0
59.     For Each testmass In TestMasses 'resets the testmass list
60.         testmass = Nothing
61.     Next
62.     MsgBox("done")
63. End Sub
64. Public Sub Potentials_click()
65.     AddHandler Canvas.Paint, PaintPotentialshandle
66.
67.     Dim index As Integer = 0
68.     For x As Integer = 0 To Canvas.Width - 1 'defines each fieldpoint in the
canvas and calculates the potential
69.         For y As Integer = 0 To Canvas.Height - 1
70.             FieldPoints(index) = New FieldPoint(New Point(x, y))
71.             FieldPoints(index).CalculatePotential(satellites, Canvas)
72.             index += 1
73.         Next
74.     Next
75.
76.     Canvas.Invalidate() 'causes the canvas to be repainted
77.     MsgBox("done")
78. End Sub
79. Public Sub ForceLines_click()
80.     Dim paintevent As New PaintEventArgs(g, New Rectangle(Canvas.Origin, Canvas
.Size))
81.     For Sat As Integer = 0 To satellites.Length - 1 'for each placed ma
ss in the system:
82.         If Not satellites(Sat) Is Nothing Then 'prevents break
s
83.             For index As Integer = 0 To satellites(Sat).Linecount - 1 'cy
cles through each of the starting points
84.                 Dim Start As FieldPoint = satellites(Sat).FieldStart(index)
85.                 MsgBox(Start.Location.X & ", " & Start.Location.Y)
86.                 Try
87.                     CreateForceLine(paintevent, satellites, Start)
88.                 Catch ex As Exception
89.                     MsgBox(ex.Message)
90.                 End Try
91.             Next
92.         End If
93.     Next
94.
95.     MsgBox("done")
96. End Sub
97. Public Sub CreateForceLine(ByVal paintevent As PaintEventArgs, ByVal planets()
As Planet, ByVal fp As FieldPoint)
98.     Dim BorderReached As Boolean = False
99.     BorderReached = fp.CalculateForce(planets, Canvas) 'works out the forc
e on each fieldpoint, returns a boolean stating whether the edge of the canvas has
been reached --> exit clause no. 1
100.     System.Threading.Thread.Sleep(0)
101.     If fp.FieldStrength > 2 Then 'if the strength is so small that it
points to the same point --> exit clause no.2
102.         Try

```

```

103.             paintevent.Graphics.DrawLine(Pens.DarkSlateGray, fp.Location
, fp.Vector.Location)
104.             Catch ex As Exception
105.                 MsgBox(ex.Message)
106.                 BorderReached = True 'if there is a failure then the syst
em aborts
107.             End Try
108.
109.             If BorderReached = False Then CreateForceLine(paintevent, planet
s, fp.Vector) 'recursively creates each field line
110.             End If
111.         End Sub
112.     Public Sub Slider_Change()
113.         SliderValue.Text = CStr(Slider.Value)
114.     End Sub
115.     Public Sub Addmass_Click()
116.         Dim AddMassHandler As New MouseEventHandler(AddressOf PlaceMass_mous
eClick)
117.         If ControlsEnabled = True Then
118.             Dim paintevent As New PaintEventArgs(g, New Rectangle(Canvas.Ori
gin, Canvas.Size))
119.             Canvas.ResetCanvas(Canvas, paintevent)
120.             For Each sat In satellites 'redraws each placed mass
121.                 If Not sat Is Nothing Then
122.                     paintevent.Graphics.FillEllipse(Brushes.Orange, sat.Loca
tion.X - 5 * sat.Mass / 1000000, sat.Location.Y - 5 * sat.Mass / 1000000, 10 * sat.
Mass / 1000000, 10 * sat.Mass / 1000000)
123.                 End If
124.             Next
125.             AddHandler Canvas.MouseClick, AddMassHandler 'adds the handle
r to the mouse
126.             Canvas.Cursor = Cursors.Cross 'in order to make it clear to th
e user
127.         Else
128.             RemoveHandler Canvas.MouseClick, AddMassHandler 'removes the mou
se handler
129.             Canvas.Cursor = Cursors.Default
130.         End If
131.
132.         ControlsEnabled = Not ControlsEnabled 'disables all controls other
than place mass for robustness
133.         For index = 1 To Controls.Length - 1
134.             Controls(index).Enabled = ControlsEnabled
135.         Next
136.     'Next
137.     End Sub
138.     Public Sub PlaceMass_mouseMove(ByVal sender As Object, ByVal mouse As Mo
useEventArgs)
139.         Dim paintevent As New PaintEventArgs(g, New Rectangle(Canvas.Origin,
Canvas.Size))
140.         paintevent.Graphics.FillRectangle(Brushes.Black, Canvas.Origin.X, Ca
nvas.Origin.Y, Canvas.Width, Canvas.Height)
141.
142.         For index = 0 To satellites.Length - 1
143.             paintevent.Graphics.FillEllipse(Brushes.Orange, satellites(index
).Location.X - (5 * satellites(index).Mass / 1000000), satellites(index).Location.Y
- (5 * satellites(index).Mass / 1000000), (10 * satellites(index).Mass / 1000000),
(10 * satellites(index).Mass / 1000000))
144.         Next
145.
146.         paintevent.Graphics.FillEllipse(Brushes.Orange, mouse.Location.X - 5
* Slider.Value, mouse.Location.Y - 5 * Slider.Value, 10 * Slider.Value, 10 * Slide
r.Value)
147.         'have it so a circle follows the mouse
148.     End Sub

```



```

149.         Public Sub PlaceMass_mouseClick(ByVal sender As Object, ByVal mouse As M
ouseEventArgs)
150.             'validating if the location has already been added to - required due
to the way vb calls this multiple times.
151.             Dim ValidPlace As Boolean = True
152.             If satellites.Length > 0 Then
153.                 Dim index As Integer = 0
154.                 Do
155.                     If Not satellites(index) Is Nothing Then
156.                         If mouse.Location = satellites(index).Location Then Vali
dPlace = False
157.                         End If
158.                         index += 1
159.                     Loop Until index > satellites.Length - 1 Or ValidPlace = False
160.                 End If
161.
162.                 If ValidPlace = True Then
163.                     Dim paintevent As New PaintEventArgs(g, New Rectangle(Canvas.Ori
gin, Canvas.Size))
164.                     ReDim Preserve satellites(satellites.Length)
165.                     Numberofsatellites += 1
166.                     If satellites(0) Is Nothing Then
167.                         satellites(0) = New Planet(Slider.Value, mouse.X, mouse.Y)
'creates a mass at the mouse's location
168.                     Else
169.                         satellites(satellites.Length - 1) = New Planet(Slider.Value,
mouse.X, mouse.Y)
170.                     End If
171.                     paintevent.Graphics.FillEllipse(Brushes.Orange, mouse.Location.X
- 5 * Slider.Value, mouse.Location.Y - 5 * Slider.Value, 10 * Slider.Value, 10 * S
lider.Value)
172.                 End If
173.             End Sub
174.         Public Sub TestMass_click()
175.             RemoveHandler Canvas.Paint, PaintPotentialshandle
176.             Dim TimerHandler As New EventHandler(AddressOf TimeChange)
177.             Dim PaintTestmassHandler As New PaintEventHandler(AddressOf PaintTes
tMasses)
178.             If TimerEnabled = False Then 'once clicked the first time it
adds all necessary handlers, then removes all of them the second time
179.                 Timer = New Timer
180.                 AddHandler Timer.Tick, TimerHandler
181.                 AddHandler Canvas.Paint, PaintTestmassHandler
182.                 Timer.Start()
183.                 'MsgBox("timer active")
184.             Else
185.                 RemoveHandler Timer.Tick, TimerHandler
186.                 RemoveHandler Canvas.Paint, PaintTestmassHandler
187.                 Timer.Stop()
188.             End If
189.             For index = 0 To Controls.Length - 1
190.                 If index <> 3 Then 'disables all controls except the testmass c
ontrol
191.                     Controls(index).Enabled = Not Controls(index).Enabled
192.                 End If
193.             Next
194.
195.             TimerEnabled = Not TimerEnabled 'enabled/disables the timer.
196.         End Sub
197.         Private Sub TimeChange(ByVal sender As Object, ByVal args As System.Even
tArgs) 'every time change
198.             Dim PlaceTestMassHandler As New MouseEventHandler(AddressOf PlaceTes
tMass)
199.             Dim paintevent As New PaintEventArgs(g, New Rectangle(Canvas.Origin,
Canvas.Size))
200.

```

```

201.         Try
202.             Timesteps += 1
203.             time = CSng(Timesteps / TimeStepsPerSecond)
204.             'time += dt
205.             'Controls(5).Text = CStr(time)
206.             If time Mod 0.5 = 0 Then
207.                 TestMassPlaceable = True
208.                 'only one mass placed if commented out
209.                 AddHandler Canvas.MouseMove, PlaceTestMassHandler
210.                 'enables a mass to be placed.
211.             End If
212.             Canvas.Invalidate(New Region(New RectangleF(Canvas.Origin, Canva
213. s.Size)))
214.         Catch ex As Exception
215.             Timer.Stop()
216.             MsgBox("Timing error: " & ex.Message)
217.         End Try
218.     End Sub
219.     Public Sub PlaceTestMass(ByVal sender As Object, ByVal mouse As MouseEve
220. ntArgs)
221.         Dim paintevent As New PaintEventArgs(g, New Rectangle(Canvas.Origin,
222. Canvas.Size))
223.         If TestMassPlaceable = True And NextIndex < 200 Then
224.             'paintevent.Graphics.FillEllipse(Brushes.Red, mouse.Location.X -
225. 2, mouse.Location.Y - 2, 4, 4)
226.             TestMasses(NextIndex) = New Testmass(mouse.Location)
227.             NextIndex += 1
228.             TestMassPlaceable = False
229.         End If
230.     End Sub
231.     Public Sub Measure_click()
232.         Dim Addmeasurehandler As New MouseEventHandler(AddressOf Measure_mou
233. remove)
234.         If ControlsEnabled Then 'enabled or disables the control
235.             AddHandler Canvas.MouseClick, Addmeasurehandler
236.         Else
237.             RemoveHandler Canvas.MouseClick, Addmeasurehandler
238.         End If
239.         ControlsEnabled = Not ControlsEnabled
240.         For index = 0 To Controls.Length - 1 'disables all other controls
241.             If index <> 4 Then Controls(index).Enabled = ControlsEnabled
242.         Next
243.         Addmass.Enabled = ControlsEnabled
244.     End Sub
245.     Private Sub Measure_mousemove(ByVal sender As Object, ByVal e As MouseEv
246. entArgs)
247.         Dim G As Single = 6.67408 * 10 ^ -11
248.         Dim testfieldpoint As New FieldPoint(e.Location)
249.         Dim testfieldproperty As New Fieldproperty()
250.         Dim potential, force As Single
251.         testfieldpoint.CalculatePotential(satellites, Canvas)
252.         potential = testfieldpoint.Potential
253.         testfieldpoint.CalculateForce(satellites, Canvas)
254.         force = testfieldpoint.FieldStrength
255.         potential *= G
256.         force *= G
257.         'testfieldpoint.Potential *= G
258.         MsgBox("Potential: " & potential & "
259. Strength: " & force & "N") 'wouldnt work with a textbox so had to use a mes
260. sage box
261.     End Sub
262.     Private Sub CompressTestmassList(ByVal Testmasses() As Testmass, ByRef R
263. emoveCount As Integer)

```

```

256.         Dim newnextindex As Integer = NextIndex
257.         If RemoveCount > 0 Then
258.             Dim ShiftTo As Integer = 0
259.             Dim ShiftFrom As Integer = 0
260.             While ShiftTo <= NextIndex - RemoveCount
261.                 While Testmasses(ShiftFrom) Is Nothing And ShiftFrom < Testma
asses.Length - 2
262.                     ShiftFrom += 1
263.                 End While
264.                 If ShiftTo <> ShiftFrom Then
265.                     Testmasses(ShiftTo) = Testmasses(ShiftFrom)
266.                     Testmasses(ShiftFrom) = Nothing
267.                     newnextindex = ShiftTo
268.                 End If
269.                 ShiftTo += 1
270.                 ShiftFrom += 1
271.                 If ShiftFrom > Testmasses.Length - 1 Then ShiftFrom = Testma
sses.Length - 1
272.             End While
273.             RemoveCount = 0
274.             NextIndex = newnextindex
275.         End If
276.     End Sub
277.     Private Sub PaintPotentials(ByVal sender As Object, ByVal paintargs As P
aintEventArgs)
278.         Try
279.             Canvas.PaintCanvas(satellites, Numberofsatellites, FieldPoints,
paintargs, Canvas)
280.         Catch ex As Exception
281.             MsgBox("petit erreur" & ex.Message)
282.         End Try
283.     End Sub
284.     Private Sub PaintTestMasses(ByVal sender As Object, ByVal paintargs As P
aintEventArgs)
285.         Dim removeCount As Integer = 0
286.
287.         Try
288.             If satellites.Length <> 0 Then 'draws each mass
289.                 For Each planet In satellites
290.                     If Not planet Is Nothing Then
291.                         Dim width As Integer = CInt(planet.Mass / 1000000 *
10)
292.                         paintargs.Graphics.FillEllipse(Brushes.Orange, CSng(
planet.Location.X - 0.5 * width), CSng(planet.Location.Y - 0.5 * width), CInt(width
), CInt(width))
293.                     End If
294.                 Next
295.
296.                 For index = 0 To TestMasses.Length - 1 'draws each testmass
297.                     If Not TestMasses(index) Is Nothing Then
298.                         TestMasses(index).UpdateLocation(satellites)
299.                         TestMasses(index).Lifespan -= 1
300.                         If TestMasses(index).Lifespan <= 0 Then
301.                             TestMasses(index) = Nothing 'need to th
en delete all testmasses that are 'nothing'
302.                             'paintargs.Graphics.FillEllipse(Brushes.Blue, te
stmass.Location.X - 3, testmass.Location.Y - 3, 6, 6)
303.                             removeCount += 1
304.                         Else
305.                             paintargs.Graphics.FillEllipse(Brushes.Red, Test
Masses(index).Location.X - 3, TestMasses(index).Location.Y - 3, 6, 6)
306.                         End If
307.                     End If
308.                 Next
309.             End If

```

```

310.
311.         'loop to clean the testmasses list
312.         CompressTestmassList(TestMasses, removeCount)
313.     Catch ex As Exception
314.         Timer.Stop()
315.         MsgBox("Ein Fault! " & ex.Message)
316.     End Try
317. End Sub
318. End Module
319.
320. Public Class Environment
321.     Inherits PictureBox
322.     Private _maximum As Point
323.     Private _origin As Point
324.     Private _maximumPotential As Single 'highest potential in the environmen
325.     Private _minimumPotential As Single 'lowest potential in the environment
326.
327.     Public Sub New()
328.         MyBase.Size = New Size(800, 500)
329.         MyBase.Location = New Point(5, 5)
330.         MyBase.BackColor = Color.Black
331.         _origin = New Point(0, 0)
332.         _maximum = MyBase.Location + MyBase.Size
333.     End Sub
334.
335.     Public Property Origin As Point
336.     Set(value As Point)
337.         _origin = value
338.     End Set
339.     Get
340.         Return _origin
341.     End Get
342. End Property
343.
344.     Public Property Maximum As Point
345.     Set(value As Point)
346.         _maximum = value
347.     End Set
348.     Get
349.         Return _maximum
350.     End Get
351. End Property
352.
353.     Public Property MaximumPotential As Single
354.     Set(value As Single)
355.         _maximumPotential = value
356.     End Set
357.     Get
358.         Return _maximumPotential
359.     End Get
360. End Property
361.
362.     Public Property MinimumPotential As Single
363.     Set(value As Single)
364.         _minimumPotential = value
365.     End Set
366.     Get
367.         Return _minimumPotential
368.     End Get
369. End Property
370.
371.     Public Sub ResetCanvas(ByRef canvas As Environment, ByVal paintargs As P
372.     aintEventArgs)
373.         paintargs.Graphics.FillRegion(Brushes.Black, New Region(New Rectangl
374.         e(canvas.Origin, canvas.Size)))
375.     End Sub

```

```

370.         Public Sub PaintCanvas(ByVal Satellites() As Object, ByVal NumberOfSatel
lites As Integer, ByVal FieldPoints() As FieldPoint, ByVal paintargs As PaintEventA
rgs, ByVal env As Environment)
371.             'coordinates from top left corner
372.             'paintargs.Graphics.DrawEllipse(Pens.Red, _origin.X, _origin.Y, 30,
30)
373.
374.             Dim CustomBrush As New SolidBrush(Color.White)
375.             Dim index As Integer = 0
376.             For Each Point In FieldPoints
377.                 If Not Point Is Nothing Then
378.                     Try
379.                         If Point.Potential <> 0 Then
380.                             CustomBrush.Color = (ConvertPotentialToColour(Point.
Potential, env.MaximumPotential, env.MinimumPotential))
381.                         Else
382.                             CustomBrush.Color = Color.Black
383.                         End If
384.                     Catch ex As Exception
385.                         MsgBox("brush error " & ex.Message)
386.                         CustomBrush.Color = Color.White
387.                     End Try
388.                     index += 1
389.                     paintargs.Graphics.FillEllipse(CustomBrush, Point.Location.X
, Point.Location.Y, 2, 2)
390.                 End If
391.             Next
392.         End Sub
393.         Public Function ConvertPotentialToColour(ByVal Potential As Single, ByVa
l Optional maxpotential As Single = 10000, Optional minPotential As Single = 0) As
Color
394.             '/converting to a rainbow of RGB/'
395.
396.             'firstly normalising the scalar value of potential in the range 0 -
> 1
397.             ' f = (potential - minPotential)/(Maxpotential-Minpotential)
398.
399.             'linearising the potential
400.             'Potential = Math.Log(Potential)
401.             'If minPotential <> 0 Then minPotential = Math.Log(minPotential)
402.             'maxpotential = Math.Log(maxpotential)
403.
404.             Dim f As Double
405.             Dim X As Integer
406.             Dim y As Integer
407.             Dim colour As Color
408.
409.             f = (Potential - minPotential) / (maxpotential - minPotential)
410.             If Potential <> Double.PositiveInfinity Then
411.                 If f < 0 Then f = 0
412.                 'now to split the scalar into 5 distinct colour groups
413.                 'f = (1 - Math.Log(f)) / 0.2 ' now f will be ranging from 0 to
5
414.                 f = (1 - f) / 0.2
415.                 X = CInt(Math.Floor(f))
416.                 y = CInt(Math.Floor(255 * (f - X)))
417.
418.                 'as x is will be either 0,1,2,3,4, or 5 this can be used for the
select statement
419.                 If y Mod 17 = 0 Then 'creates apparent equipotential lines as o
nly certain colours are selected.
420.                     Select Case X
421.                         Case 0
422.                             'keep red at maximum, blue at minimum and vary green
423.                             colour = Color.FromArgb(255, y, 0)

```

```

424.         Case 1
425.             'max green, min blue, vary red
426.             colour = Color.FromArgb(255 - y, 255, 0)
427.         Case 2
428.             'min red, max green, vary blue
429.             colour = Color.FromArgb(0, 255, y)
430.         Case 3
431.             ' min red, max blue, vary green
432.             colour = Color.FromArgb(0, 255 - y, 255)
433.         Case 4
434.             'min green, max blue, vary red
435.             colour = Color.FromArgb(y, 0, 255)
436.         Case 5
437.             colour = Color.FromArgb(255, 0, 255)
438.         Case Else
439.             colour = Color.White
440.         End Select
441.     Else
442.         colour = Color.Black
443.     End If
444.     Dim col As Integer = colour.ToArgb
445.     Return colour
446. End If
447. Return Color.Black
448. End Function
449.
450. End Class
451.
452. Public Class Planet
453.     Protected _mass As Single
454.     Protected _location As Point
455.     Private _linecount As Integer
456.     Private _Fieldstartingpoints() As FieldPoint
457.     'Private _Lines() as point
458.     Sub New(ByVal Optional mass As Single = 1, ByVal Optional xPoint As Integer = 250, ByVal Optional yPoint As Integer = 250)
459.         _mass = 1000000 * mass
460.         _location = New Point(xPoint, yPoint)
461.         _linecount = Cint(mass) * 8
462.         FieldStartingPoints()
463.     End Sub
464.     Public Property Mass As Single
465.         Set(value As Single)
466.             _mass = value
467.         End Set
468.         Get
469.             Return _mass
470.         End Get
471.     End Property
472.     Public Property Location As Point
473.         Set(value As Point)
474.             _location = value
475.         End Set
476.         Get
477.             Return _location
478.         End Get
479.     End Property
480.     Public Property Linecount As Integer
481.         Set(value As Integer)
482.             _linecount = value
483.         End Set
484.         Get
485.             Return _linecount
486.         End Get
487.     End Property
488.     Public Function FieldStart(ByVal index As Integer) As FieldPoint

```

```

489.         If _linecount > 0 Then Return _Fieldstartingpoints(index)
490.         Return Nothing
491.     End Function
492.     Public Sub FieldStartingPoints() 'creates a circle of equidistant poi
nts from the centre of the mass
493.         ReDim _Fieldstartingpoints(_linecount - 1)
494.         Dim x, y As Integer
495.         Dim dTheta, r As Double
496.         r = 5 * CInt(_mass / 1000000)
497.         dTheta = 2 * Math.PI / (_linecount)
498.         'If r * CInt(Math.Cos(2 * dTheta)) <= 1 Or r * CInt(Math.Cos(((Math.
PI / 2 - dTheta)))) <= 1 Then
499.
500.         For index = 0 To _linecount - 1
501.             x = CInt(r * Math.Cos(index * dTheta))
502.             y = CInt(r * Math.Sin(index * dTheta))
503.             _Fieldstartingpoints(index) = New FieldPoint(New Point(_location
.X + x, _location.Y - y)) 'y location increases as it goes down the screen
504.             'MsgBox(_Fieldstartingpoints(index).Location.X & ", " & _Fieldst
artingpoints(index).Location.Y)
505.         Next
506.     End Sub
507.     Public Function CalculatePotential(ByVal point As Point) As Single
508.         Dim radius As Single
509.         radius = CSng(Math.Sqrt((point.X - _location.X) ^ 2 + (point.Y - _lo
cation.Y) ^ 2))
510.         Return (_mass / radius)
511.     End Function
512.     Public Sub CalculateForce(ByVal location As Point, ByRef Field As Fieldp
roperty)
513.         Dim radius As Single
514.         Dim minradius As Integer = 20
515.         Dim ScalingFactor As Single = 0.005
516.
517.         radius = CSng(Math.Sqrt((location.X - _location.X) ^ 2 + (location.Y
- _location.Y) ^ 2))
518.         Field.Strength = _mass / 50000
519.         If radius < minradius Then radius = 20 'min radius required to ensu
re the size of the force around the smallest masses does not appear greater due to
the x^2 nature of the
520.         Field.Strength = CSng(_mass / radius ^ 2) * ScalingFactor
521.         Field.Direction = CalculateDirection(location, Field)
522.     End Sub
523.     Public Function CalculateDirection(ByVal location As Point, ByVal field
As Fieldproperty) As Single
524.         Dim Direction, angle As Double
525.         Dim Dy, Dx As Integer
526.         Direction = 0
527.         'direction is the angle of the vector from the fieldpoint to mass
528.
529.         Dy = _location.Y - location.Y 'vertical change top of screen t
o bottom
530.         Dx = location.X - _location.X 'horizontal change left to right
531.
532.         If Dy = 0 And Dx = 0 Then Return 9999 'case where the mass is in t
he planet
533.
534.         angle = Math.Atan(Math.Abs(Dy / Dx))
535.
536.         If Dy >= 0 Then
537.             If Dx >= 0 Then 'top right and right |_
538.                 Direction = angle
539.             Else
540.                 Direction = Math.PI - angle 'top left
541.             End If

```

```

542.         ElseIf Dy < 0 Then
543.             If Dx >= 0 Then                                     'bottom right
544.                 Direction = 0 - angle
545.             Else                                             'bottom left
546.                 Direction = Math.PI + angle
547.             End If
548.         End If
549.         Return CSng(Direction Mod 2 * Math.PI)
550.     End Function
551. End Class
552.
553. Public Class Testmass
554.     Inherits Planet
555.     Private _TMlocation As Point
556.     Private _Vx, _Vy, _Ax, _Ay As Single
557.     Private _field As Fieldproperty
558.     Private _lifespan As Integer
559.
560.     Sub New(ByVal location As Point)
561.         _mass = 1
562.         _TMlocation = location
563.         _field = New Fieldproperty
564.         _Vx = 0
565.         _Vy = 0
566.         _Ax = 0
567.         _Ay = 0
568.         _lifespan = 100
569.     End Sub
570.     Public Overloads Property Location As Point
571.         Set(value As Point)
572.             _TMlocation = value
573.         End Set
574.         Get
575.             Return _TMlocation
576.         End Get
577.     End Property
578.     Public Property Lifespan As Integer
579.         Set(value As Integer)
580.             _lifespan = value
581.         End Set
582.         Get
583.             Return _lifespan
584.         End Get
585.     End Property
586.     Public Sub UpdateAcceleration(ByVal planets() As Planet)
587.         Dim OldStrength, OldDirection As Single
588.         OldStrength = 0
589.         OldDirection = 9999
590.         For Each planet In planets
591.             If Not planet Is Nothing Then
592.                 planet.CalculateForce(_TMlocation, _field)
593.                 If _field.Strength > planet.Mass / 1000000 * 10 Then _lifesp
an = 0           'roughly within the draw circle of the planet
594.                 _field.Direction += CSng(Math.PI)           'correction as for fie
ldlines it is shifted 180 degrees needs to be shifted back.
595.                 _field.SumofGravitationalForce(_field, OldStrength, OldDirec
tion)
596.                 OldStrength = _field.Strength
597.                 OldDirection = _field.Direction
598.             Else
599.                 OldStrength = OldStrength
600.             End If
601.         Next
602.         _field.Strength *= CSng(0.05)           'scaling factor
603.         'force on the test mass is stored in the _field variable
604.         'f = ma so a = f/m = f/1 = f

```



```

605.         If _field.Direction <> 9999 Then
606.             _Ax = CSng(_field.Strength * Math.Cos(_field.Direction))
607.             _Ay = -CSng(_field.Strength * Math.Sin(_field.Direction))
608.             'Controls(0).Text = CStr(_Ax)           'make controls private again
609.             'Controls(1).Text = CStr(_Ay)
610.         Else
611.             _lifespan = 0
612.             'MsgBox("NaN")
613.         End If
614.     End Sub
615.     Public Sub UpdateVelocity(ByVal planets() As Planet)
616.         UpdateAcceleration(planets)
617.         If _lifespan <> 0 Then
618.             _Vx += _Ax
619.             _Vy += _Ay
620.         End If
621.     End Sub
622.     Public Sub UpdateLocation(ByVal planets() As Planet, Optional ByVal dt As
        Single = 1)
623.         UpdateVelocity(planets)
624.         If _lifespan <> 0 Then
625.             _TMlocation.X += CInt(_Vx * dt * 10)
626.             _TMlocation.Y += CInt(_Vy * dt * 10)
627.
628.             For Each planet In planets
629.                 If Not planet Is Nothing Then TooCloseToPlanet(planet) 'delete clause for the testmass, it has collided with the mass
630.             Next
631.         End If
632.     End Sub
633.     Public Sub TooCloseToPlanet(ByVal planet As Planet)
634.         If Math.Sqrt((planet.Location.X - _TMlocation.X) ^ 2 + (planet.Location.Y - _TMlocation.Y) ^ 2) < planet.Mass / 1000000 Then _lifespan = 0
635.     End Sub
636. End Class
637.
638. Public Class FieldPoint
639.     'Inherits Planet
640.     Private _location As Point
641.     Private _potential As Single
642.     Private _field As Fieldproperty
643.
644.     Public Sub New(location As Point, Optional potential As Single = 0, Optional fieldstrength As Single = 0)
645.         _location = location
646.         _field = New Fieldproperty(fieldstrength)
647.         _potential = potential
648.     End Sub
649.     Public Property Location As Point
650.         Set(value As Point)
651.             _location = value
652.         End Set
653.         Get
654.             Return _location
655.         End Get
656.     End Property
657.     Public Property Potential As Single
658.         Set(value As Single)
659.             _potential = value
660.         End Set
661.         Get
662.             Return _potential
663.         End Get
664.     End Property
665.     Public Property FieldStrength As Single

```

```

666.         Set(value As Single)
667.             _field.Strength = value
668.         End Set
669.         Get
670.             Return _field.Strength
671.         End Get
672.     End Property
673.     Public Property Vector As FieldPoint
674.         Set(value As FieldPoint)
675.             _field.Vector = value
676.         End Set
677.         Get
678.             Return _field.Vector
679.         End Get
680.     End Property
681.     Public Property Field As Fieldproperty
682.         Set(Value As Fieldproperty)
683.             _field = Value
684.         End Set
685.         Get
686.             Return _field
687.         End Get
688.     End Property
689.
690.     Public Sub CalculatePotential(planets() As Planet, env As Environment)
691.         For Each planet In planets
692.             If Not planet Is Nothing Then
693.                 _potential += planet.CalculatePotential(_location)
694.             Else
695.                 _potential += 0
696.             End If
697.         Next
698.
699.
700.
701.         If _potential < 1 Then _potential = 1 'adds limits so that the col
our scaling works with the potentials function.
702.         If _potential > 100000 Then _potential = 100000
703.
704.         If _potential > env.MaximumPotential And _potential <> Double.Positi
veInfinity Then
705.             env.MaximumPotential = _potential 'stores the maximum potentia
l found in the system.
706.         End If
707.         If _potential < env.MinimumPotential Then
708.             env.MinimumPotential = _potential 'stores the minimum potentia
l in the system.
709.         End If
710.         MsgBox(_potential)
711.     End Sub
712.     Public Function CalculateForce(ByVal planets() As Planet, ByVal canvas A
s Environment) As Boolean
713.         Dim CurrentDirection, CurrentMagnitude As Single
714.         For Each planet In planets
715.             If Not planet Is Nothing Then
716.                 CurrentDirection = _field.Direction 'saves old values of mag
nitude and direction so the vector sum can be calculated
717.                 CurrentMagnitude = _field.Strength 'first time through the
magnitude of the force will be 0 so there will be no force
718.
719.                 planet.CalculateForce(_location, _field) 'calculates the
attraction from the next planet
720.                 'only need to sum two forces at a time
721.                 If CurrentDirection <> 9999 Then
722.                     _field.SumofGravitationalForce(_field, CurrentMagnitude,
CurrentDirection) 'takes field byref so updated values stored in _field

```

```

723.             End If
724.
725.             Else
726.                 _field.Strength += 0
727.             End If
728.         Next
729.         Return _field.CalculateVector(_location, canvas)
730.     End Function
731. End Class
732.
733. Public Class Fieldproperty
734.     Private _FieldStrength As Single
735.     Private _FieldDirection As Single
736.     Private _FieldVector As FieldPoint 'instead of a traditional vector each
point on the field will point to another point, creating a list that will be the p
ath of the line
737.     'the line should be created from the centre of each plant
738.     Public Sub New(ByVal Optional strength As Single = 0, ByVal Optional dir
ection As Single = 9999) '9999 is a null value of direction
739.         _FieldStrength = strength
740.         _FieldDirection = direction
741.         _FieldVector = Nothing
742.     End Sub
743.
744.     Public Property Strength As Single
745.         Set(value As Single)
746.             _FieldStrength = value
747.         End Set
748.         Get
749.             Return _FieldStrength
750.         End Get
751.     End Property
752.     Public Property Direction As Single
753.         Set(Value As Single)
754.             _FieldDirection = Value
755.         End Set
756.         Get
757.             Return _FieldDirection
758.         End Get
759.     End Property
760.     Public Property Vector As FieldPoint
761.         Set(Value As FieldPoint)
762.             _FieldVector = Value
763.         End Set
764.         Get
765.             Return _FieldVector
766.         End Get
767.     End Property
768.
769.     Public Sub SumofGravitationalForce(ByRef field As Fieldproperty, ByVal 0
ldMagnitude As Single, ByVal OldDirection As Single)
770.         Dim x1, y1, x2, y2, xR, yR, Direction, angle As Double
771.
772.         x1 = (field.Strength * Math.Cos(field.Direction)) 'new direction of
the field to be added
773.         y1 = (field.Strength * Math.Sin(field.Direction))
774.
775.         x2 = (OldMagnitude * Math.Cos(OldDirection)) 'old direction of the f
ield to be added
776.         y2 = (OldMagnitude * Math.Sin(OldDirection))
777.
778.         xR = x1 + x2 'sum of components/resultant force
779.         yR = y1 + y2
780.         angle = Math.Atan(Math.Abs(yR / xR))
781.
782.         If yR >= 0 Then

```

```

783.             If xR >= 0 Then           'top right and right |_
784.                 Direction = angle
785.             Else
786.                 Direction = Math.PI - angle   'top left
787.             End If
788.             ElseIf yR < 0 Then
789.                 If xR >= 0 Then           'bottom right
790.                     Direction = 0 - angle
791.                 Else
792.                     Direction = Math.PI + angle   'bottom left
793.                 End If
794.             End If
795.
796.
797.             field.Strength = CSng(Math.Sqrt(xR ^ 2 + yR ^ 2))
798.             field.Direction = CSng(Direction Mod 2 * Math.PI)
799.
800.         End Sub
801.         Public Function CalculateVector(ByVal currentlocation As Point, ByVal ca
nvas As Environment) As Boolean
802.             Dim dest As Point
803.             dest = New Point(currentlocation.X + CInt(_FieldStrength * Math.Cos(
_FieldDirection)), currentlocation.Y - CInt(_FieldStrength * Math.Sin(_FieldDirecti
on))) 'y location increases as it goes down the screen
804.             _FieldVector = New FieldPoint(dest)
805.             If dest.X > canvas.Maximum.X Or dest.X < canvas.Origin.X Or dest.Y <
canvas.Origin.Y Or dest.Y > canvas.Maximum.Y Then 'end recursive loop
806.                 Return True 'returns all the way to borderreached in createforce
line subroutine
807.             End If
808.             Return False
809.         End Function
810.     End Class

```

Login System

A separate form that enables the user to log into their account or access the registry form. While not essential it be a separate form for ease of design and separation it has been done in this way.

In order to gather the combination of user details it must have strong links to the database. A dictionary has been selected to store the user details as it enabled a key value combination of each user code and password in the database to be created.

```

1. Public Class Login_System
2.     Dim PriveledgeLevel As UserLevel
3.     Dim StudentLoginDetails As Dictionary(Of String, String)
4.     Dim TeacherLoginDetails As Dictionary(Of String, String)
5.     Public Enum UserLevel
6.         Admin
7.         Teacher
8.         Student
9.     End Enum
10.    Public Sub Login_Load()
11.        PriveledgeLevel = UserLevel.Student
12.        StudentLoginDetails = New Dictionary(Of String, String)
13.        TeacherLoginDetails = New Dictionary(Of String, String)
14.        StudentLoginDetails = Database.GatherLoginDetails(StudentLoginDetails, "Stu
dents")
15.        TeacherLoginDetails = Database.GatherLoginDetails(TeacherLoginDetails, "Tea
chers")
16.    End Sub

```

```

17.     Private Sub Register_Click(sender As Object, e As EventArgs) Handles B_Register
18.         Register_System.PassOverLoginList(StudentLoginDetails, TeacherLoginDetails)
19.         Register_System.Show()
20.         Me.Close()
21.     End Sub
22.     Private Sub StudentsLogin_Click(sender As Object, e As EventArgs) Handles StudentsLogin.Click
23.         B_Login.BackColor = StudentsLogin.BackColor
24.         PriveledgeLevel = UserLevel.Student
25.     End Sub
26.     Private Sub TeachersLogin_Click(sender As Object, e As EventArgs) Handles TeachersLogin.Click
27.         B_Login.BackColor = TeachersLogin.BackColor
28.         PriveledgeLevel = UserLevel.Teacher
29.     End Sub
30.     Private Sub Login_Click(sender As Object, e As EventArgs) Handles B_Login.Click
31.         'check database for valid login
32.         Dim validLogin As Boolean = False
33.         Select Case PriveledgeLevel
34.             Case UserLevel.Teacher
35.                 validLogin = CheckLoginDetails(TeacherLoginDetails, Text_Username.Text, Text_Password.Text)
36.             Case Else
37.                 validLogin = CheckLoginDetails(StudentLoginDetails, Text_Username.Text, Text_Password.Text)
38.         End Select
39.         MsgBox(CStr(validLogin))
40.         If validLogin Then
41.             Home.LoggedIn = True
42.             Home.UserCode = Text_Username.Text
43.             Home.PriveledgeLevel = PriveledgeLevel
44.             Me.Close()
45.         End If
46.     End Sub
47.     Private Function CheckLoginDetails(ByVal LoginDetails As Dictionary(Of String, String), ByVal Username As String, ByVal Password As String) As Boolean
48.         Dim Keyvaluepair As New KeyValuePair(Of String, String)(Username, Password)
49.
50.         If LoginDetails.Contains(Keyvaluepair) Then Return True
51.         Return False
52.     End Function
53.     Private Sub ForgotPassword_LinkClicked(sender As Object, e As LinkLabelLinkClickedEventArgs)
54.
55.     End Sub
56. End Class

```

Register System

The register system works with the login system, allowing a new user to create an account by entering in their forename, surname, email and a password. This account must be added to the database and should allow a user with the correct login details into the account again.

```
1. Public Class Register_System
2.     Private StudentLogins As Dictionary(Of String, String)
3.     Private TeacherLogins As Dictionary(Of String, String)
4.     Private Usercode As String
5.     Private isTeacher As Boolean
6.     Public Sub PassOverLoginList(ByVal Students As Dictionary(Of String, String), B
yVal Teachers As Dictionary(Of String, String))
7.         StudentLogins = Students
8.         TeacherLogins = Teachers
9.     End Sub
10.    Private Sub Register_System_Load(sender As Object, e As EventArgs) Handles MyBase
.Load
11.
12.    End Sub
13.    Private Sub B_Register_Click(sender As Object, e As EventArgs) Handles B_Regist
er.Click
14.        Dim validdetails As Boolean = False
15.        'Check names
16.        If IsValidNameFormat(Text_Username.Text) And IsValidNameFormat(Text_Surname
.Text) And IsValidEmailFormat(Text_Email.Text) Then
17.            If Text_Password1.Text = Text_Password2.Text Then
18.                If TeacherBox.Checked Then
19.                    Home.PriveledgeLevel = Login_System.UserLevel.Teacher
20.                    Usercode = GenerateUserCode(False)
21.                    MsgBox("Your usercode is: " & Usercode & " make sure to remembe
r it.")
22.                    isTeacher = True
23.                    Database.AddTeacherData(Usercode, Text_Username.Text, Text_Surn
ame.Text, Text_Email.Text, Text_Password1.Text)
24.                    Dim Classcode As Integer = TeacherLogins.Count
25.                    MsgBox("Classcode: " & Classcode)
26.                    AddTeacherClass(Classcode, Usercode)
27.                Else
28.                    Home.PriveledgeLevel = Login_System.UserLevel.Student
29.                    Usercode = GenerateUserCode(True)
30.                    MsgBox("Your usercode is: " & Usercode & " make sure to remembe
r it.")
31.                    isTeacher = False
32.                    Database.AddStudentData(Usercode, Text_Username.Text, Text_Surn
ame.Text, Text_Email.Text, Text_Password1.Text)
33.                    'Database.AddClassCode(Usercode, "10010")
34.                End If
35.                Home.UserCode = Usercode
36.                Home.LoggedIn = True
37.                Me.Close()
38.            Else
39.                MsgBox("Passwords do not match, please retry")
40.            End If
41.        Else
42.            MsgBox("Invalid username and/or surname and/or email, please retry")
43.        End If
44.
45.        'check email duplicate?
46.        'check no duplicates -
cant be a usercode duplicate as not generated yet!
47.
```

```

48.         'ADD DETAILS TO THE DATABASE
49.
50.     End Sub
51.     Function IsValidNameFormat(ByVal s As String) As Boolean
52.         If Not System.Text.RegularExpressions.Regex.IsMatch(s, "[A-Za-
z]+$") Then MsgBox("Name must only contain letters")
53.         Return System.Text.RegularExpressions.Regex.IsMatch(s, "[A-Za-z]+$")
54.     End Function
55.     Function IsValidEmailFormat(ByVal s As String) As Boolean
56.         If Not System.Text.RegularExpressions.Regex.IsMatch(s, "^[0-9a-zA-Z]([-
.\w]*[0-9a-zA-Z])*@[0-9a-zA-Z]([-.\w]*[0-9a-zA-Z]\.)+[a-zA-
Z]{2,9})$" ) Then MsgBox("Email must contain an @ sign and a domain")
57.         Return System.Text.RegularExpressions.Regex.IsMatch(s, "^[0-9a-zA-Z]([-
.\w]*[0-9a-zA-Z])*@[0-9a-zA-Z]([-.\w]*[0-9a-zA-Z]\.)+[a-zA-Z]{2,9})$" )
58.     End Function
59.
60.     Function GenerateUserCode(ByVal Isstudent As Boolean) As String
61.         If Isstudent Then 'it is a student
62.             Return 10000 + StudentLogins.Count
63.         Else
64.             Return 1000 + TeacherLogins.Count
65.         End If
66.     End Function
67. End Class

```

Account

After a user has logged in they should be able to access the account form where they are able to alter their class code. Enabling a student to connect themselves to a teacher within the database.

```

1. Public Class Account
2.     Private UserCode, classcode As String
3.     Public Sub LoadStudentAccount()
4.         UserCode = Home.UserCode
5.         'get current ClassCode
6.         MsgBox("Usercode: " & UserCode)
7.         classcode = Database.GetClassCode(UserCode)
8.         ClassCodeTextBox.Text = classcode
9.     End Sub
10.
11.     Public Sub LoadTeacherAccount()
12.         ClassCodeButton.Visible = False
13.         ClassCodeTextBox.Visible = False
14.     End Sub
15.     Private Sub LogOut_click(sender As Object, e As EventArgs) Handles LogOutButton
.Click
16.         ClassCodeButton.Visible = True
17.         ClassCodeTextBox.Visible = True
18.         UserCode = ""
19.         Home.PriveledgeLevel = Nothing
20.         Home.LoggedIn = False
21.         Me.Close()
22.     End Sub
23.     Private Sub ClassCodeButton_Click(sender As Object, e As EventArgs) Handles Cla
ssCodeButton.Click
24.         Database.AddClassCode(UserCode, ClassCodeTextBox.Text)
25.     End Sub
26. End Class

```

Database

The data back end of the system where all the login details of both students and teachers, and the test results of each student. It needs to be robust and capable of handling large amounts of data.

```
3.  'added reference Microsoft Ado Ext 2.8 for DDL and Security
4.  Option Strict On
5.  Imports ADOX
6.  Imports System.Data.OleDb
7.  Module Database
8.      Public DatabaseName As String = "TestDB"
9.      Private MyConnection As New OleDbConnection
10.     Private MyCommand As New OleDbCommand
11.     Private DDLstr As String
12.     Public Sub DatabaseMasterCreator(name As String)
13.         CreateDatabase(DatabaseName)
14.         OpenDatabase(name)
15.         CreateStudentTable() 'creates each table independently
16.         CreateTeacherTable()
17.         CreateClassesTable()
18.         CreateResultsTable()
19.         CreateTestsTable()
20.         MyConnection.Close()
21.         'creating all the tables
22.     End Sub
23.     Public Sub CreateDatabase(name As String)
24.         Dim cat As Catalog = New Catalog()
25.         Try
26.             cat.Create("Provider=Microsoft.Jet.OLEDB.4.0;" &
27.                 "Data Source=" & name & ".mdb;" &
28.                 "Jet OLEDB:Engine Type=5")
29.             MyConnection = New OleDbConnection
30.             MyConnection.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" & name & ".mdb"
31.             MyConnection.Open()
32.             Catch ex As Exception
33.                 MsgBox("Database Creation failed " & ex.Message)
34.             Finally
35.                 MyConnection.Close()
36.             End Try
37.             cat = Nothing
38.         End Sub
39.         Public Sub OpenDatabase(name As String)
40.             Try
41.                 MyConnection.ConnectionString = ("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" & name & ".mdb")
42.                 MyConnection.Open()
43.                 Catch ex As Exception
44.                     MsgBox("Database not opened " & ex.Message)
45.                 End Try
46.             End Sub
47.             Public Sub CreateTable(tablename As String)
48.                 Try
49.                     MyCommand = New OleDb.OleDbCommand("CREATE TABLE [" & tablename & "]",
MyConnection)
50.                     MyCommand.ExecuteNonQuery()
51.                     Catch ex As Exception
52.                         MsgBox(ex.Message)
53.                     End Try
54.             End Sub
```



```

55.     Private Sub CreateStudentTable()
56.         DDLstr = "CREATE TABLE [Students] ([StudentCode] INT CONSTRAINT PrimaryKey
PRIMARY KEY, [StudentClass] INT, [Forename] Varchar(25), [Surname] Varchar(25), [Em
ail] Varchar(25), [Password] Varchar(25), [DateJoined] DATE)"
57.         Try
58.             MyCommand = New OleDb.OleDbCommand(DDLstr, MyConnection)
59.             MyCommand.ExecuteNonQuery()
60.             'msgbox("Students table created")
61.         Catch ex As Exception
62.             MsgBox("Students table not created: " & ex.Message)
63.         End Try
64.     End Sub
65.     Private Sub CreateTeacherTable()
66.         DDLstr = "CREATE TABLE [Teachers] ([TeacherCode] INT CONSTRAINT PrimaryKey
PRIMARY KEY, [Forename] Varchar(25), [Surname] Varchar(25), [Email] Varchar(25), [P
assword] Varchar(25))"
67.         Try
68.             MyCommand = New OleDb.OleDbCommand(DDLstr, MyConnection)
69.             MyCommand.ExecuteNonQuery()
70.         Catch ex As Exception
71.             MsgBox("Teacher table not created: " & ex.Message)
72.         End Try
73.     End Sub
74.     Private Sub CreateClassesTable()
75.         DDLstr = "CREATE TABLE [Classes] ([Classcode] INT CONSTRAINT PrimaryKey PRI
MARY KEY, [TeacherCode] INT)"
76.         Try
77.             MyCommand = New OleDb.OleDbCommand(DDLstr, MyConnection)
78.             MyCommand.ExecuteNonQuery()
79.         Catch ex As Exception
80.             MsgBox("Teacher table not created: " & ex.Message)
81.         End Try
82.     End Sub
83.     Private Sub CreateResultsTable()
84.         DDLstr = "CREATE TABLE [Results] ([StudentCode] INT, [HomeworkCode] INT, [P
ercentage] INT, [Feedback] Varchar(50))"
85.         Dim SQLstr As String = "ALTER TABLE Results ADD CONSTRAINT PrimaryKey PRIMA
RY KEY ([StudentCode], [HomeworkCode])"
86.         Try
87.             MyCommand = New OleDb.OleDbCommand(DDLstr, MyConnection)
88.             MyCommand.ExecuteNonQuery()
89.             MyCommand = New OleDb.OleDbCommand(SQLstr, MyConnection)
90.             MyCommand.ExecuteNonQuery()
91.         Catch ex As Exception
92.             MsgBox("Results table not created: " & ex.Message)
93.         End Try
94.     End Sub
95.     Private Sub CreateTestsTable()
96.         DDLstr = "CREATE TABLE [Tests] ([TestCode] INT CONSTRAINT PrimaryKey PRIMAR
Y KEY, [Questions] INT)"
97.         Try
98.             MyCommand = New OleDb.OleDbCommand(DDLstr, MyConnection)
99.             MyCommand.ExecuteNonQuery()
100.        Catch ex As Exception
101.            MsgBox("Tests table not created: " & ex.Message)
102.        End Try
103.    End Sub
104.    Public Sub AddField(tablename As String, field As String, type As String
)
105.        Try
106.            MyCommand = New OleDb.OleDbCommand("ALTER TABLE [" & tablename &
"] ADD COLUMN [" & field & "]" & type, MyConnection)
107.            MyCommand.ExecuteNonQuery()
108.        Catch ex As Exception
109.            MsgBox(ex.Message)
110.        End Try

```

```

111.         End Sub
112.     Public Sub AddClassCode(ByVal ID As String, ByVal data As String)
113.         Dim SQLstr As String
114.         Dim RowsUpdated As Integer = 0
115.         OpenDatabase(DatabaseName)
116.         Try
117.             SQLstr = "UPDATE Students SET [StudentClass] = @data WHERE [Stud
entCode] = @condition"
118.             MyCommand = New OleDbCommand(SQLstr, MyConnection)
119.             MyCommand.Parameters.AddWithValue("@data", data)
120.             MyCommand.Parameters.AddWithValue("@condition", ID)
121.             RowsUpdated = MyCommand.ExecuteNonQuery()
122.             If RowsUpdated > 0 Then MsgBox("Successfully added")
123.         Catch ex As Exception
124.             MsgBox("error inserting the students class: " & ex.Message)
125.         Finally
126.             MyConnection.Close()
127.         End Try
128.     End Sub
129.     Public Sub AddTeacherClass(ByVal Classcode As Integer, ByVal teachercode
As String)
130.         Dim SQLstr As String
131.         OpenDatabase(DatabaseName)
132.         Try
133.             SQLstr = "INSERT INTO Classes ([ClassCode], [TeacherCode]) VALUE
S(@classcode, @teachercode)"
134.             MyCommand = New OleDbCommand(SQLstr, MyConnection)
135.             With MyCommand.Parameters
136.                 .AddWithValue("@classcode", Classcode)
137.                 .AddWithValue("@teachercode", teachercode)
138.             End With
139.             MyCommand.ExecuteNonQuery()
140.         Catch ex As Exception
141.             MsgBox(ex.Message)
142.         Finally
143.             MyConnection.Close()
144.         End Try
145.     End Sub
146.     Public Sub AddStudentData(ByVal usercode As String, ByVal forename As St
ring, ByVal surname As String, ByVal email As String, ByVal password As String)
147.         Dim SQLstr As String
148.         OpenDatabase(DatabaseName)
149.         Try
150.             SQLstr = "INSERT INTO Students ([StudentCode], [Forename], [Surn
ame], [Email], [Password], [DateJoined]) VALUES (@StudentCode, @Forename, @Surname,
@email, @Password, @DateJoined)"
151.             MyCommand = New OleDbCommand(SQLstr, MyConnection)
152.             With MyCommand.Parameters
153.                 .AddWithValue("@StudentCode", usercode)
154.                 .AddWithValue("@Forename", forename)
155.                 .AddWithValue("@Surname", surname)
156.                 .AddWithValue("@Email", email)
157.                 .AddWithValue("@Password", password)
158.                 .AddWithValue("@DateJoined", System.DateTime.Today)
159.             End With
160.             MyCommand.ExecuteNonQuery()
161.         Catch ex As Exception
162.             MsgBox(ex.Message)
163.         Finally
164.             MyConnection.Close()
165.         End Try
166.     End Sub
167.     Public Sub AddTeacherData(ByVal usercode As String, ByVal forename As St
ring, ByVal surname As String, ByVal email As String, ByVal password As String)
168.         Dim SQLstr As String
169.         OpenDatabase(DatabaseName)

```

```

170.         Try
171.             SQLstr = "INSERT INTO Teachers ([TeacherCode], [Forename], [Surname], [Email], [Password]) VALUES (@TeacherCode, @Forename, @Surname, @Email, @Password)"
172.             MyCommand = New OleDbCommand(SQLstr, MyConnection)
173.             With MyCommand.Parameters
174.                 .AddWithValue("@TeacherCode", usercode)
175.                 .AddWithValue("@Forename", forename)
176.                 .AddWithValue("@Surname", surname)
177.                 .AddWithValue("@Email", email)
178.                 .AddWithValue("@Password", password)
179.             End With
180.             MyCommand.ExecuteNonQuery()
181.         Catch ex As Exception
182.             MsgBox(ex.Message)
183.         Finally
184.             MyConnection.Close()
185.         End Try
186.     End Sub
187.     Public Sub AddFeedback(ByVal testcode As Integer, ByVal forename As String, ByVal surname As String, ByVal feedback As String)
188.         Dim SQLstr As String
189.         Dim RowsUpdated As Integer = 0
190.         OpenDatabase(DatabaseName)
191.         Try
192.             SQLstr = "UPDATE ((Tests INNER JOIN Results On Tests.TestCode = Results.HomeworkCode) INNER JOIN Students On Results.StudentCode = Students.StudentCode) Set Results.Feedback = '" & feedback
193.             SQLstr += "' WHERE Tests.TestCode = @testcode And Students.Forename = @forename And Students.Surname = @surname"
194.             MyCommand = New OleDbCommand(SQLstr, MyConnection)
195.             MyCommand.Parameters.Add("@testcode", OleDbType.Integer).Value = testcode
196.             MyCommand.Parameters.AddWithValue("@forename", OleDbType.VarChar).Value = forename
197.             MyCommand.Parameters.AddWithValue("@surname", OleDbType.VarChar).Value = surname
198.             RowsUpdated = MyCommand.ExecuteNonQuery()
199.             If RowsUpdated > 0 Then MsgBox("Successfully added")
200.         Catch ex As Exception
201.             MsgBox("Error inserting the students feedback: " & ex.Message)
202.         Finally
203.             MyConnection.Close()
204.         End Try
205.     End Sub
206.     Public Sub AddResult(ByVal testcode As Integer, ByVal studentcode As Integer, ByVal percent As Integer, ByVal feedback As String)
207.         Dim SQLstr As String
208.         Dim RowsUpdated As Integer = 0
209.         OpenDatabase(DatabaseName)
210.         Try
211.             SQLstr = "INSERT INTO Results ([StudentCode], [HomeworkCode], [Percentage], [Feedback]) VALUES (@testcode, @studentcode, @percent, '" & feedback & "')"
212.             MyCommand = New OleDbCommand(SQLstr, MyConnection)
213.             With MyCommand.Parameters
214.                 .AddWithValue("@testcode", OleDbType.Integer).Value = testcode
215.                 .AddWithValue("@studentcode", OleDbType.Integer).Value = studentcode
216.                 .AddWithValue("@percent", OleDbType.Integer).Value = percent
217.             End With
218.             RowsUpdated = MyCommand.ExecuteNonQuery()
219.             If RowsUpdated > 0 Then MsgBox("Successfully added")
220.         Catch ex As Exception

```

```

221.             MsgBox("Could not add result: " & ex.Message)
222.         Finally
223.             MyConnection.Close()
224.         End Try
225.
226.     End Sub
227.     Public Function GetColumnNamesInTable(ByVal tableName As String) As Array
228.         Dim restrictions As String() = New String() {Nothing, Nothing, tableName, Nothing}
229.         Try
230.             OpenDatabase(DatabaseName)
231.             Dim dataTable As DataTable = MyConnection.GetSchema("Columns", restrictions)
232.             Dim Cols() As DataRow = dataTable.Select()
233.             Dim ArrayOfNames(Cols.Length - 1) As String
234.             For i As Integer = 0 To Cols.Length - 1
235.                 ArrayOfNames(i) = CStr(Cols(i)("column_name"))
236.             Next
237.             Return ArrayOfNames
238.         Catch ex As Exception
239.             MsgBox(ex.Message)
240.         Finally
241.             MyConnection.Close()
242.         End Try
243.         Return Nothing
244.     End Function
245.     Public Function GetStudentnameFromResultsTable(ByVal Teachercode As String) As Array
246.         Dim SQLstr, name As String
247.         Dim names As New List(Of String)
248.         SQLstr = "SELECT Students.Forename, Students.Surname FROM ((Students INNER JOIN Classes On Students.StudentClass = Classes.ClassCode) INNER JOIN Teachers On Classes.TeacherCode = Teachers.TeacherCode) WHERE Classes.TeacherCode = @condition"
249.         'SQLstr = "Select Forename, Surname FROM Students WHERE Classes.ClassCode = Students.StudentClass And Classes.TeacherCode = @condition"
250.
251.         Dim Datareader As OleDbDataReader
252.         Try
253.             OpenDatabase(DatabaseName)
254.             MyCommand = New OleDbCommand(SQLstr, MyConnection)
255.             MyCommand.Parameters.AddWithValue("@condition", Teachercode)
256.             Datareader = MyCommand.ExecuteReader() 'ExecuteReader()
257.             If Datareader.HasRows Then
258.                 While Datareader.Read()
259.                     If Not Datareader.IsDBNull(0) Then
260.                         name = Datareader.GetString(0) & " " & Datareader.GetString(1)
261.                         'MsgBox("name: " & name)
262.                         names.Add(name)
263.                     End If
264.                 End While
265.             End If
266.         Catch ex As Exception
267.             MsgBox("Student names could not be gathered " & ex.Message)
268.         Finally
269.             MyConnection.Close()
270.         End Try
271.         Dim formattednames(names.Count - 1) As String
272.         For index As Integer = 0 To names.Count - 1
273.             formattednames(index) = CStr(names(index))
274.         Next
275.         Return formattednames
276.     End Function
277.     Public Function GetTableNames() As Array

```

```

278.         Dim connection As New OleDb.OleDbConnection(MyConnection.ConnectionS
tring)
279.         Dim restrictions As String() = New String() {Nothing, Nothing, Nothi
ng, "Table"} 'catalog, owner, table name, table type
280.         Try
281.             connection.Open()
282.             Dim dataTable As DataTable = connection.GetSchema("Tables", rest
rictions)
283.             Dim Tables() As DataRow = dataTable.Select()
284.             Dim ArrayOfNames(Tables.Length) As String
285.             For i As Integer = 0 To Tables.Length - 1
286.                 ArrayOfNames(i) = CStr(Tables(i)("table_name"))
287.             Next
288.             Return ArrayOfNames
289.         Catch ex As Exception
290.             MsgBox(ex.Message)
291.         Finally
292.             connection.Close()
293.         End Try
294.         Return Nothing
295.     End Function
296.     Public Function GatherLoginDetails(ByVal LoginDetails As Dictionary(Of S
tring, String), ByVal Tablename As String) As Dictionary(Of String, String)
297.         Dim SQLstr, Usercode, Password As String
298.         Dim Datareader As OleDbDataReader
299.         Select Case Tablename
300.             Case "Teachers"
301.                 SQLstr = "SELECT TeacherCode, Password FROM " & Tablename
302.             Case "Students"
303.                 SQLstr = "SELECT StudentCode, Password FROM " & Tablename
304.             Case Else
305.                 Return LoginDetails
306.         End Select
307.         Try
308.             OpenDatabase(DatabaseName)
309.             Dim SqlCommand As New OleDbCommand(SQLstr, MyConnection)
310.             Datareader = SqlCommand.ExecuteReader
311.             While Datareader.Read()
312.                 Usercode = CStr(Datareader.GetInt32(0))
313.                 Password = Datareader.GetString(1)
314.                 LoginDetails.Add(Usercode, Password)
315.             End While
316.         Catch ex As Exception
317.             MsgBox("Data could not be gathered " & ex.Message)
318.         Finally
319.             MyConnection.Close()
320.         End Try
321.         Return LoginDetails
322.     End Function
323.     Public Function GetClassCode(ByVal usercode As String) As String
324.         Dim SQLstr, classcode As String
325.         classcode = "nothing"
326.         Dim Datareader As OleDbDataReader
327.         SQLstr = "SELECT StudentClass FROM Students WHERE Students.StudentCo
de = @condition"
328.         Try
329.             OpenDatabase(DatabaseName)
330.             MyCommand = New OleDbCommand(SQLstr, MyConnection)
331.             MyCommand.Parameters.AddWithValue("@condition", OleDbType.Intege
r).Value = CInt(usercode)
332.             Datareader = MyCommand.ExecuteReader() 'ExecuteReader()
333.             If Datareader.HasRows Then
334.                 While Datareader.Read()
335.                     If Not Datareader.IsDBNull(0) Then
336.                         classcode = CStr(Datareader.GetInt32(0))
337.                     End If

```

```

338.         End While
339.     End If
340.     MsgBox("Classcode: " & classcode)
341. Catch ex As Exception
342.     MsgBox("Class code could not be gathered " & ex.Message)
343. Finally
344.     MyConnection.Close()
345. End Try
346. Return classcode
347. End Function
348. Public Function GetTestCodes() As Array
349.     Dim SQLstr, test As String
350.     Dim Tests As New List(Of String)
351.     Dim Datareader As OleDbDataReader
352.     SQLstr = "SELECT TestCode FROM Tests"
353.     Try
354.         OpenDatabase(DatabaseName)
355.         MyCommand = New OleDbCommand(SQLstr, MyConnection)
356.         Datareader = MyCommand.ExecuteReader() 'ExecuteReader()
357.         If Datareader.HasRows Then
358.             While Datareader.Read()
359.                 If Not Datareader.IsDBNull(0) Then
360.                     test = CStr(Datareader.GetInt32(0))
361.                     Tests.Add(test)
362.                 End If
363.             End While
364.         End If
365.     Catch ex As Exception
366.         MsgBox("Tests codes could not be gathered " & ex.Message)
367.     Finally
368.         MyConnection.Close()
369.     End Try
370.     Dim formattedtests(Tests.Count - 1) As String
371.     For index As Integer = 0 To Tests.Count - 1
372.         formattedtests(index) = CStr(Tests(index))
373.     Next
374.     Return formattedtests
375. End Function
376. Public Function GetStudentResults(ByVal usercode As String) As DataTable

377.     Dim ResultsTable As New DataTable
378.     Dim SQLstr As String
379.     Dim DataAdapter As OleDbDataAdapter
380.     Try
381.         OpenDatabase(DatabaseName)
382.         SQLstr = "SELECT HomeworkCode, Percentage, Feedback FROM Results
WHERE Results.StudentCode = @condition" ' OrderBy HomeworkCode DESC"
383.         MyCommand = New OleDbCommand(SQLstr, MyConnection)
384.         MyCommand.Parameters.AddWithValue("@condition", OleDbType.Integer).Value = CInt(usercode)
385.         DataAdapter = New OleDbDataAdapter(MyCommand)
386.         DataAdapter.Fill(ResultsTable)
387.     Catch ex As Exception
388.         MsgBox("Student results could not be gathered: " & ex.Message)
389.         Return Nothing
390.     Finally
391.         MyConnection.Close()
392.     End Try
393.     Return ResultsTable
394. End Function
395. Public Function GetStudentCode(ByVal forename As String, ByVal surname As String) As Integer
396.     Dim SQLstr As String
397.     Dim studentcode As Integer
398.     studentcode = 0
399.     Dim Datareader As OleDbDataReader

```

```

400.         SQLstr = "SELECT StudentCode FROM Students WHERE Students.Forename =
         @forename AND Students.Surname = @surname"
401.         Try
402.             OpenDatabase(DatabaseName)
403.             MyCommand = New OleDbCommand(SQLstr, MyConnection)
404.             MyCommand.Parameters.AddWithValue("@forename", forename)
405.             MyCommand.Parameters.AddWithValue("@surname", surname)
406.             Datareader = MyCommand.ExecuteReader() 'ExecuteReader()
407.             If Datareader.HasRows Then
408.                 While Datareader.Read()
409.                     If Not Datareader.IsDBNull(0) Then
410.                         studentcode = (Datareader.GetInt32(0))
411.                     End If
412.                 End While
413.             End If
414.             MsgBox("StudentCode: " & studentcode)
415.         Catch ex As Exception
416.             MsgBox("Student code could not be gathered " & ex.Message)
417.         Finally
418.             MyConnection.Close()
419.         End Try
420.         Return studentcode
421.     End Function
422.     Public Function GetTestmax(ByVal testcode As Integer) As Integer
423.         Dim SQLstr As String
424.         Dim Testmax As Integer
425.         Testmax = 0
426.         Dim Datareader As OleDbDataReader
427.         SQLstr = "SELECT Questions FROM Tests WHERE Tests.TestCode = @condit
         ion"
428.         Try
429.             OpenDatabase(DatabaseName)
430.             MyCommand = New OleDbCommand(SQLstr, MyConnection)
431.             MyCommand.Parameters.AddWithValue("@condition", OleDbType.Intege
         r).Value = (testcode)
432.             Datareader = MyCommand.ExecuteReader() 'ExecuteReader()
433.             If Datareader.HasRows Then
434.                 While Datareader.Read()
435.                     If Not Datareader.IsDBNull(0) Then
436.                         Testmax = (Datareader.GetInt32(0))
437.                     End If
438.                 End While
439.             End If
440.             MsgBox("testmax: " & Testmax)
441.         Catch ex As Exception
442.             MsgBox("testmax could not be gathered " & ex.Message)
443.         Finally
444.             MyConnection.Close()
445.         End Try
446.         Return Testmax
447.     End Function
448.     Public Function StudentResultsByName(ByVal Studentname As String) As Dat
         aTable
449.         'split student name into first and second
450.         Dim name(1), SQLstr As String
451.         Dim ResultsTable As New DataTable
452.         Dim dataadapter As OleDbDataAdapter
453.         name = Studentname.Split
454.         SQLstr = "SELECT Results.HomeworkCode, Results.Percentage, Results.F
         eedback FROM Students INNER JOIN Results ON Students.StudentCode = Results.StudentC
         ode WHERE Students.Forename = @forename AND Students.Surname = @surname;"
455.         Try
456.             OpenDatabase(DatabaseName)
457.             MyCommand = New OleDbCommand(SQLstr, MyConnection)
458.             With MyCommand.Parameters
459.                 .AddWithValue("@forename", name(0))

```

```

460.         .AddWithValue("@surname", name(1))
461.         End With
462.         dataadapter = New OleDbDataAdapter(MyCommand)
463.         dataadapter.Fill(ResultsTable)
464.         Catch ex As Exception
465.             MsgBox("Student Data could not be gathered by name " & ex.Message)
466.         End With
467.         Return Nothing
468.     Finally
469.         MyConnection.Close()
470.     End Try
471.     Return ResultsTable
472. End Function
473. Public Function StudentResultsByTest(ByVal testcode As String, ByVal TeacherCode As String) As DataTable
474.     Dim SQLstr As String
475.     Dim ResultsTable As New DataTable
476.     Dim dataadapter As OleDbDataAdapter
477.     SQLstr = "SELECT Students.Forename, Students.Surname, Results.Percentage, Results.Feedback
478.     FROM ((Students INNER JOIN Results ON Students.StudentCode = Results.StudentCode) INNER JOIN Tests ON Results.HomeworkCode = Tests.TestCode) INNER JOIN (Classes INNER JOIN Teachers ON Classes.TeacherCode = Teachers.TeacherCode) ON Students.StudentClass = Classes.Classcode
479.     WHERE Teachers.TeacherCode = @teachercode AND Tests.TestCode = @testcode"
480.     Try
481.         OpenDatabase(DatabaseName)
482.         MyCommand = New OleDbCommand(SQLstr, MyConnection)
483.         With MyCommand.Parameters
484.             .AddWithValue("@teachercode", TeacherCode)
485.             .AddWithValue("@testcode", testcode)
486.         End With
487.         dataadapter = New OleDbDataAdapter(MyCommand)
488.         dataadapter.Fill(ResultsTable)
489.         Catch ex As Exception
490.             MsgBox("Student Data could not be gathered by test number " & ex.Message)
491.         Return Nothing
492.     Finally
493.         MyConnection.Close()
494.     End Try
495.     Return ResultsTable
496. End Function
497. End Module

```


Results

The results page allows users to view either their results or the results of students in their class, this should enable a teacher to make multiple queries and gather data about their students in multiple ways.

```
1. 'Option Strict On
2. Module Results
3.     Private UserCode As String
4.     Private PriveledgeLevel As Login_System.UserLevel
5.     Private ResultsGrid As New DataGrid
6.     Private ResultsView As New DataGridView
7.     Private Labels(1) As Label
8.     Private DropDownLists(1) As ComboBox
9.     Private Checkboxes(1) As CheckBox
10.    Private QueryButton As New Button
11.    Private ResultsHandler As New DataGridViewCellEventHandler(AddressOf CellChange
    d)
12.    Public Sub PassVariables(ByVal code As String, ByVal level As Login_System.User
    Level)
13.        UserCode = code
14.        PriveledgeLevel = level
15.    End Sub
16.    Public Sub FormatStudentPage()
17.        ResultsView.DataSource = Database.GetStudentResults(UserCode)
18.        ResultsView.Location = New Point(50, 100)
19.        ResultsView.Size = New Size(700, 400)
20.        ResultsView.AutoSizeColumns()
21.        Home.Controls.Add(ResultsView)
22.    End Sub
23.    Public Sub FormatTeacherPage()
24.        For index As Integer = 0 To 1
25.            Labels(index) = New Label
26.            With Labels(index)
27.                .Location = New Point(250 * index + 90, 85)
28.                If index = 0 Then .Text = "By Student Name:"
29.                If index = 1 Then .Text = "By Test:"
30.            End With
31.            DropDownLists(index) = New ComboBox
32.            With DropDownLists(index)
33.                .Location = New Point(250 * index + 100, 110)
34.
35.                If index = 0 Then
36.                    .Text = "Student"
37.                    .DataSource = (Database.GetStudentnameFromResultsTable(UserCode
    ))
38.                ElseIf index = 1 Then
39.                    .Text = "Homework"
40.                    .DataSource = (Database.GetTestCodes())
41.                End If
42.                '.Items = Nothing 'get students from teacher class
43.            End With
44.            Checkboxes(index) = New CheckBox
45.            With Checkboxes(index)
46.                .Location = New Point((250 * index) + 225, 110)
47.                .Size = New Size(15, .Height)
48.                .Text = ""
49.            End With
50.            If index = 0 Then AddHandler Checkboxes(index).CheckedChanged, AddressOf
    f Uncheck1
51.            If index = 1 Then AddHandler Checkboxes(index).CheckedChanged, AddressOf
    f Uncheck0
52.
```

```

53.         Home.Controls.Add(Labels(index))
54.         Home.Controls.Add(DropDownLists(index))
55.         Home.Controls.Add(Checkboxes(index))
56.     Next
57.
58.     QueryButton.Location = New Point(510, 108)
59.     QueryButton.Text = "Query"
60.     Home.Controls.Add(QueryButton)
61.     AddHandler QueryButton.Click, AddressOf QueryResults
62. End Sub
63. Private Sub Uncheck1(ByVal sender As Object, e As EventArgs) 'ensures only o
ne of the checkboxes can be selected
64.     If Checkboxes(0).Checked = True Then Checkboxes(1).Checked = False
65. End Sub
66. Private Sub Uncheck0(ByVal sender As Object, e As EventArgs)
67.     If Checkboxes(1).Checked = True Then Checkboxes(0).Checked = False
68. End Sub
69. Private Sub QueryResults(ByVal sender As Object, e As EventArgs) 'adds all t
he results to a results view object
70.     MsgBox("Querying")
71.     ResultsView.Location = New Point(50, 150)
72.     ResultsView.Size = New Size(700, 400)
73.     ResultsView.AutoSizeColumns()
74.     If Checkboxes(0).Checked Then 'query by student name
75.         ResultsView.DataSource = Database.StudentResultsByName(DropDownLists(0)
.Text)
76.     Else 'query by test no.
77.         ResultsView.DataSource = Database.StudentResultsByTest(DropDownLists(1)
.Text, UserCode)
78.     End If
79.     AddHandler ResultsView.CellEndEdit, ResultsHandler
80.     Home.Controls.Add(ResultsView)
81. End Sub
82. Private Sub CellChanged(ByVal sender As Object, e As DataGridViewCellEventArgs)
83.     Dim row, column As Integer
84.     If Checkboxes(1).Checked And e.ColumnIndex = 3 Or Checkboxes(0).Checked And
e.ColumnIndex = 2 Then
85.         row = e.RowIndex
86.         column = e.ColumnIndex
87.         'append feedback to the DB
88.         If Checkboxes(0).Checked Then 'by student name
89.             Dim names(1) As String
90.             names = DropDownLists(0).Text.Split
91.             Database.AddFeedback(CInt(ResultsView(0, row).Value), names(0), nam
es(1), ResultsView(2, row).Value)
92.             'Database.AddFeedback(ResultsView(0, row).Value, names(0), names(1)
, ResultsView(2, row).Value)
93.         Else 'by test number
94.             Database.AddFeedback(CInt(DropDownLists(1).Text), ResultsView(0, ro
w).Value, ResultsView(1, row).Value, ResultsView(3, row).Value)
95.         End If
96.     End If
97.     MsgBox("active")
98. End Sub
99. End Module

```

Tests

The tests page is only accessible to teachers, it enables them to add a students result to the database, where the student can access it and view the feedback left for them.

```
1. Public Class Tests
2.     Private Sub Tests_Load(sender As Object, e As EventArgs) Handles MyBase.Load
3.         With StudentsList
4.             .DataSource = (Database.GetStudentnameFromResultsTable(Home.UserCode))
5.         End With
6.         With TestsList
7.             .DataSource = (Database.GetTestCodes())
8.         End With
9.     End Sub
10.    Private Sub B_Addresult_Click(sender As Object, e As EventArgs) Handles B_Addre
    sult.Click
11.        Dim name(1) As String
12.        Dim percent, maxscore As Integer
13.        maxscore = 0
14.        name = StudentsList.Text.Split
15.        maxscore = GetTestmax(CInt(TestsList.Text))
16.        If maxscore <> 0 And IsValidTestScore(ScoreTextBox.Text) Then
17.            percent = CInt(ScoreTextBox.Text) / maxscore * 100
18.            If percent <= 100 And percent >= 0 Then
19.                Try
20.                    AddResult(TestsList.Text, GetStudentCode(name(0), name(1)), per
    cent, FeedbackText.Text)
21.                Catch
22.                    MsgBox("invalid entry combination, please check and try again")
23.                End Try
24.            Else
25.                MsgBox("Invalid score")
26.            End If
27.        End If
28.    End Sub
29.
30.    Function IsValidTestScore(ByVal s As String) As Boolean
31.        If Not System.Text.RegularExpressions.Regex.IsMatch(s, "[0-
    9]+$") Then MsgBox("Invalid Score")
32.        Return System.Text.RegularExpressions.Regex.IsMatch(s, "[0-9]+$")
33.    End Function
34. End Class
```

Alpha Testing

Input Testing

No.	Purpose	Description	Data Typical, Erroneous, Extreme	Expected outcome	Actual outcome	Evidence in appendix
1.1	Validate Password	A password should be accepted if it is any combination of characters longer than(or equal to) 6 characters. And the two entered passwords match.	Treg1 – “Abcd1234” Ereg1 – “Ab1234” Xreg1 – “Word”	Accept Accept Error	Accept Accept Error	I-1.1
1.2	Validate Email	Email should be accepted if it contains an @ sign and a domain	Treg1 – “LBJ@myemail.com” Ereg1 – “Heresmyemail@email.com” Xreg1 – “ Notanemail”	Accept Accept Error	Accept Accept Error	I-1.2
1.3	Validate Forename	Forename should be accepted if it only contains letters.	Treg1 – “James” Ereg1 – “James” Xreg1 – “James12”	Accept Accept Error	Accept Accept Error	I-1.3
1.4	Validate Surname	Surname should be accepted if it only contains letters	Treg1 – “Hancock” Ereg1 – “Hancock” Xreg1 – “Hancock12”	Accept Accept Error	Accept Accept Error	I-1.4
2.1	Validate Classcode	Classcode must be of the correct format i.e an integer	Treg2 – 1 Ereg2 – 1 Xreg2 – 1a	Accept Accept Error	Accept Accept Error	I-1.5

Flow of Control Testing

VARIABLE PASSING TEST FOR STUDENT USER

No.	Variable	Description	Expected value	Actual value	Evidence in appendix
1.0	LoggedIn	Boolean variable that states whether a user is currently logged in	True	True	S-2.1
1.1	Usercode	String that contains the users code, not dependent on whether or not the user is a student or teacher	10000	"10000"	S-2.1
1.2	PriveledgeLevel	Innumerable which defines what level of access the user has	Userlevel.Student	Student{2}	S-2.1

VARIABLE PASSING TEST FOR TEACHER USER

No.	Variable	Description	Expected value	Actual value	Evidence in appendix
1.0	LoggedIn	Boolean variable that states whether a user is currently logged in	True	True	S-2.2
1.1	Usercode	String that contains the users code, not dependent on whether or not the user is a student or teacher	1000	"1000"	S-2.2
1.2	PriveledgeLevel	Innumerable which defines what level of access the user has	Userlevel.Teacher	Teacher{1}	S-2.2

Process Testing

CALCULATION OF GRAVITATIONAL POTENTIAL

To be a suitable simulation the calculation for the value of gravitational potential should be within 1%, seeing as there should be no errors within the application except possible rounding errors.

Expected Value	Tolerance	Actual Value	Actual Difference	Pass/Fail	Evidence in appendix
$6.26 * 10^6$	±1%	$6.251 * 10^6$	0.144%	Pass	P-1.1

As the actual value is within tolerance of the expected value any discrepancy in the result can be ignored.

CALCULATION OF GRAVITATIONAL FORCE

Like the calculation of gravitational potential the calculation of the gravitational force must be as accurate as possible to have little to no ramification on the function of the rest of the program.

Expected Value	Tolerance	Actual Value	Actual Difference	Pass/Fail	Evidence in appendix
9.81 N	1%	9.8134	0.347%	Pass	P-1.2

As the actual value is within the tolerance region the calculation is proved to be correct.

SUMMATION OF FORCES

For the summation of forces to be accurate both the calculation of the gravitational force and the calculation of the horizontal and vertical components of the force, summing these individual components respectively and then recombining the force to a magnitude and direction.

To accomplish this test two masses can be placed of relative masses 1 and 2. To ensure the correct testing distance has been used the radius variable in the calculate force function can be fixed to 1000. This means that the distance can be specified and it is easier to calculate the actual value instead of having to work out the actual distance between each mass and the testing field point. Which due to the programming methods used is a complex and time consuming task.

Expected Value	Tolerance	Actual Value	Actual Difference	Pass/Fail	Evidence in appendix
$6.67 * 10^{-11}$ N	1%	$6.67 * 10^{-11}$ N	<1%	Pass	P-1.2

The value for this is clearly accurate although the actual difference may be greater due to automatic rounding in the programming environment.

OTHER ALGORITHMIC FUNCTIONS

The other notable algorithms, the recursive elements within the field line function and the compression of the test mass list, are more easily demonstrated during the [testing video](#), so as a result there is no evidence of them within this documentation.

VISUAL TESTING

The appropriateness of the simulations visual appearance is vital to its overall effectiveness as a product, so to show its suitability to the product the following evidence is provided, though it will also be available within the [testing video](#).

Function description	Evidence
Potentials Function	P-2.1
Generate Field Line function	P-2.2
Test Mass Function	P-2.3

Storage Testing

Evidence for the correct creation of the database as to the DDL strings contained within the design.

The results should show that the database tables have been correctly created according to the design; with each table made up of its respective fields that have the correct data type.

Table	Evidence in appendix
Students	S-1.1
Teachers	S-1.2
Results	S-1.3
Classes	S-1.4
Tests	S-1.5

LOCATION TESTING FOR STUDENT REGISTERING

Once a student has registered an account all the information they entered should be input into the database. It is important that all the information should go into the correct field within the database so that it can be queried successfully.

Field	Entered Value
Usercode	Generated by the system (10003)
Forename	Albert
Surname	Einstein
Email	AE@genius.net
Password	Relative
Date Joined	Generated by the system (10/03/2020)

Criteria	Expected Field	Expected Data type	Actual Field	Actual Data type	Evidence in appendix
Usercode	Students.StudentCode	Integer	StudentCode	Number	S-2.1
Forename	Students.forename	Varchar	forename	Short text	S-2.1
Surname	Students.Surname	Varchar	Surname	Short text	S-2.1
Email	Students.Email	Varchar	Email	Short text	S-2.1
Password	Students.password	Varchar	password	Short text	S-2.1
Date Joined	Students.DataJoined	Date	DataJoined	Date/Time	S-2.1

The evidence should show that each of the entered values has been correctly placed within the database.

LOCATION TESTING FOR TEACHER REGISTERING

Field	Entered Value
Usercode	Generated by the system (1002)
Forename	Phillip
Surname	Fender
Email	PF@godalming.ac.uk
Password	PhysicsChallenge
Classcode	Generated by the system (2)

Criteria	Expected Field	Expected Data type	Actual Field	Actual Data type	Evidence in appendix
Usercode	Teachers.TeacherCode	Integer	Teachers.TeacherCode	Number	S-2.2
Usercode	Classes.TeacherCode	Integer	Classes.TeacherCode	Number	S-2.3
Forename	Teachers.forename	Varchar	Teachers.forename	Short text	S-2.2
Surname	Teachers.Surname	Varchar	Teachers.Surname	Short text	S-2.2
Email	Teachers.Email	Varchar	Teachers.Email	Short text	S-2.2
Password	Teachers.password	Varchar	Teachers.password	Short text	S-2.2
Classcode	Classes.Classcode	Integer	Classes.Classcode	Number	S-2.3

Like before it is important that the data is correctly entered into the database so that it can be successfully queried at a later date. All the data should be normalised to fulfil the requirements.

Requirements Testing

SIMULATION/MODEL REQUIREMENTS

Below is a summary of the pre-defined system requirements for the simulation aspect of the product along with a statement saying if the requirement has been met.

Requirement	Summary	Requirement met? (Pass/Fail)
1.0	Is the simulation easily useable?	Pass
1.1	The interface should well designed and organised logically	Pass
1.2	The interface controls should be suitably named	Pass
1.3	There should be a brief descriptions of the controls function	Pass
1.4	Any inputs must be limited so that a system failure is avoided	Pass
2.0	The visualisation should be understandable	Pass
2.1	Model should be clear and its visualisation should be quickly understandable.	Pass
2.2	Gravitational field strength must be correctly calculated	Pass
2.3	Gravitational field strength must be correctly calculated	Pass
2.4	The gravitational potential function should work correctly	Pass
2.5	The field line function should work correctly	Pass
2.6	The test mass function should work correctly	Pass
2.7	The model should be of use within a classroom and at home	Pass

Some of the above requirements are objective, such as whether or not it is easily useable or if it is understandable, for these requirements the beta testing results were used to decide the outcome of the test.

This is a justifiable approach as the system's usefulness is also defined by how students (like those used for the beta testing) find the system, and its effectiveness in covering the subject content.

To ensure that the end user was content with the final product I contacted Mr Weston to assess how well it met the original system requirements.

He commented that he was mostly thrilled with the final product, stating that it was largely simple to use and had the potential to be very helpful within his tutorials. However he did mention that the functionality of the measuring tool could be improved along with a few minor graphical details: such as the field lines produced should technically go to infinity and how close to the masses the equipotential lines weren't incredibly clear. Although he continued to say that these issues would not be a problem and the key messages were transmitted.

LEARNING RESOURCE REQUIREMENTS

Requirement	Summary	Requirement met? (Pass/Fail)
1.0	The resource should be educational and useable at home and in a classroom	Pass
1.1	The information should cover the entirety of the subject content.	Pass
	The detail of the information must be sufficient to leave the students with a strong understanding of the subject.	Pass
2.0	There should be a range of suitable questions to answer	Pass
2.1	The questions should have a range of difficulties	Pass
2.1.1	All students should be able to answer the easiest questions	Pass
2.1.2	The most challenging questions should be testing for the majority of students	Pass
2.1.3	The questions should prepare student for common exam question scenarios	Pass
3.0	Is feedback available to the students?	Pass
3.1	The students should be able to see the percentage they achieve on each test.	Pass
3.2	The students should receive a feedback message from their teacher.	Pass

The requirements for the learning resource will always have an element of objectiveness as the outcome may vary from user to user depending on the experience they receive. However the results from the beta testing clearly showed that the learning resource contained a sufficiently broad range of testing and clear resources for educating the students on the subject content.

DATABASE REQUIREMENTS

Requirement	Summary	Requirement met? (Pass/Fail)
1.0	Database should be designed logically	Pass
1.1	All tables should be normalised	Pass
1.1.1	Tables should only contain atomised values	Pass
1.1.2	Values stored within the same column should be similar	Pass
1.1.3	Columns should be appropriately named	Pass
1.1.4	The order in which data should not matter	Pass
1.2	All stored data should be useful	Pass
2.0	The system must be able to interact with the database	Pass
2.1	The database must be created correctly	Pass
2.1.1	Each table must be created with the correct name	Pass
2.1.2	Each table must be created with the correct number of fields	Pass
2.1.3	All key fields must be defined	Pass
2.1.4	Each field must be correctly named	Pass
2.1.5	Each field must have the correct data type	Pass
2.2	The database should be able to store all necessary values	Pass

2.2.1	When an account is created the details must be stored	Pass
2.2.2	When a student's class code is changed the alteration must be made to the database	Pass
2.2.3	When a teacher adds a students work the results table should be correctly updated.	Pass
2.2.4	When a teacher adjusts a students feedback the changed must be updated	Pass
2.3	The database must be able to answer a range of queries	Pass
2.3.1	Data required to fill controls in the system must be gathered	Pass
2.3.2	Teachers must be able to view the feedback they have left students in their class	Pass
2.3.3	Teachers must be able to view and alter the data involving students within their class	Pass

It is essential to the function of large portions of the system that not only the database is created correctly but that the database can perform a range of SQL commands.

For these SQL commands it is imperative that all tables and fields are created accurately to the design with the correct identifiers and data types, otherwise multiple errors will be encountered when attempting to query fields that do not exist or input/read data of the wrong data type. For the robustness of the system these requirements must be met.

Beta Testing

Results from Beta Testing Questionnaires

Did you encounter any issues with the product?

During the beta testing a few minor issues were found relating to the overall performance of the product. Such issues include:

- getting some notifications multiple times (receiving the same message box multiple times).
- The simulation becoming slow after an extended period of use.

While neither of these issues are massive issues to improve the products ease of use they should be fixed or a method should be changed to avoid the issue occurring.

To what extent was the product easy to use?

Evidently with a product of such size with its numerous different components each aspect should be easy to use and be instantly understandable.

With this said it was found that students and teachers found certain aspects easier to use than others. By common consensus it was found that the beta testers found the simulation easiest to use. Although it should be noted that the results of the beta testing questionnaire show that the ease of use of the product was found to be average to above average.

How beneficial was the simulation to your (students) understanding of gravitational potential?

From the brief amount of time the students were able to utilise the system it would appear that it is indeed beneficial to understanding the concept of gravitational potential gravitational fields in general.

It may prove useful to expand the research into this question with a wider sample as the students used had already studied gravitational fields due to the stage of their learning, however it was stated that they thought the diagrams were accurate and helpful. This means that it is likely that the product would be beneficial to developing the understanding of gravitational fields.

Would you use a resource such as this within lessons?

Response to this question was very mixed with a roughly equal amount of responses saying yes and no. This shows that as roughly half the users agreed that it would be of use within lessons that it would be of benefit to teachers and students as it is a relevant result representative of a strong proportion of the population. While for the students who said they wouldn't use it as a resource it may still benefit their understanding of the topic

How easy were the accounts and results page to use?

This question had a larger range of answers averaging around the average mark, with a few below averages and above averages. This shows that the system is passable could be improved leading to a development in the product.

How easy was it to access results?

The students generally agreed it was very easy to access their results, and as the system simply displays all the students results this is the only real acceptable state.

On the other hand the teachers feedback, while positive, pointed out some difficulties where the instructions had not been clear or a wanted to query by both a student and a test and was unable to, while by design this could be a possible improvement.

Was there a suitable range of questions?

It was clear from the beta testing that there is indeed a suitable range of range of questions available. This definitively positive response showed that the questions available are suitable to the students requirements and the subjects specification. However the quantity of questions could always be increased to improve the diversity of questions available.

What improvements, if any, would you like to see implemented?

A few comments related to the overall performance of the product with the small number of issues found being fixed. Other than this there were a few requests for further developments such as a gravitational collapse simulation, or the large masses attracting each other, or more ways to query the database for information.

User feedback

The user feedback from the initial testing was overwhelmingly positive. With the students pleasantly surprised at the effectiveness of the simulation. The response to the visualisations were positive and excited the students into exploring and testing the bounds of the system.

Particularly positive functions included the gravitational potential mapping where the visualisation of equipotential lines was greatly appreciated (although if the screen is overpopulated with large masses then it causes a poor visual, as the screen is filled with red. While not an error and the system is still working it may be a future improvement to add scaling elements in order to make these circumstances possible to clearly visualise). However this would also make the program harder to use in normal situations as the masses would no longer have the same effect after the scaling parameters have been added.

Another popular function was the test mass function where the movement of the masses was found to be interesting. Quickly the students had discovered that the masses could be forced into an orbit if correctly placed in a multi mass system or the challenge of finding a Lagrangian point – where the resultant gravitational force is 0N and the test mass would hover in place.

In reality the gravitational field lines would in fact stretch out towards infinity, however due to the recursive process used without a geometrically increasing scaling factor this would be a challenge to implement, and it does not affect the system as it just shows that beyond that point the gravitational force is negligible.

Evaluation

To assess the successfulness of my solution it is important to evaluate how well it meets the specified requirements set at the beginning of the project. As seen in the requirements testing the program meets all the requirements in some aspects but do varying degrees, and it is that degree of success that I plan to discuss here.

As normal it is practical to split the solution up into its three separate parts; the simulation, the learning resource and the database back end. Firstly focusing on the simulation I would define this as the strongest success. With most requirements being comfortably met. The visualisations are clear and effective and while there is some scope for improvement these are only minor fine tunings, with the possible exception of the measuring tool.

This evaluation is supported by the results of the alpha and beta testing. From the alpha testing all the calculations that create the data required to produce all the visual outputs are accurate to within 1%, which covers the uncertainties from rounding errors. Positive feedback from the beta testing showed that the simulation aspect was clearly working well, a few requests for minor improvements or changes were left but no users found any error of large significance. The suggested improvements included some of the points I have already outlined such as the measuring tool; which I would like to rework given time, and in the case where the screen is overpopulated with large masses in close proximity then the visualisations produced are sometimes poor. However, there is no real fix for this as the user can continuously add masses and it is under the users control, so the only real solution is to limit how many masses a user can place.

If given the opportunity I would have liked to rework the measuring function to better emulate some of the other similar resources explored within the research where it is possible to measure the difference in potential between two points. In these cases, there is a specific drag and drop tool for this which makes use easy and entertaining. However, in these cases this tool forms a large part of the product and is essential to the use. In revisiting the project, it may be an idea to define the simulation as a class instead of a module, but this would require a large amount of refactoring of code, and I believe that for this purpose a module was better suited. In hindsight, I also believe I made the correct decision to calculate the field lines recursively as if I were to brute force them and select specific lines the simulation would take considerably longer to run. Though if it is possible I would like to find a proper way to calculate the equipotential lines instead of just grouping together potentials by magnitude and displaying them, which may solve the issues experienced close to a mass; where the equipotential lines become disjointed. However, this method would likely require higher levels of physics to implement and a physics engine to run.

As previously mentioned, Mr Wester was able to give feedback on the simulation and was ecstatic about how quick and easy it was to use, after a few minutes practice and exploring he was able to quickly set up a large multitude of interesting systems. He believed that the simulation strongly met the requirements it had been set and 'definitely could be used within a classroom. He did mention that it would be nice if the system was scalable and it was possible to change the scale so that larger systems could be made though it is not essential to the use of the program. This inclusion could be furthered by adding in velocities for each mass so that the model was able to simulation solar systems, however this alteration would undoubtedly make the program harder and slower to use as more details have to be set up before a simulation could be run. Which while may improve its usefulness in an all-day demonstration at a science exhibition, would make the program more cumbersome and less suited to a learning environment. One

thing that he would have liked to see is a visualisation of the potential wells from a side on view, which could be created given further time and was not specified within the requirements.

An area that I would consider reworking to a large degree is the learning resource, while it does meet its requirements and given the time frame is more suited to the client's needs I believe that a more comprehensive system may have had these aspects more implemented. With endless practice questions and answers generated by the system and the ability for the students to answer the tests online, reducing the environmental impact of each student printing multiple sheets of paper. This may also result in easing a teacher's workload as the system could mark the tests, meaning the teacher would only have to add some feedback.

From the testing results it has been ascertained that the resources supplied and the practice questions available are helpful and relevant to the course. Both resources supplying a good level of detail and leaving the students with a thorough understanding spanning the breadth of the subject.

During my Mr Wester's use of the final product he was able to explore the available lessons and question resources. By chance Mr Wester occasionally uses some of the same resources within his lessons, giving praise to Khan Academy and some of the YouTube videos attached. Suggesting that this product would enable him to spend less time using them within his lessons and enable him to show students some other important demonstrations. The large array of questions available on the system are straight from the AQA exam specification so they are completely relevant to the course, and the large volume available meant that all common exam scenarios that Mr Wester was aware of were covered.

While not given direct access to the database, during his time with the system Mr Wester was able to formulate several queries on a temporary account. Seemingly pleased with the opportunity to move more of the current system online and improving its organisation. He said that he was frustrated with having to keep note of which students had handed in what work and that it may be helpful to see which students had not completed the homework instead of just those who had. But other than that he did not have any other methods of querying results that he would add. This positive feedback of the results was reflected in the beta testing as no users struggled to use the system, finding little fault with the login services, results page and the teacher's version of the test page.

The more important aspect of the database is the database itself, which strongly meets all the requirements specified in the design. With the database being correctly created with all fields correctly named and of the data type required, it is fully normalised and has protection against common forms of fault for robustness. As a result, it can only be said that the database has been successful in meeting its requirements. However due to the system it is currently impossible to have multiple classes for one teacher, even though it is technically possible within the database and this capability should be included.

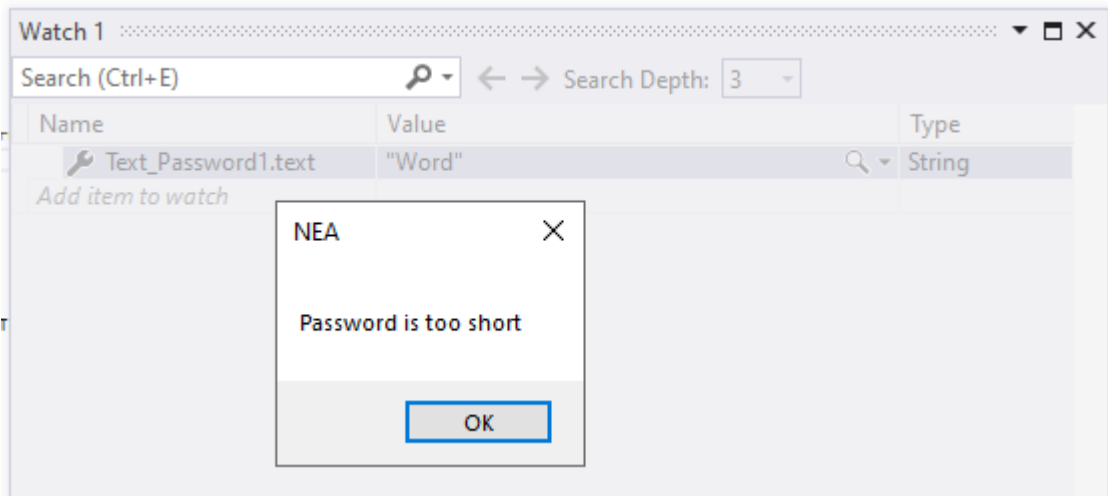
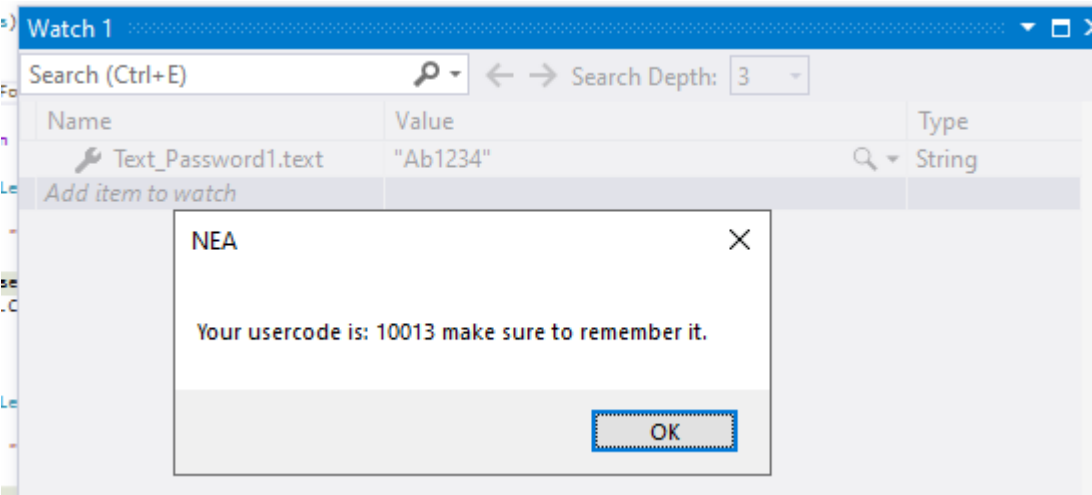
Appendix

Testing Video

To demonstrate parts of the programme screen captures have been used to show it is functioning correctly, this evidence has been uploaded to the following playlist on Youtube.

<https://www.youtube.com/playlist?list=PLHRLr1INrSQ-bce32fnUhFbC1qGJw2FM0>

Input testing

Index	Screenshot
I-1.1	<p>Validation of password</p>  <p>The screenshot shows a 'Watch 1' window with a search bar and a table. The table has columns 'Name', 'Value', and 'Type'. The first row shows 'Text_Password1.text' with value '"Word"' and type 'String'. Below the table is an 'Add item to watch' button. Overlaid on the watch window is a dialog box titled 'NEA' with the message 'Password is too short' and an 'OK' button.</p>  <p>The screenshot shows a 'Watch 1' window with a search bar and a table. The table has columns 'Name', 'Value', and 'Type'. The first row shows 'Text_Password1.text' with value '"Ab1234"' and type 'String'. Below the table is an 'Add item to watch' button. Overlaid on the watch window is a dialog box titled 'NEA' with the message 'Your usercode is: 10013 make sure to remember it.' and an 'OK' button.</p> <p>(The usercode message is a success as it means the account has been created)</p>
I-1.2	Validation of email, returns true if valid, false if not

Watch 1	
Search (Ctrl+E) Search Depth: 3 A	
Name	Value
System.Text.RegularExpressions.Regex.IsMatch(s, "^([0-9a-zA-Z]{1,}\\.\\w)*[0-9a-zA-Z]*@[0-9a-zA-Z]{1,}\\.([a-zA-Z]{2,9})\$")	True
s	"LBJ@gmail.com"
<i>Add item to watch</i>	

Watch 1	
Search (Ctrl+E) Search Depth: 3 A	
Name	Value
System.Text.RegularExpressions.Regex.IsMatch(s, "^([0-9a-zA-Z]{1,}\\.\\w)*[0-9a-zA-Z]*@[0-9a-zA-Z]{1,}\\.([a-zA-Z]{2,9})\$")	True
s	"Heresmyemail@gmail.com"
<i>Add item to watch</i>	

Watch 1	
Search (Ctrl+E) Search Depth: 3 A	
Name	Value
System.Text.RegularExpressions.Regex.IsMatch(s, "^([0-9a-zA-Z]{1,}\\.\\w)*[0-9a-zA-Z]*@[0-9a-zA-Z]{1,}\\.([a-zA-Z]{2,9})\$")	False
s	"Notanemail"
<i>Add item to watch</i>	

I-1.3

Watch 1	
Search (Ctrl+E) Search Depth: 3 A	
Name	Value
s	"James"
System.Text.RegularExpressions.Regex.IsMatch(s, "^([A-Za-z]+)\$")	True
<i>Add item to watch</i>	

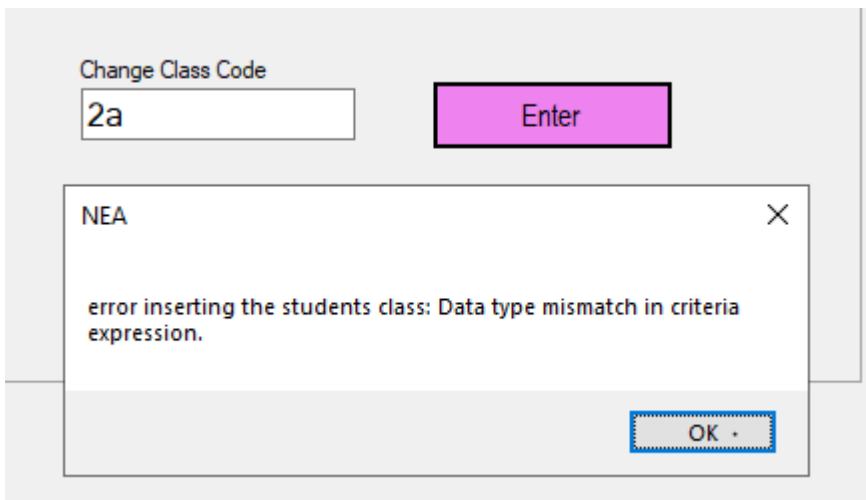
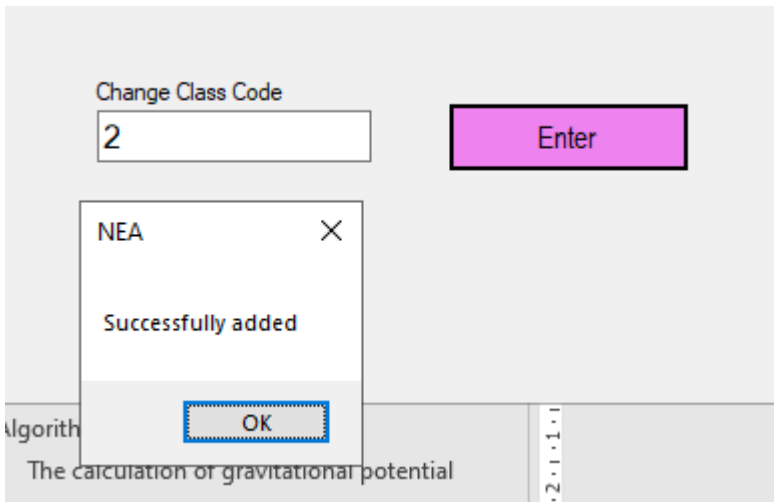
Watch 1	
Search (Ctrl+E) Search Depth: 3 A	
Name	Value
s	"James12"
System.Text.RegularExpressions.Regex.IsMatch(s, "^([A-Za-z]+)\$")	False
<i>Add item to watch</i>	

I-1.4

Watch 1	
Search (Ctrl+E) Search Depth: 3 A	
Name	Value
s	"Hancock"
System.Text.RegularExpressions.Regex.IsMatch(s, "^([A-Za-z]+)\$")	True
<i>Add item to watch</i>	

Watch 1	
Search (Ctrl+E) <input type="text"/> Search Depth: 3 <input type="button" value="A"/>	
Name	Value
5	"Hancock12"
System.Text.RegularExpressions.Regex.IsMatch(s, "[A-Za-z]+\$")	False
<i>Add item to watch</i>	

I-1.5



I-2.1

While not quite exactly the same as the expected results they are all equivalent so the test is passed

Watch 1	
Search (Ctrl+E) <input type="text"/> Search Depth: 3 <input type="button" value="A"/>	
Name	Value
loggedin	True
usercode	"10000"
PriveledgeLevel	Student {2}
<i>Add item to watch</i>	

I-2.2

--	--

Process Testing

Index	Screenshot
P-1.1	<p>Runtime watch of the Calculate Potential Function to ensure it is outputting the correct value</p>
P-1.2	<p>Runtime watch of the calculate force function to ensure correctness. Note, mass 2 is assumed to be 1kg in the formula.</p>
P-1.3	Test for the summation of gravitational force

```

717 Public Function CalculateForce(ByVal planets() As Planet, ByVal canvas As Environment) As Boolean
718 Dim CurrentDirection, CurrentMagnitude As Single
719 For Each planet In planets
720     If Not planet Is Nothing Then
721         CurrentDirection = _field.Direction 'saves old values of magnitude and direction so the vector sum can be
722         CurrentMagnitude = _field.Strength 'first time through the magnitude of the force will be 0 so there will
723
724         planet.CalculateForce(_location, _field) 'calculates the attraction from the next planet
725         'only need to sum two forces at a time
726         If CurrentDirection <> 9999 Then
727             _field.SumofGravitationalForce(_field, CurrentMagnitude, CurrentDirection) 'takes field byref so upda
728         End If
729     Else
730         _field.Strength += 0
731     End If
732 Next
733 Dim G As Single = 6.67 * 10 ^ -11
734 _field.Strength *= G
735

```

Watch 1

Search (Ctrl+E) Search Depth: 3

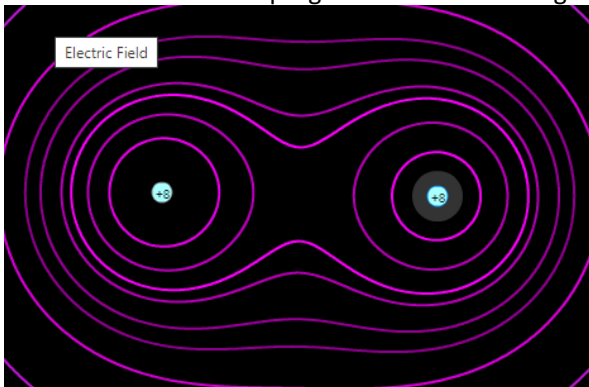
Name	Value	Type
Field.strength	6.67E-11	Single
planets(0).mass	1000000	Single
Planets(1).mass	2000000	Single
field.direction	3.14159274	Single

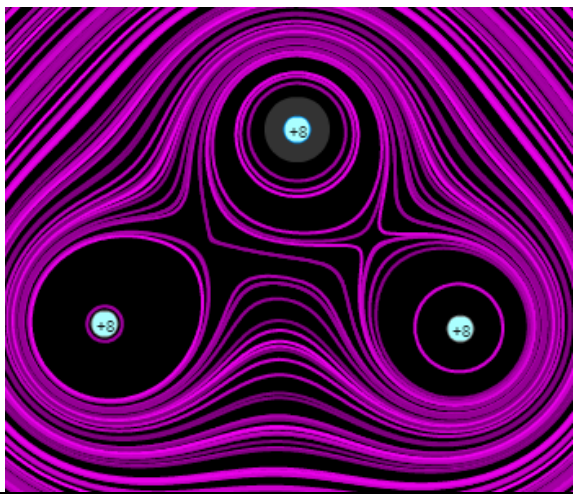
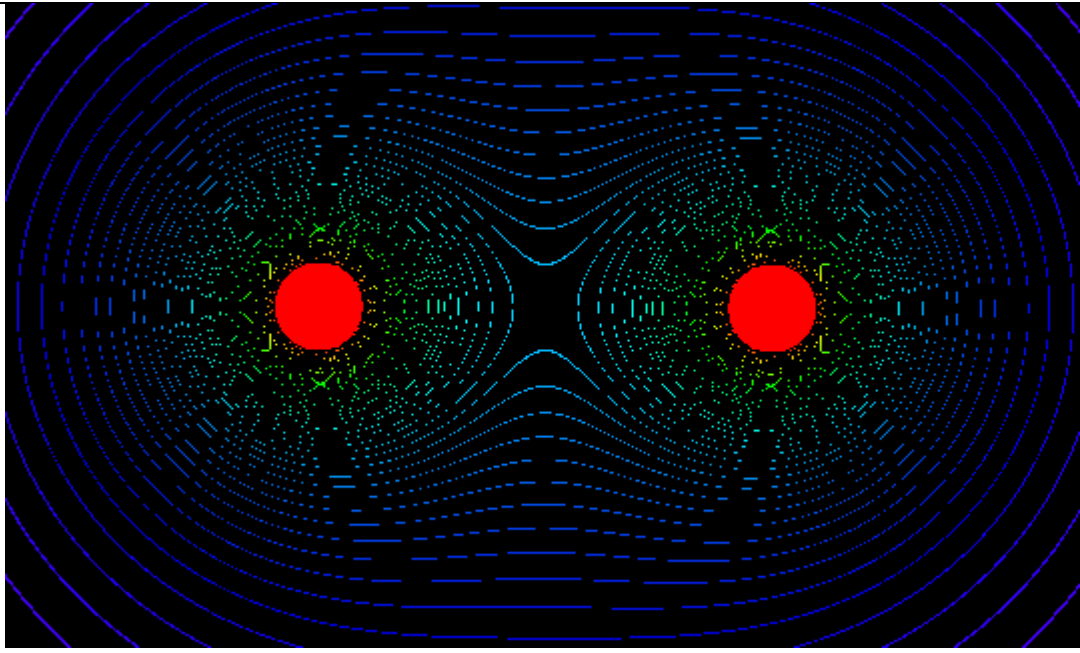
Add item to watch

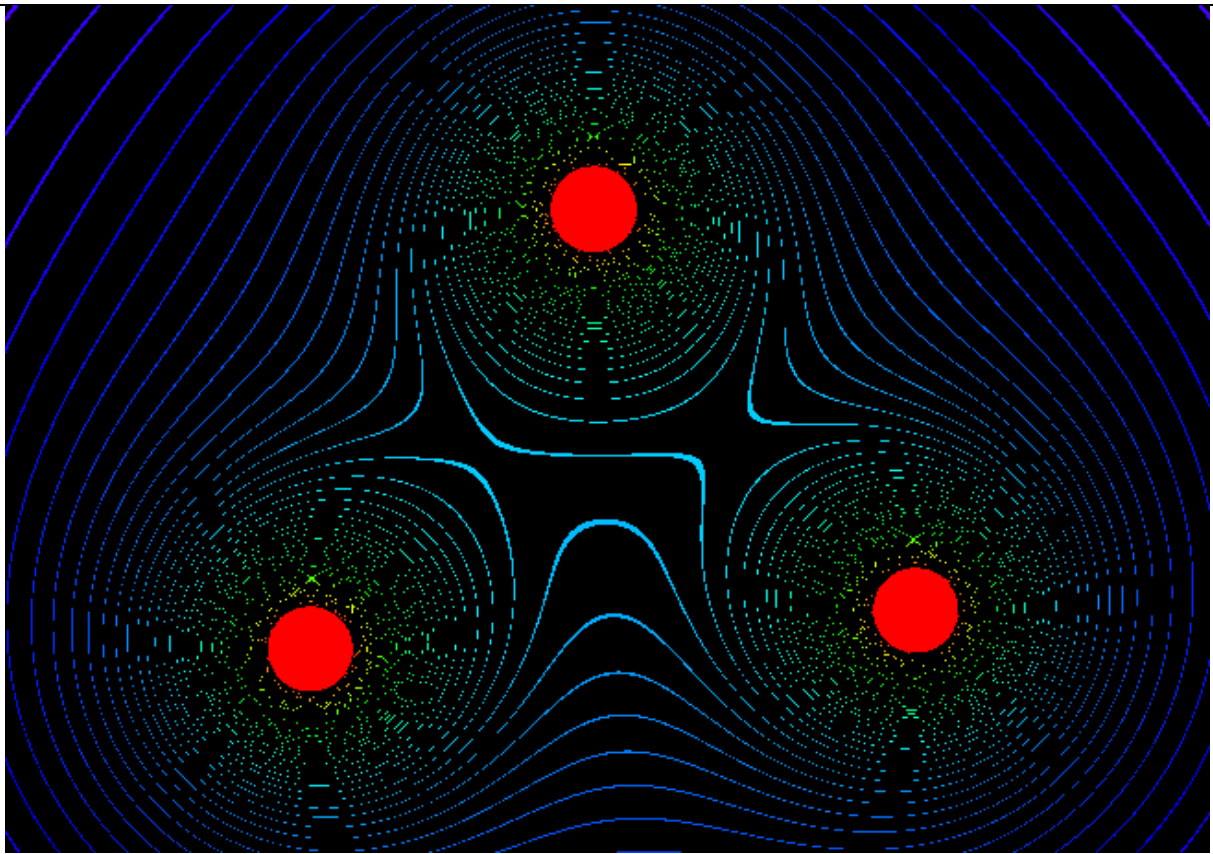
Note that the field direction is angled at 3.1415 radians to the horizontal which is approximately horizontal to the right. Which is consistent with how the masses were oriented. (size is proportional to mass)



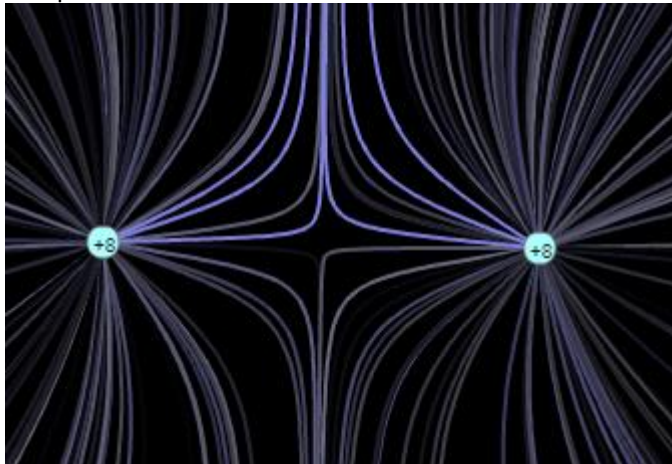
P-2.1 The screenshots of the programme should be vaguely similar to the following examples,

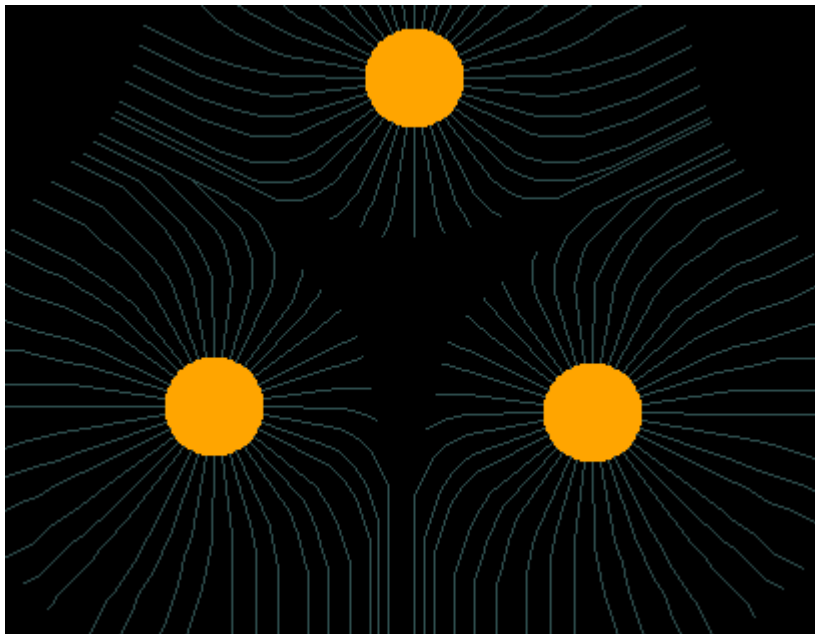
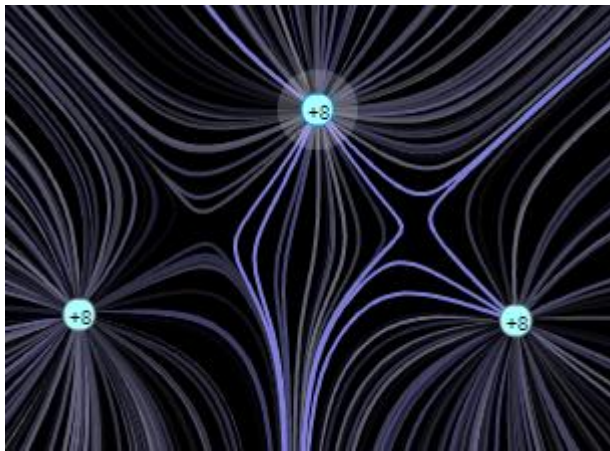
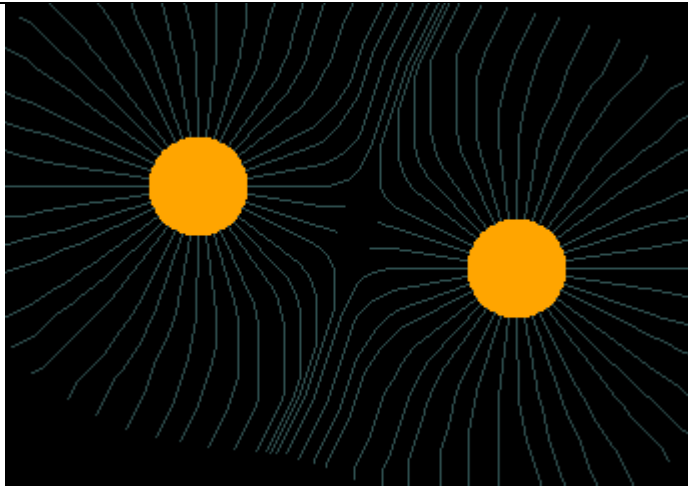






P-2.2 Like previous the screenshots should be close to these examples





P-2.3

Storage Testing

Index	Screenshot
S-1.1	
S-1.2	
S-1.2	
S-1.3	
S-1.4	
S-2.1	<p>Screen clipping of students table after data added</p>
S-2.2	<p>Screen clipping of the teacher table after data added</p>

Teachers				
TeacherCod	Forename	Surname	Email	Password
1002	Phillip	Fender	PF@godalming.ac.uk	PhysicsChallenge

S-2.3 Screen clipping of the classes table after teacher data added

Classes		
Classcode	TeacherCod	Click to Add
2	1002	