[Analysis 3](#_Toc478644396)

[Observation 6](#_Toc478644397)

[Document Specification Sheet (Worksheet) 7](#_Toc478644398)

[Requirements 12](#_Toc478644399)

[Design 13](#_Toc478644400)

[Possible solutions 13](#_Toc478644401)

[Prototype 14](#_Toc478644402)

[Data flow Diagram 14](#_Toc478644403)

[Top Level design 15](#_Toc478644404)

[Class Diagram 15](#_Toc478644405)

[Process Design 15](#_Toc478644406)

[Human Computer Interface 16](#_Toc478644407)

[IPSO 18](#_Toc478644408)

[Database design DDL Statements 18](#_Toc478644409)

[Technical Solution 20](#_Toc478644410)

[Simulation.html 20](#_Toc478644411)

[path\_to\_db\_connect.php 35](#_Toc478644412)

[phpServerContent.php 35](#_Toc478644413)

[Table Relationships 36](#_Toc478644414)

[Detailed Code Analysis 37](#_Toc478644415)

[SimData start and isNumberKey 37](#_Toc478644416)

[getCanvas and getContexts function 37](#_Toc478644417)

[Initialise,set,getQuestionsChosen 37](#_Toc478644418)

[Set,get FormHTML 38](#_Toc478644419)

[Set,get currentSimulation 38](#_Toc478644420)

[getRandom function 38](#_Toc478644421)

[getSliderValues function 38](#_Toc478644422)

[getQuestionType function 39](#_Toc478644423)

[Set,get UserID 39](#_Toc478644424)

[Set,get,and increment QuestionNumber 39](#_Toc478644425)

[Set questionsAnsweredCorrect, initialiseQuestionsAnswered, get questionsAnswered and questionsCorrect 40](#_Toc478644426)

[Set, get AlreadyClicked 40](#_Toc478644427)

[ClassBodyLarge properties 40](#_Toc478644428)

[ClassBodyLarge functions 41](#_Toc478644429)

[getRadius and getName 41](#_Toc478644430)

[getColour 42](#_Toc478644431)

[classBodySmall 42](#_Toc478644432)

[classBodyQuestion 42](#_Toc478644433)

[classSimulation 43](#_Toc478644434)

[Initialise 43](#_Toc478644435)

[NewSim 44](#_Toc478644436)

[GenerateQuestion 44](#_Toc478644437)

[ValidateUserDetails 45](#_Toc478644438)

[CapitalizeFirstLetter 45](#_Toc478644439)

[DebugData 46](#_Toc478644440)

[MoveProjectile 46](#_Toc478644441)

[dispLargeObj 46](#_Toc478644442)

[DrawProjectile 47](#_Toc478644443)

[SubmitAnswer 47](#_Toc478644444)

[IsCorrect 47](#_Toc478644445)

[EndSim 48](#_Toc478644446)

[PostUserData 48](#_Toc478644447)

[CreateCookie 48](#_Toc478644448)

[Php\_to\_database\_connect 48](#_Toc478644449)

[PHPServerContent 49](#_Toc478644450)

[Testing 50](#_Toc478644451)

[Youtube link 50](#_Toc478644452)

[Evaluation 53](#_Toc478644453)

# Analysis

The aim of this project is to create a program for the Godalming College Physics Department, which will include a gravitation simulator to teach students about the effects of gravity, and a question & answer section which links back to a database for self and teacher review of each student’s ability. This would be built to support all the college’s physics classes, of which there are 5 at A2, in which there are approximately 18 students per class, and 3 teachers that operate in all of them.

Currently, the college uses a paper based system for teacher review of ability, and internet simulations to illustrate gravitation alongside regular teaching. This can prove to be difficult as gravity on massive scales is difficult to demonstrate within a classroom, and that service may not always be provided by internet simulations, as they are not accessible if the internet is unavailable, and are not bespoke to the topics taught within the physics classroom. Paper based review is also somewhat unreliable, as they can be misplaced, unanswered or damaged, and as well as this is also time consuming to mark and evaluate upon. Many students are also visual learners, which means that explaining the theory in class, or using just paper based teaching may not be beneficial to every student learning.

I will investigate what a project of this scope will require by analysing the college’s current resources as well as interviewing a member of the Physics department staff. I feel this will give me an accurate standpoint on what the college has, and what parts of their current system would be analogous the project. By interviewing a member of staff, I am receiving one on one information and can therefore tailor the specification of my project more precisely to the needs of the Physics Department. I will make sure my questions during this interview are focused and are used to identify key query points within my project, so I can receive direct answers concerning the current system, and what they feel the benefits of a bespoke simulation would be.

I will examine and break down how the college receives performance data surrounding the students’ knowledge of gravity currently, and see how this could be improved. Often the systems are paper based and require a significant amount of time to mark down the number of correct answers and feedback for each student. By breaking this down into its constituent elements – Questioning, Data Storage, and Feedback, I can analyse how I can computerise the system. This would allow for good development within these areas, and would also allow for me to consider ease of replacement. That is, the difficulty surrounding migrating the college from a paper based system onto the electronic database, and issues surrounding it.

Question

Andy’s Response

Question Reasoning

Response analysis

Can you explain to me the need for physics simulations?

I asked this question to see why my project would be useful to the physics department, and what elements of a simulation are unique to it, i.e. cannot be replicated in real life.

The need for physics simulations is that you can only go so far with a laboratory, and for exploring the nature of gravitational fields- You can’t do that in the lab at all. You need a simulation to test your theories and see how they stand up.

From this response I can see that simulations provide something that practicals don’t, and this is especially the case in gravitational fields. This highlights the need for a project of this type.

What levels of simulations does the college possess currently?

I wanted to see and analyse what the college possesses, as this will give me a standpoint on what the college already has access to and what would be required in a new system. Furthermore, it gives me an idea as to where to start in a new system.

There are some simulation packs that we’ve bought, but they’re very old, so they might run on outdated, like Adobe, platforms instead of something newer. I’ve used maybe 3 or 4, but I know the other physics teachers use something called PhET which is the Colorado physics simulations which use shockwave and Javascript, but they’re somewhat simplistic in terms of the simulations and all internet based.

Looking at this response, I can see that the major simulations being used are from PhET. He has described the simulation packs the college possesses as outdated, and they almost solely use the Colorado simulations. I will investigate these further. Since the college is then relying upon Internet based simulations, these could be considered unreliable as they would cease to function without internet connection. Also, this gives me an idea as to what programming language would be useful, highlighting the Javascript may be the most appropriate language for the task.

What features would you require for it to be used as a teaching resource?

I asked this question as it gives me an idea as to what would be required in this product. This would allow me to fit it into my project scope, as it would be needed to fulfil the role of being useful as a teaching simulation.

Good resolution so that it is visible on a large board, easy to use interface, you want the simulation to be smooth to keep people interested.

From this I can see that I would likely need to use vector graphics to maintain the quality of image when displayed on a large screen. Furthermore, a descriptive and easily usable UI would be required for presenting, as well as a reasonably smooth (fast) real-time program as to ensure that it doesn’t slow down or appear ‘jittery’.

How would we use a gravitational field simulator in teaching?

This question was asked to find out what properties would be needed in a simulator to be useful. It would give me detailed responses and would help me write up the project scope by giving defined requirements.

You could then change the physical properties of say, the solar system and see how they react, make gravity weaker, or stronger, and see how it affects the trajectories of the planets as they tried to orbit, and that it would become a straight line if gravity was set to 0, or it gravity was strong how long it would take for orbits to decay. It would be quite interesting to see the effects between the sun and Jupiter.

From this I can see that altering variables and analysing their effects on planetary orbits could prove to be a viable teaching method. I may choose to implement sliders for values such as the gravitational constant, G, to analyse the effect it would have on orbits. Furthermore, I can see that affecting orbits in real time may prove to be a requirement and benefit to teaching.

What would you find useful about a question and answer system given that students could use on their own computers and then have the data fed back to you?

I asked this, as I feel that the question/answer system combined with the database is as important as the simulation when it comes to the amount of attention I need to focus on it. This question gives me the details and requirements I may need within this area of my program.

That would be very useful, as you would be able to see who could understand the physics behind the gravitation, and if it could also mark their score then you could see areas in which people were weak, and they don’t understand.

From this, I can see that the Q&A system would be useful to the teacher. This is reassuring, as I will be dedicating a lot of time to making this section function. From his response I can see that I would need to find a way to separate each question out by topic or question type, as this would allow him to see topic weakness in each student.

How easy is it to represent gravity as a whole without simulations?

I asked this question so I could see what elements in my project cannot be represented in real life. This gives me an idea towards how useful my project would be to the college and what they use for real life, as gravity is not the easiest to simulate without a computer.

Pretty difficult, you can show g by just dropping objects, and that they react under gravity, but if you were to go more advanced you would need a sheet of fabric and a heavy mass to represent the bending of space-time, otherwise it is very difficult.

From this I can see that the usage of practicals to show gravity is fairly limited. I will analyse a practical that they use concerning circular motion to see what this teaches students, and what elements of it may be useful in making my program.

Is it easy to tie in gravitation to circular motion?

I asked this question as it allows me to see if I would be able to mix in multiple topics in physics into this simulation. If the answer to this question is positive, I would be able to use circular motion style questions as part of my Q&A section.

Oh, very easy, in a simulation you could see it in front of your eyes, which helps explain it a lot better than just a teacher at the front of the classroom trying to explain it.

This response was unusual to analyse, as he does not answer the question directly. However, from the small amount of detail given, I think inserting circular motion questions into the simulation would be a good idea.

Are they limitations or complications you would expect to find in a simulation like this?

I asked this as a final question to see what he thought might be difficult to make within the simulation. This would give me a general area to focus on while making my project, so that I wouldn’t run into unexpected difficulties during programming.

Many variables would be a limitation, and the simulation would struggle a little bit – Allowing two heavy masses to interact requires a lot of calculation to work out their resultant velocity, also I can imagine it would be pretty tough to simulate many bodies at once, as this would also be tough on a processor. It would also be difficult to have two large fields affecting one object, as you would have to program the resultant force, and furthermore making this work for eccentricity and even the 3D plane.

From what I can infer from this, I can see one of my main problems will be real time calculations. These must be done quickly to ensure that the simulation doesn’t lag if multiple calculations are going on at once. If I were to tackle more than 3 celestial bodies at once this may introduce more issues due to the forces at work. I am not planning on simulating eccentricity and 3 dimensions, as I feel this goes beyond the requirements of the A level syllabus.

### Observation

From seeing Andy’s response, I believe that the college would make good use of a gravity simulator. This is apparent as the college doesn’t possess a suitable way to represent gravity when it is not assumed to be a uniform field. Also, whilst circular motion can be taught within class, it is sometimes not apparent to the students that this also applies to gravity in radial fields. Furthermore, the usage of a database for recording student performance would be far superior to what the college currently uses.

Some requests were made concerning what should be included in the program. Sliders for variables such as mass, or even the gravitational constant would provide interesting simulations and students could see how the values changing affects the orbits within the program. Also, some points concerning displaying the program on a large screen were apparent. The simulation should use high quality or even vector graphics so that it appears detailed on a large screen. Alongside this, a smooth simulation with little ‘jittery’ movements would make sure the program looked good when displayed to an audience.

### Document Specification Sheet (Worksheet)

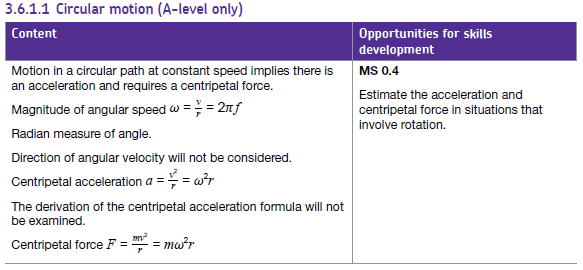
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Volumetrics** | | | | | | | | | | | |
| Document description | | | System | | | | Document | | Name | | Sheet |
| Assessment Questions | | | Paper and Database | | | | 1 | | Ciaran Quigley | | 1 |
| Stationery ref. | | | Size | | | | Number of parts | | Method of preparation | | |
| Worksheet and Database | | | A4 & Database | | | | 2 | | Written  Data is typed onto database | | |
| Filing sequence | | | | | Medium | | | | Prepared by | | |
| Scores => Database under student name | | | | | Paper and Computer | | | | Physics Teacher | | |
| Frequency of preparation | | | | | Retention period | | | | Location of file | | |
| During relevant Physics lessons  ~ 5 times per year across all classes | | | | | Data – 1 Year for results and comparable retests | | | | Database – Teacher’s Computer  Paper – Teacher’s Computer | | |
| Volume | Minimum | | | Maximum | | Av/Abs | | Growth rate/fluctuations | | | |
| 0 | | | 1 per student | | / | | / | | | |
| Users/receipts | | Purpose | | | | | | | | Frequency of use | |
| Physics Teacher | | To acquire scores and assess ability of students | | | | | | | | 1 per class | |
| Physics Students | | To test themselves using questions | | | | | | | | once | |
| **Data Dictionary** | | | | | | | | | | | |
| Ref | Name | Data Type | | | Regex | | | Occurrence | | Source of data / description | |
| 1 | Student Name | A | | | [^ ]{2,20} | | | No. of students | | Student writes name on worksheet | |
| 2 | Teacher Name | A | | | [^ ]{2,20} | | | 1 | | Teacher writes name on database | |
| 3 | Score | 9 | | | [1-9] | | | No. of students | | Number of questions answered correctly on sheet | |
| 4 | Correct Answers | Z,9 | | | . | | | No. of Questions | | Answer Sheet associated with Worksheet | |
| 5 | Answer given | Z,9 | | | . | | | No. of Questions | | Student writes answer on worksheet | |

Document Specification Sheet (Simulator)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Volumetrics** | | | | | | | | |
| Document description | | System | | | Document | | Name | Sheet |
| Simulator | | Javascript Simulation | | | 1 | | Ciaran Quigley |  |
| Stationery ref. | | Size | | Number of parts | | | Method of preparation | |
| Simulation | | Program | | 1 | | | Downloaded and run on computer | |
| Filing sequence | | | Medium | | | | Prepared by | |
| / | | | Computer | | | | Physics Teacher or student | |
| Frequency of preparation | | | Retention period | | | | Location of file | |
| During relevant Physics lessons  ~ 5 times per year across all classes | | | Not retained | | | | https://phet.colorado.edu/en/simulation/legacy/gravity-and-orbits | |
| Volume | Minimum | Maximum | | Av/Abs | | Growth rate/fluctuations | | | |
| / | / | | / | | / | | |
| Users/receipts | | Purpose | | | | | Frequency of use | |
| Physics Teacher | | To show gravitational interactions | | | | | 1-2 times per class per year | |
| Physics Students | | To see how gravity affects velocity | | | | | 1-2 times | |
| **Data Dictionary** | | | | | | | | |
| Ref | Name | Data Type | Regex | | | Occurrence | Source of data / description | |
| 1 | BodyName | A | [A-Z][a-z]{2,15} | | | No. of Bodies | Variable in Program | |
| 2 | Velocity | 0 | \d+\.?\d\* | | | No. of Bodies | Variable in Program | |
| 3 | BodyMass | 0 | \d+\.?\d\* | | | No. of Bodies | Variable in Program | |
| 4 | Time | 0 | \d+ | | | 1 | Variable in Program | |
| 5 | Force | 0 | \d+\.?\d\* | | | No. of Bodies | Variable in Program | |

From my investigation, I have seen how gravitation is currently taught, and information regarding student review is stored. From the interview and current documents, I have written a scope on what I would like my project to achieve. This includes automation of current systems, specifically with student review and assessment, as well as improvement upon the current teaching, such as variable altering and high visual quality. The main part of my project is a simulation, which displays an accurate model of celestial bodies, and their interactions with each other. This will include a path displayed showing the orbits of each body the end user will receive a UI that will be able to navigate and examine the database, allowing to filter queries to check student progress in certain topics contained within the simulation.

Since my program will be analysing key elements of the Physics A Level Syllabus, I have extracted the appropriate parts from the specification to break down into requirements.



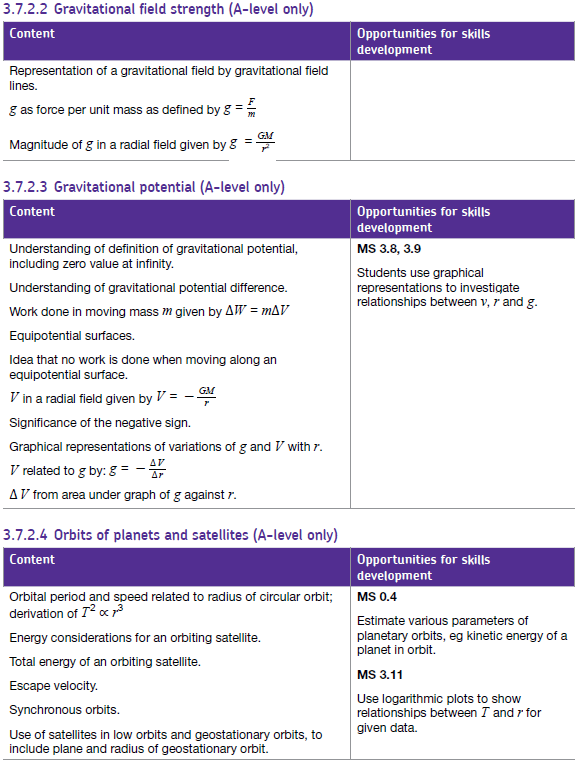
In this, I can see that the questions I would need to make sure were investigated would be

: The magnitude of angular speed

: The centripetal acceleration

: The centripetal force

Furthermore, answers to questions of angular speed should be answered in radians. This is useful, and I will apply it to my program.



I can see from these

# Requirements

1. Ease of Use
   1. Questions are easily understandable to any physics student
   2. Inputs are easily recognised and their purpose understood
   3. Simulation looks good when enlarged
      1. Vector Graphics to help when displayed on a large screen
      2. System moves smoothly, meaning that the simulation runs without jitters
         1. Calculations performed time efficient
2. Build Simulation
   1. Generate Celestial Bodies (sun, moon, planet, satellite)
      1. Different Simulation Types
         1. Planet & Satellite
         2. Planet & Moon
         3. Star & Planet
      2. Realistic Sizes
         1. Planets contain radii between 2 \* 107m – 3 \* 106m
         2. Stars contain radii between 1\* 1010m – 7\* 107m
         3. Satellites given approximate radii between 1 and 100m
   2. Give Realistic Velocities
      1. Should equal the orbital velocity at that height, meaning a circular orbit is created
   3. Plot Orbits
      1. Circular orbits so it’s easy to perform calculations on the simulation, including circular motion questions
      2. Line Drawn from moving Body
3. Create Assessment Questions
   1. Analyse the student’s ability using certain questions
      1. Random Questions
      2. Specific Questions
         1. Chosen by the user at the start of the program
   2. Receive Variables from Sim
      1. Round to Sensible Values (Whole number calculations)
   3. Display Question
      1. Question is easily understood by anyone reading it
4. Send Answers to Database
   1. Saves their individual score, and saves their score in each question and each question type.
   2. Has to be able to be easily connected to the actual college database
5. Develop Queries/Reports for Teacher
   1. Class Specific Averages
   2. Points of Focus Generated

# **Design**

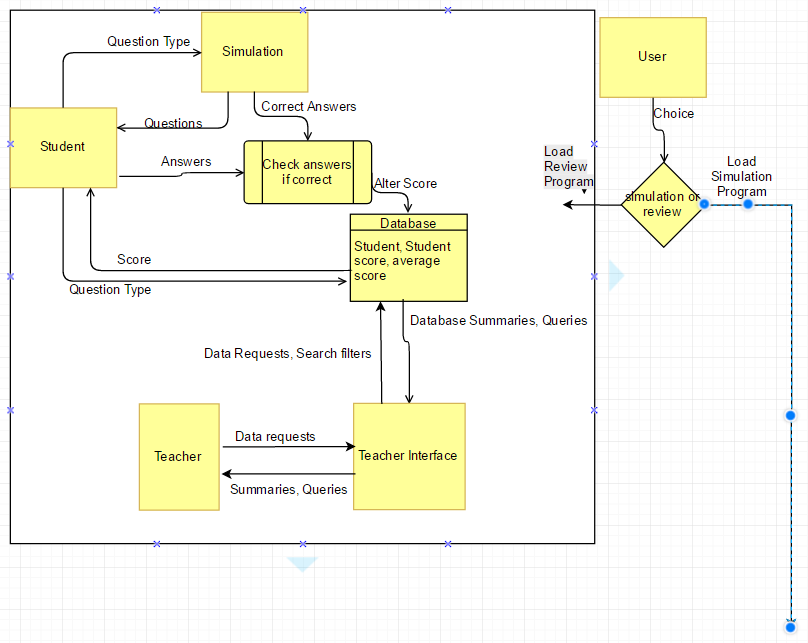
### Possible solutions

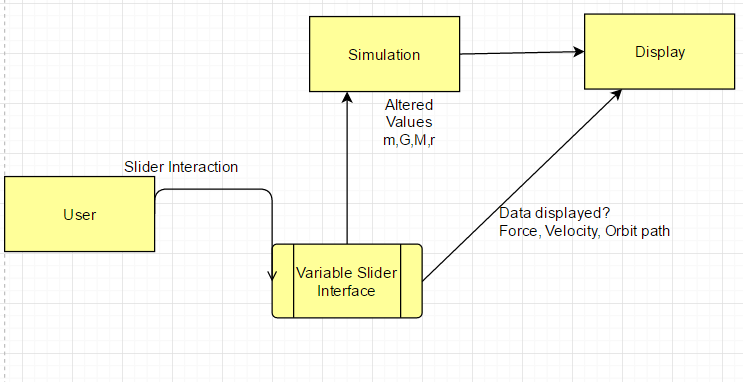
For my current investigation, I have seen how the college system for teaching gravitation works. From, this I can see that the college requires automation of their review system, as their current system relies upon paper based data, which is not easily organised or queried. Furthermore, in teaching, a simulation is required to be able to show the effects of gravity upon a system in real time. This displays gravity far better than current systems, which currently teaches circular motion and gravitation separately without providing appropriate linkage and equivalent visuals to highlight it. From this, I have devised some solutions that I could investigate. I will weigh the positives and negatives of each solution and compare before deciding on which to build my project upon.

1. A “sandbox” style simulation, allowing for control over celestial bodies with simple mouse control, such as dragging forward to enact a force on a celestial body. This would happen in a simulation containing 3 or more celestial bodies at a time. It would be a smooth interface, meaning that variables would be controlled by drag and click mouse movements on the simulation. Student review would be achieved by a set of multiple choice questions which would appear separate from the simulation. Upon finishing the test, a score would be displayed alongside their name. This data can then be collated by the teacher and stored in their current paper based system, which would ensure that there wouldn’t be any compatibility issues with the scoring and other paper based tests.
2. A simulation/review system containing a simulation of two celestial bodies. This could be altered using variable sliders and visual choices such as orbital path display and values associated with each body. This could be altered to be either for star-planet, planet-moon or planet-satellite interactions. The main section of the program would be the review system, which would randomly generate articulated mathematical questions, and alter the simulation to display the data given in the question. Answers would be received from the program and sent back to a database that the teacher could then use an interface to navigate. It would also provide an ability to add data not inserted by the system into the database, which would allow compatibility and therefore provide an easy transfer between the paper based system and computer based.

**Prototype**

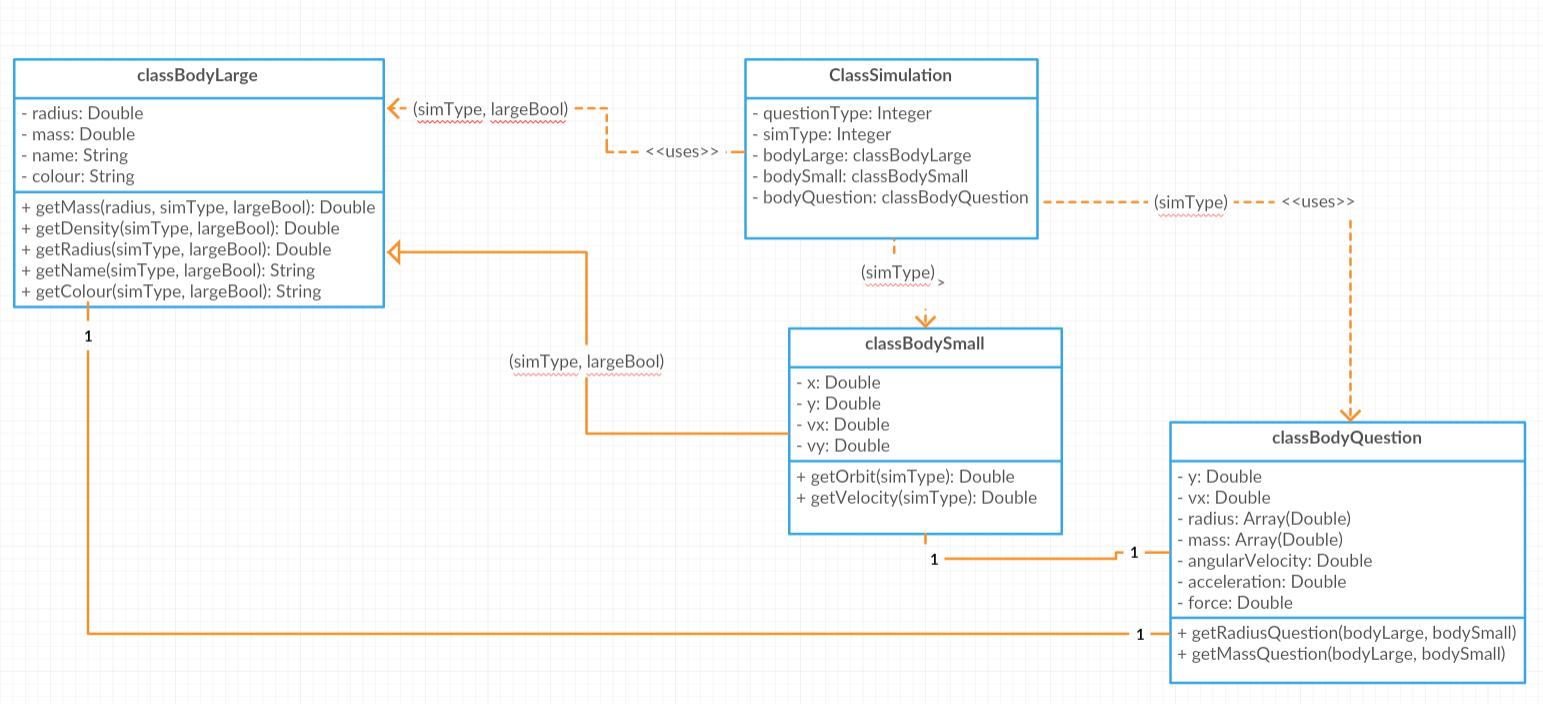
**Data flow Diagram**



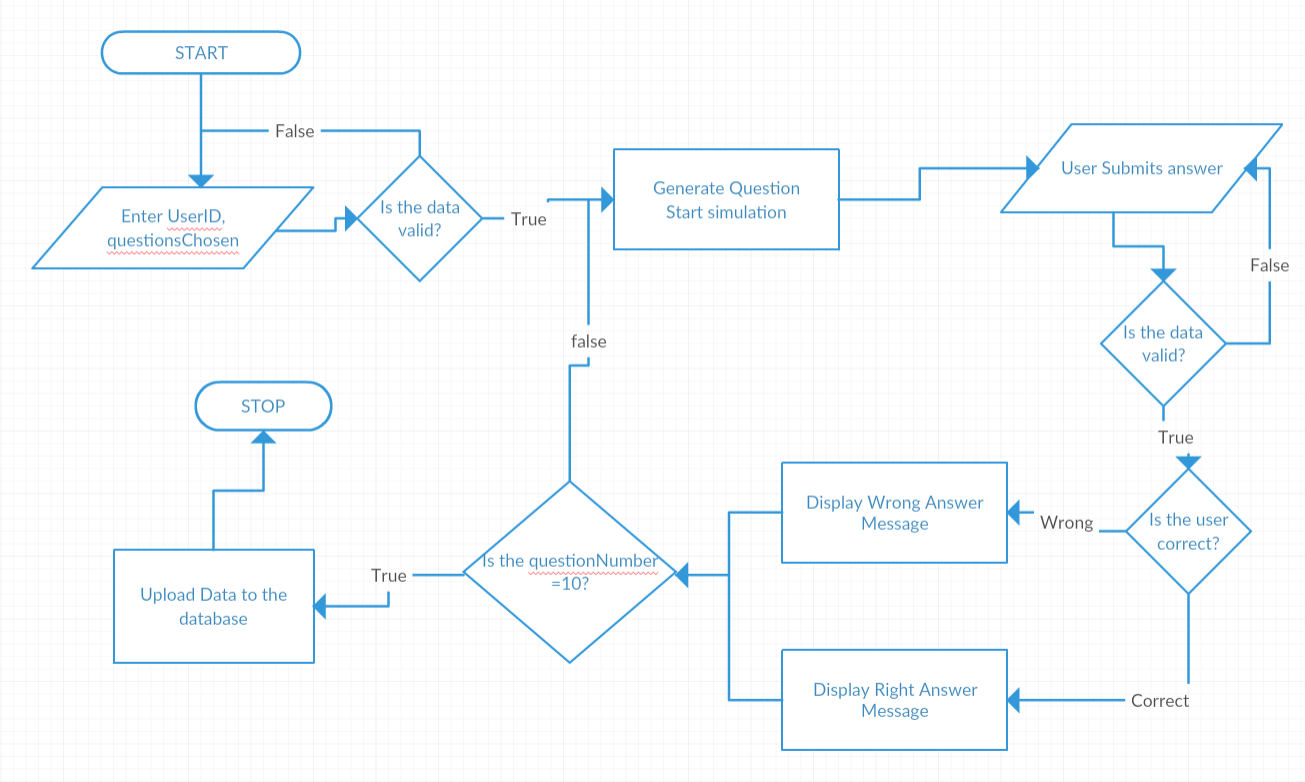


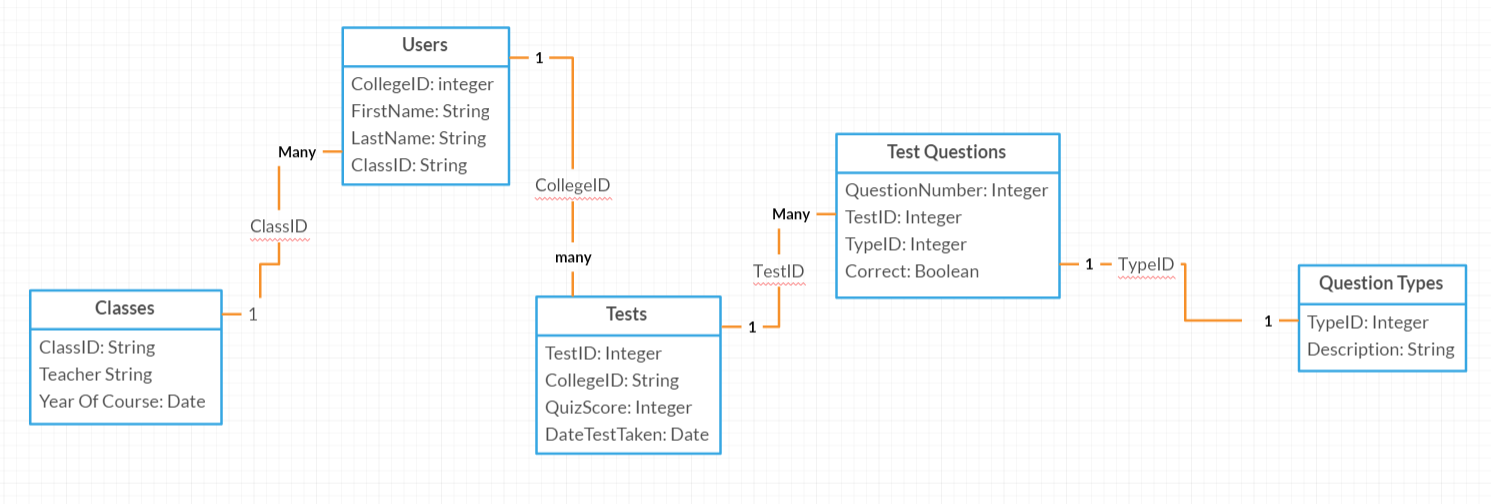
### **Top Level design**

**Class Diagram**



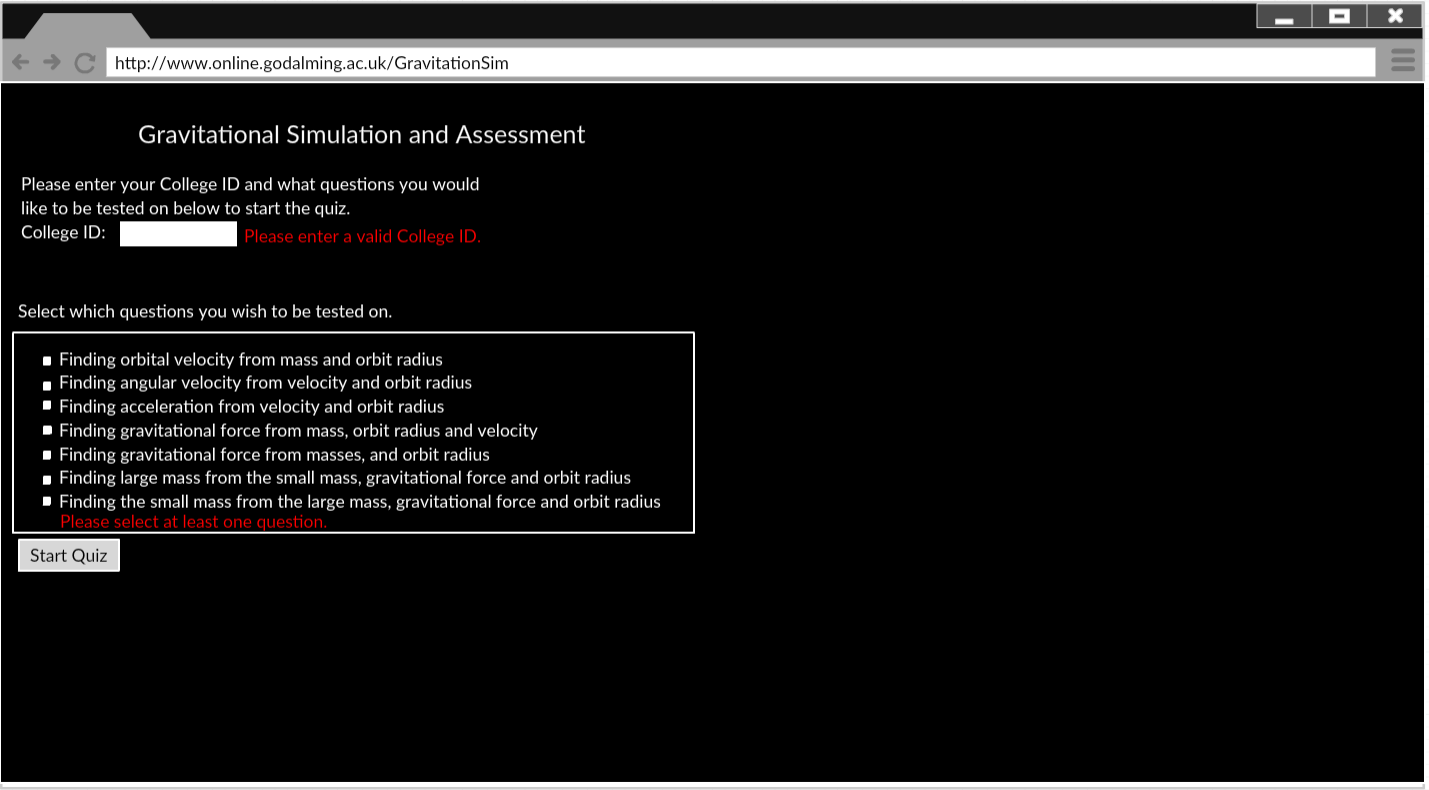
**Process Design**



**Relationship Diagram**

**Human Computer Interface**

**User Form**



1

6

5

4

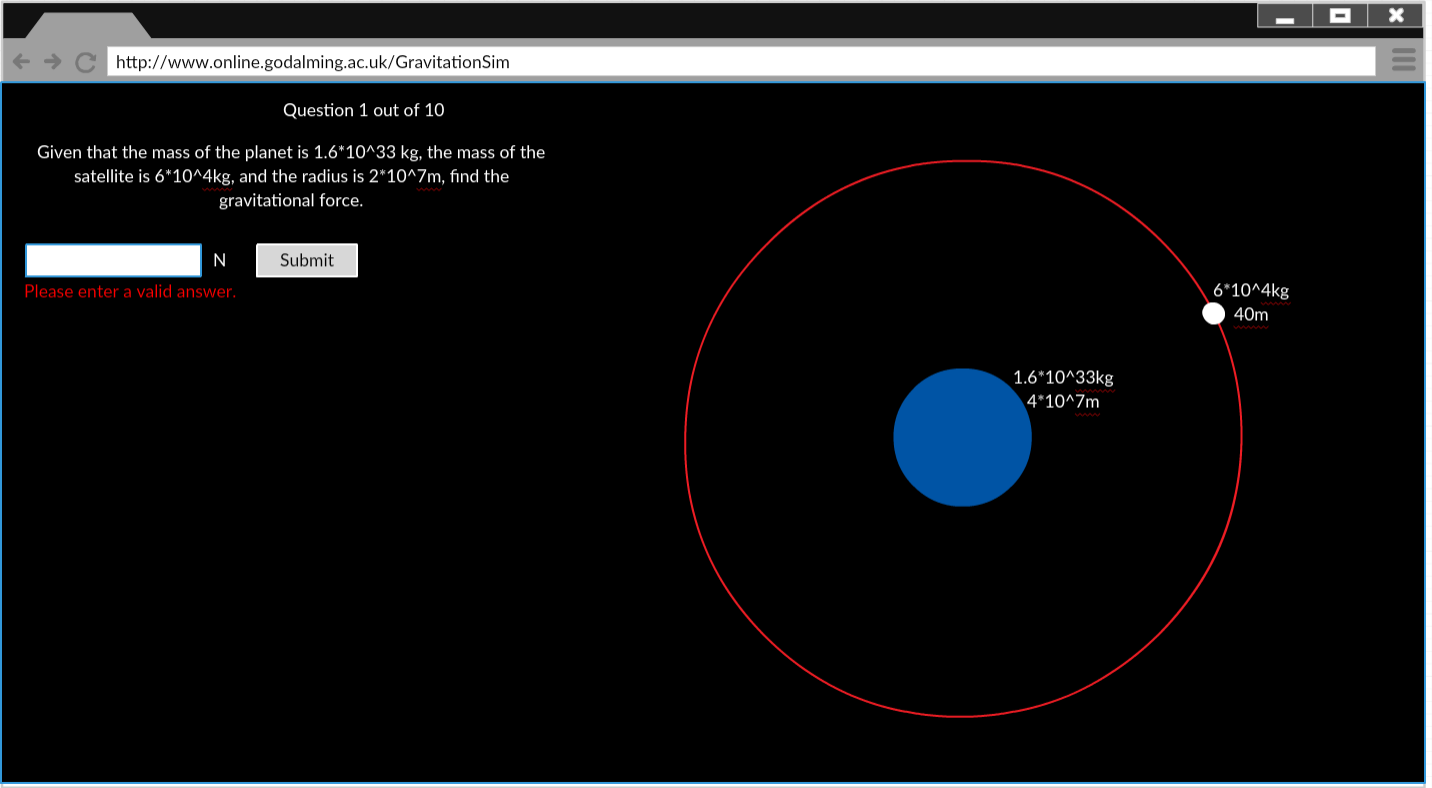
2

3

**Shown here, we have the mockup for my initial user form displayed for my program.**

1. **Title of the Program**
2. **Input box for CollegeID. This has been labelled with the tag “CollegeID:” so the user understands its purpose. This will have a 6 digit college ID entered into it.**
3. **Error message for CollegeID. If the textbox does not contain a valid 6 digit code, then the error will appear, and request that they enter a valid value.**
4. **Box containing the checkboxes for each question. 7 questions are listed, each with their own checkbox. These can be checked and unchecked according to which sort of questions the user would like to be tested on.**
5. **Error message for the checkboxes. If no checkboxes are ticked, this message will display, telling them that they are required to select at least one checkbox.**
6. **Start Quiz button. This performs a validation check on the inputs, if they don’t follow the required input then the respective error messages are displayed. If they are valid, this button starts the quiz.**

**Simulation**

****

8

6

7

5

4

3

2

1

**Shown here, we have the mockup for the quiz part of my program.**

**Annotations**

1. **Question Number out of 10 displayed at the top of the program**
2. **Question displayed. Each piece of required data has been listed in the question, and it is written in an understandable and comprehensible format.**
3. **Enter box for the answer. Data would be accepted in both normal form and exponent form. A textbox label displays the unit next to the answer box.**
4. **An error message if the enter box has invalid data written into it. This will instruct the user to enter a valid answer.**
5. **The submit button. This is pressed once the user has entered an answer. If the contents of the answer box is invalid, the error message is displayed. If the answer is numeric and wrong, it will come up with a message saying “Wrong” before going onto the next question. If the answer is correct, it displays “correct” before advancing.**
6. **The small body. It is displayed in a colour relating to the object it is portraying, white for satellite, grey for moon. It moves around the large body in a circular orbit. Its radius and mass display as a label, moving alongside it.**
7. **The large body. It is displayed in a colour relating to the object it is portraying, blue for planet, and yellow for star. It stays stationary in the centre. Its radius and mass display as a label.**
8. **The path. This is a red line that is drawn by the small body, indicating the path that the small body has taken.**

**IPSO**

|  |  |
| --- | --- |
| **Input** | **Process/Decision** |
| * **College ID – Integer** * **Questions Selected – Array of Boolean** * **Question Answers - Integer** | * **Check if CollegeID and Questions Selected are valid** * **Generate question** * **Display simulation according to selected question** * **Find Correct Question Answer, compare with student’s answer, return correct or wrong** |
| **Storage** | **Output** |
| * **Stores all the users – connects to the college database** * **Stores all the user tests – quiz score, and types of questions and if they got them right** * **Stores details of each type of question** * **Stores details of each class** | * **Correct/false messages if they get it right or wrong during quiz** * **Error messages if they enter something invalid as an answer/ during the form** * **A simulation based on the question generated, accurate to the data given** * **Output to the database** * **Outputs a summary of the student response** |

### **Database design** DDL Statements

CREATE TABLE `classes` (

`ClassID` varchar(5) NOT NULL PRIMARY KEY,

`Teacher` varchar(3) DEFAULT NULL,

`YearofCourse` varchar(4) DEFAULT NULL

)

CREATE TABLE `questiontypes` (

`TypeID` int(11) NOT NULL PRIMARY KEY,

`Description` varchar(30) DEFAULT NULL

)

CREATE TABLE `testquestions` (

`QuestionNumber` int(11) NOT NULL PRIMARY KEY,

`TestID` int(11) NOT NULL PRIMARY KEY,

`TypeID` int(11) NOT NULL,

`Correct` tinyint(1) NOT NULL

)

CREATE TABLE `tests` (

`TestID` int(11) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

`CollegeID` int(11) NOT NULL,

`QuizScore` int(11) NOT NULL,

`DateTestTaken` date DEFAULT NULL

)

CREATE TABLE `users` (

`CollegeID` int(11) NOT NULL PRIMARY KEY,

`FirstName` varchar(10) DEFAULT NULL,

`LastName` varchar(15) DEFAULT NULL,

`ClassID` varchar(5) DEFAULT NULL

)

ALTER TABLE `testquestions`

ADD CONSTRAINT `testquestions\_ibfk\_2` FOREIGN KEY (`TypeID`) REFERENCES `questiontypes` (`TypeID`),

ADD CONSTRAINT `testquestions\_ibfk\_3` FOREIGN KEY (`TestID`) REFERENCES `tests` (`TestID`);

ALTER TABLE `tests`

ADD CONSTRAINT `tests\_ibfk\_1` FOREIGN KEY (`CollegeID`) REFERENCES `users` (`CollegeID`);

ALTER TABLE `users`

ADD CONSTRAINT `users\_ibfk\_1` FOREIGN KEY (`ClassID`) REFERENCES `classes` (`ClassID`);

INSERT INTO `questiontypes` (`TypeID`, `Description`) VALUES ('1', 'Given MR find V'),

('2', 'Given VR find W'),('3', 'Given VR find A'),('4', 'Given MVR find GF'),

('5', 'Given M1M2R find GF'),('6', 'Given GFM1R find M2'),('7', 'Given GFM2R find M1')

# **Technical Solution**

#### **Simulation.html**

<!DOCTYPE html>

<html>

<head>

<meta charset=**"utf-8"**>

<title>**Newton's Cannon**</title>

</head>

<script src=**"http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"**>

</script>

<body style=**"font-family:sans-serif; font-size:15px; width:600px; margin-left:0; margin-right:auto; background-color:#000000; float=left"**>

<h1 style=**"font-size:24px; text-align:center; color:#ffffff"**>

**Gravitation Simulation and Assessment**

</h1>

<div id=**"userForm"** style=**"top:0px; left:0px;"**>

<p id=**"questionCounter"** style=**"font-size:18px; text-align:right; color:#ffffff;"**>

</p>

<div style=**"width:1100px; margin-left:auto; margin-right:auto; color:#ffffff; float=left"**>

<canvas id=**"lrgCanvas"** width=**"1000"** height=**"1000"** style=**"position:absolute; top:0px; left:20%; z-index:1"**>

**Canvas not supported; please update your browser.**

</canvas>

<canvas id=**"trailCanvas"** width=**"1000"** height=**"1000"** style=**"position:absolute; top:0px; left:20%; z-index:2"**>

</canvas>

<canvas id=**"theCanvas"** width=**"1000"** height=**"1000"** style=**"position:absolute; top:0px; left:20%; z-index:3"**>

</canvas>

<div id=**"sideBar"** style=**"font-size:15px; text-align:left; color:#ffffff; width:750px; float=left; z-index:4;"**>

<p id=**"txtQuestion"**>

</p>

<p id=**"txtValues"**>

</p>

<input type = **"text"** id = **"txtAnswer"** value = **""** onkeypress=**'return simData.isNumberKey(event) '**/><span id=**"answerUnit"** ></span>

<button type = **"button"** id = **"btnSubmit"** onclick=**"submitAnswer(txtAnswer.value)"**>**Submit**</button>

<p style=**"font-size:9px"**>**Note: exponential form e.g. (1.123e+29) are accepable answers** </p>

<span id=**"txtAnswerError"** style=**"color:#ff0000"**></span>

<p> **Change Speed of Simulation:** <br>

<input type=**"range"** name=**"dt"** id=**"difTime"** value=**"50"** min=**"10"** max=**"1000"**></p>

</div>

</div>

</div>

<script>

//ALL CALCULATIONS DONE IN SI (Meters, Seconds, Kilograms)

***const*** newtonG **=** 6.67e**-**11**;** // grav. constant in SI unit

simData **=** **{**

/\*Storing all variables that are needed across multiple set,get and initialise functions

in one globally defined object to declutter the global workspace

and keeping the variables private, keeping my code Object Oriented

by utilising encapsulation \*/

isNumberKey**:** ***function*(**evt**){**

***var*** charCode **=** **(**evt.which**)** **?** evt.which **:** evt.keyCode**;**

***if*** **((**charCode **!=** 46 **&&** charCode **!=** 43 **&&** charCode **!=** 101**)** **&&** charCode **>** 31

**&&** **(**charCode **<** 48 **||** charCode **>** 57**))**

***return*** ***false*;**

***return*** ***true*;**

**},**

getCanvases**:*function*()** **{**

/\*Initialises the canvases upon call, returning the canvases in an

array\*/

***var*** canvases **=** ***new*** Array**(**3**)**

canvases**[**0**]=**document.getElementById**(**"theCanvas"**);**

canvases**[**1**]=**document.getElementById**(**"trailCanvas"**);**

canvases**[**2**]=**document.getElementById**(**"lrgCanvas"**);**

***return*** canvases //aggregation of all canvases

**},**

getContexts**:*function*(){**

***var*** context **=** ***new*** Array**(**3**)**

/\*Calls getCanvases() to recieve an array of initialised canvases

gets the context from them and returns it in an array\*/

canvases **=** this.getCanvases**()**

***for*(**x**=**0**;**x**<=**2**;**x**++){**

context**[**x**]=**canvases**[**x**].**getContext**(**"2d"**);**

**}**

***return*** context //aggregation of all contexts

**},**

clearAll**:*function*(){**

/\*Clears a rectangle the size of the canvas to empty all canvases for next simulation\*/

context**[**0**].**clearRect**(**0**,**0**,**theCanvas.width**,**theCanvas.height**);**

context**[**2**].**clearRect**(**0**,**0**,**lrgCanvas.width**,**lrgCanvas.height**);**

context**[**1**].**clearRect**(**0**,**0**,**trailCanvas.width**,**trailCanvas.height**);**

**},**

initialiseQuestionsChosen**:** ***function*(){**

questionsChosen **=** ***new*** Array**(**7**);**

**},**

setQuestionsChosen**:*function*(){**

//Gets the checkbox data from the the form, storing it in an array, the array index corresponds to the question type

***for*(**x**=**0**;** x **<=** 6**;** x**++){**

questionsChosen**[**x**]** **=** document.startForm.questionTypes**[**x**].**checked**;**

**}**

**},**

getQuestionsChosen**:** ***function*(){**

/\*Returns the questionsChosen array\*/

***return*** questionsChosen**;**

**},**

setformHTML**:** ***function*(){**

/\* saves the current form to a variable, to be copied back in later \*/

formHTML **=** document.getElementById**(**"userForm"**).**innerHTML**;**

**},**

getformHTML**:** ***function*(){**

/\* retrieves the previous form \*/

***return*** formHTML**;**

**},**

setCurrentSimulation**:** ***function*(**simulation**){**

/\* gets the simulation (private object) and sets currentSimulation (public object)

equal to it, so it can be retrieved by the answering functions\*/

currentSimulation **=** simulation**;**

**},**

getCurrentSimulation**:** ***function*(){**

/\* retrieves currentSimulation from the object to be used in the answer functions \*/

***return*** currentSimulation**;**

**},**

getRandom**:** ***function*(**maxVal**,**minVal**)** **{**

//Generalised Random number generator, recieves a high and low value. Used to stop having to call similar randomisation statements

***return*** **((**Math.random**()\*(**maxVal**-**minVal**))** **+** minVal**);**

**},**

setSliderValues**:** ***function*(**simType**)** **{**

/\*After finding the simulation type, speed slider is adjusted to suitable min, max and default values

so the object moves at appropriate speed\*/

***switch*(**simType**)** **{**

***default*:**difTime.max **=** 200**;**

difTime.min **=** 0.1**;** //slowest speed values given to the satellite simulation, which orbits with v.low orbit time

difTime.value **=** 10**;**

***break*;**

***case*** 1**:** difTime.max **=** 1000**;**

difTime.min **=** 20**;** //middle speed values given to moon simulation, with moderate orbit time

difTime.value **=** 200**;**

***break*;**

***case*** 2**:** difTime.max **=** 2000**;**

difTime.min **=** 50**;** //high speed values given to star simulation, with v.long orbit times

difTime.value **=** 300**;**

**}**

**},**

getQuestionType**:** ***function*(**questionsChosen**)** **{**

/\* Randomly selects all 10 question types

Initialises the questionType variable with an invalid number\*/

***var*** questionType **=** 10 **;**

//Initialises a boolean flag to indicate when a valid question has been found

***var*** questionTypeFound **=** ***false*;**

//Loops until a valid question is chosen

***while*(**questionTypeFound **===** ***false*)** **{**

questionType **=** Math.floor**(**simData.getRandom**(**6.9999999**,**0**));** //gets a random question number between 0 and 6

***if*(**questionsChosen**[**questionType**]** **===** ***true*){**

questionTypeFound **=** ***true*** **;** //returns true if questionType is one of the question types selected by the user

**}** ***else*** **{**

questionTypeFound **=** ***false*;** //returns false if otherwise

**}**

**}**

***return*** questionType**;** //returns value

**},**

setUserID**:** ***function*()** **{**

userID **=** document.getElementById**(**"userid"**).**value**;** //setter for UserID, gets value from textbox on initialisation

**},**

getUserID**:** ***function*()** **{**

***return*** userID**;** //getter for UserID

**},**

setQuestionNumber**:** ***function*()** **{**

questionNumber **=** 0**;** //intialiser for questionNumber at 0, to be incremented

**},**

incrementQuestionNumber**:** ***function*()** **{**

questionNumber **+=** 1**;** //increments questionNumber upon calling

**},**

getQuestionNumber**:** ***function*()** **{**

***return*** questionNumber**;** //getter for questionNumber

**},**

setQuestionsAnsweredCorrect**:** ***function*(**questionNumber**,**isCorrect**,**questionType**){**

questionsAnswered**[**questionNumber**]** **=** questionType**;** //setter for questionsAnswered and questionsCorrect

***if*(**isCorrect **==** ***true*)** **{** //sets value at (questionNumber) of the array

questionsCorrect**[**questionNumber**]** **=** 1**;**

**}** ***else*** **{** //questionsAnswered recieves and stores questionType

questionsCorrect**[**questionNumber**]** **=** 0**;**

console.log**(**"isCorrect = false"**);** //questionsCorrect recieves and stores boolean Correct/Wrong (1/0)

**}**

**},**

initialiseQuestionsAnswered**:** ***function*(){**

questionsAnswered **=** ***new*** Array**(**10**);** //initialiser for questionsAnswered

questionsCorrect **=** ***new*** Array**(**10**);** //and questionsCorrect as an array of size 10

**},**

getQuestionsAnswered**:** ***function*(){**

***return*** questionsAnswered**;** //getter for questionsAnswered

**},**

getQuestionsCorrect**:** ***function*(){**

***return*** questionsCorrect**;** //getter for questionsCorrect

**},**

setAlreadyClicked**:** ***function*(**clickedBool**){** //setter for alreadyClicked

alreadyClicked **=** clickedBool**;**

**},** //alreadyClicked serves as a flag for if the button has already been

getAlreadyClicked**:** ***function*()** **{** //clicked so it can't be spammed

***return*** alreadyClicked**;** //getter for alreadyClicked

**}**

**}**

initialise**(*true*);** //starts the whole program

***class*** classBodyLarge **{**

//properties

constructor**(**simType**,**largeBool**)** **{**

this.radius **=** this.getRadius**(**simType**,**largeBool**);** //all properties call their respective get function

this.mass **=** this.getMass**(**this.radius**,**simType**,**largeBool**);** //all pass in the simType and largeBool variables to indicate what it's getting

this.name **=** this.getName**(**simType**,**largeBool**);**

this.colour **=** this.getColour**(**simType**,**largeBool**);**

**}**

/\*ecapsulated functions

throughout all of the get functions for this class, if the value is invalid it will

default to the satellite/planet simulation. This improves the reliability of the code

as it means that there's no case that would cause errors.

\*/

getMass**(**radius**,**simType**,**largeBool**){** //getter for the mass

***return*** **(**Math.pow**(**radius**,**3**)\***4**/**3**\***Math.PI **\*** this.getDensity**(**simType**,**largeBool**));**

//calculating a suitable mass by multiplying the density by the volume of the object

**}**

getDensity**(**simType**,**largeBool**)** **{** //getter for the density

***if*** **(**largeBool **=** ***true*)** **{** //Large body densities

***switch*(**simType**)** **{**

/\*returns suitable densities for planetary (rocky) bodies, I researched high and low values, using

Ceres as a basis for minimum density, and approx 1.2\*Earth density as max density\*/

***default*:**console.log**(**"Planet density"**);**

***return*** simData.getRandom**(**6500**,**2000**);**

***break*;**

/\*returns suitable densities for stellar bodies, I researched high and low values, using

the sun as the max density, and then using a slightly lower for the min. I decided to limit

the diversity of star densities, as they vary wildly according to stellar type and that was beyond

the specification of my project\*/

***case*** 2**:** console.log**(**"sun density"**);**

***return*** simData.getRandom**(**1500**,**1000**);**

**}}** ***else*** **{**

***switch*(**simType**)** **{** //Small body densities

***default*:** console.log**(**"satellite density"**);**

***return*** 500**;**

/\*Gives the approx density of a satellite, I used the ISS as a basis for this number, taking the volume as

1000m^3, and the mass as 500000kg, leading the density to be 500kg/m^3.

This is used as a static number for ease of simulation\*/

***break*;**

***case*** 1**:**

***case*** 2**:** console.log**(**"Planet density"**);**

***return*** getDensity**(**3**,*true*);**

//Calls itself to get a planetary density

***break*;**

**}**

**}}**

getRadius**(**simType**,**largeBool**){** //getter for the radius

***if*** **(**largeBool **===** ***true*)** **{**

***switch*(**simType**)** **{**

***default*:** console.log**(**"Planet radius"**);**

***return*** simData.getRandom**(**2 **\*** Math.pow**(**10**,**7**),**3 **\*** Math.pow**(**10**,**6**));**

/\*Planet Radii, I researched high and low values

for rocky planets, using mercury as min radius (approx)

and using approx 3\* earth radius as max radius, therefore only simulating rocky planets \*/

***break*;**

***case*** 2**:** console.log**(**"sun radius"**);**

***return*** simData.getRandom**(**Math.pow**(**10**,**10**),** 7 **\*** Math.pow**(**10**,**7**));**

//Star Radii, I researched normal max and min values for stars

**}**

**}*else*** **{**

***switch*(**simType**)** **{**

***default*:** console.log**(**"satellite radius"**);**

***return*** simData.getRandom**(**100**,**1**);**

/\*Satellite Radii (Assuming to be a sphere) I used 100 as the maximum for something on the

scale of the ISS, and 1m for the minimum small satellites \*/

***break*;**

***case*** 1**:** console.log**(**"moon radius"**);**

***return*** simData.getRandom**(**2 **\*** Math.pow**(**10**,**6**),**Math.pow**(**10**,**4**));**

/\*Moon Radii, using ganymede's radius as an approximation for the maximum radius,

using 10km as a minimum scale\*/

***break*;**

***case*** 2**:** ***return*** this.getRadius**(**3**,*true*);**

//Calls itself to get Planet Radii

**}**

**}**

**}**

getName**(**simType**,**largeBool**)** **{** //getter for the names of the bodies

//Returns names according to the body they are simulating, used for in asking questions

***if*** **(**largeBool **===** ***true*)** **{**

***switch*(**simType**)** **{**

***default*:** ***return*** "planet"**;** //Covers case 0 and 1

***break*;**

***case*** 2**:** ***return*** "star"**;**

**}**

**}** ***else*** **{**

***switch*(**simType**)** **{**

***default*:** ***return*** "satellite"**;**

***break*;**

***case*** 1**:** ***return*** "moon"**;**

***break*;**

***case*** 2**:** ***return*** this.getName**(**3**,*true*);** //calls itself to get planet name

**}**

**}**

**}**

getColour**(**simType**,**largeBool**)** **{** //getter for the colour of the bodies

//returns a colour according to the body they are simulating, used to represent simulation type graphically

***if*** **(**largeBool **===** ***true*)** **{**

***switch*(**simType**)** **{**

***default*:** ***return*** "blue"**;**

//covers case 0 and 1, planet colour

***break*;**

***case*** 2**:** ***return*** "yellow"**;**

//star colour

**}**

**}** ***else*** **{**

***switch*(**simType**)** **{**

***default*:** ***return*** "white"**;**

//satellite colour

***break*;**

***case*** 1**:** ***return*** "gray"**;**

//moon colour

***break*;**

***case*** 2**:** ***return*** this.getColour**(**3**,*true*);** //calls itself to get planet colour

//planet colour

**}**

**}**

**}**

**}**

***class*** classBodySmall ***extends*** classBodyLarge **{**

//Inherits the large body class

constructor**(**simType**,**largeBool**,**largeRadius**,**largeMass**)** **{**

//properties

***super*(**simType**,**largeBool**);** //Inheritance

this.x **=** 0 **;**

this.y **=** this.getOrbit**(**simType**,**largeRadius**);** //gets the radius height

this.vx **=** Math.sqrt**((**newtonG **\*** largeMass**)** **/** this.y**);** //performs a calculation and finds the orbital velocity at that height for a circular orbit

this.vy **=** 0**;**

**}**

//encapsulated functions

getOrbit**(**simType**,**largeRadius**)** **{** //getter for the orbit

***switch*** **(**simType**)** **{**

***default*:** console.log**(**"large Orbit"**)**

***return*** **(**simData.getRandom**(**60**,**5**)** **\*** largeRadius**)** **+** largeRadius**;**

/\*Gets a random radius between 60 and 5 radii away from the body's surface,

addition of 1 radius after to ensure that it cannot spawn below surface of the large body \*/

***break*;**

***case*** 0**:** console.log**(**"satellite Orbit"**)**

***return*** **(**simData.getRandom**(**6**,**1**)** **\*** largeRadius**)** **+** largeRadius**;**

/\*Gets a random radius between 6 and 1 radii away from the body's surface,

addition of 1 radius after to ensure that it cannot spawn below surface of the large body \*/

**}**

**}**

**}**

***class*** classBodyQuestion **{**

//properties

constructor**(**bodyLarge**,**bodySmall**)** **{**

//Takes values from the other objects, rounds them to 4 sig figs for easy use mathematics by the end user

this.y **=** bodySmall.y.toPrecision**(**4**);**

this.vx **=** bodySmall.vx.toPrecision**(**4**);**

this.radius **=** this.getRadiusQuestion**(**bodyLarge**,**bodySmall**);** //Gets both the large and small radii, returns to an array

this.mass **=** this.getMassQuestion**(**bodyLarge**,**bodySmall**);** //Gets both the large and small masses, returns to an array

this.angularVelocity **=** **(**bodySmall.vx **/** bodySmall.y**).**toPrecision**(**4**);** //uses physics equations to calculate remaining values

this.accel **=** **(**Math.pow**(**bodySmall.vx**,**2**)/**bodySmall.y**).**toPrecision**(**4**);**

this.force **=** **(**newtonG**\***bodySmall.mass**\***bodyLarge.mass**/**Math.pow**(**bodySmall.y**,**2**)).**toPrecision**(**4**);**

**}**

getRadiusQuestion**(**bodyLarge**,**bodySmall**)** **{**

***return*** **[**bodyLarge.radius.toPrecision**(**4**),**bodySmall.radius.toPrecision**(**4**)];** //returns large and small radii in an array

**}**

getMassQuestion**(**bodyLarge**,**bodySmall**)** **{**

***return*** **[**bodyLarge.mass.toPrecision**(**4**),**bodySmall.mass.toPrecision**(**4**)];** //returns large and small masses in an array

**}**

**}**

***class*** classSimulation **{**

constructor**()** **{**

//properties

this.questionType **=** simData.getQuestionType**(**simData.getQuestionsChosen**());** /\*questionType is set by getting the questions chosen from the user,

and selecting them using the getQuestionType function\*/

this.simType **=** Math.floor**(**simData.getRandom**(**3**,**0**));** /\*0 is planet/satellite, 1 is Planet/Moon, 2 is Star/Planet

If 3 is accidentally selected, it defaults to the satellite simulation\*/

this.bodyLarge **=** ***new*** classBodyLarge**(**this.simType**,*true*);** //sets bodyLarge as an object of class BodyLarge

this.bodySmall **=** ***new*** classBodySmall**(**this.simType**,*false*,**this.bodyLarge.radius**,**this.bodyLarge.mass**);** //sets bodySmall as an object of class BodySmall

this.bodyQuestion **=** ***new*** classBodyQuestion**(**this.bodyLarge**,**this.bodySmall**);** //sets bodyQuestion as an object of class BodyQuestion

**}**

**}**

***function*** initialise**(**isFirstTime**)** **{** //initialises the simulation on first load

***if*(**isFirstTime **==** ***true*){**

simData.setformHTML**();** //Sets formHTML as a copy of the initial simulation page

**}**

simData.setQuestionNumber**();** //sets the questionNumber

simData.setAlreadyClicked**(*false*);** //initialises (sets) alreadyClicked as false

simData.initialiseQuestionsAnswered**();** //initialises questionsAnswered

simData.initialiseQuestionsChosen**();** //initialises questionsChosen

document.getElementById**(**"userForm"**).**innerHTML **=** //Creates a form for the user to input their details into

"<form name=\"startForm\"><div style=\"color:#ffffff\">Please enter your College ID and what questions you would like to be tested on below to start the quiz.<br><form>" **+** //Title

"<div><label style=\"color:#ffffff\" for=\"userid\">College ID: </label> "**+** //College ID label

"<input type=\"text\" name =\"userid\" id=\"userid\"/><span id=\"useridError\" style=\"color:#ff0000\"></span>" **+** //text box for college ID and user ID error box

"</div><br>" **+**

"<div id=\"questionCheckbox\"><fieldset><legend>Select which questions you wish to be tested on.</legend>"**+** //a group of checkboxes to select question types

"<input type=\"checkbox\" name=\"questionTypes\" value=vFromM\_R />Finding orbital velocity from mass and orbit radius<br>" **+**

"<input type=\"checkbox\" name=\"questionTypes\" value=wFromV\_R />Finding angular velocity from velocity and orbit radius<br>" **+**

"<input type=\"checkbox\" name=\"questionTypes\" value=aFromV\_R />Finding acceleration from velocity and orbit radius<br>" **+**

"<input type=\"checkbox\" name=\"questionTypes\" value=gfFromM\_R\_V />Finding gravitational force from mass, orbit radius and velocity<br>" **+**

"<input type=\"checkbox\" name=\"questionTypes\" value=gfFromLM\_SM\_R />Finding gravitational force from masses, and orbit radius<br>" **+**

"<input type=\"checkbox\" name=\"questionTypes\" value=lmFromSM\_GF\_R />Finding large mass from the small mass, gravitational force and orbit radius<br>" **+**

"<input type=\"checkbox\" name=\"questionTypes\" value=smFromLM\_GF\_R />Finding the small mass from the large mass, gravitational force and orbit radius<br>" **+**

"<span id=\"questionCheckboxError\" style=\"color:#ff0000\"></span>"**+** //error boxes if no question is selected

"</fieldset><div id=\"div1\"></div><input type=\"button\" value =\"Start Quiz\" onclick=\"validateUserDetails()\" />"**;** //button that calls the start of the program

**}**

***function*** newSim**()** **{**

simulation **=** ***new*** classSimulation**();** //sets simulation as a new object from the class classSimulation

document.getElementById**(**"userForm"**).**innerHTML **=** simData.getformHTML**();** //gets formHTML which stores the html for the simulation/quiz

simData.setCurrentSimulation**(**simulation**);** /\*calls the setter for currentSimulation to make it available

for external function use\*/

simData.setSliderValues**(**simulation.simType**);** //calls the setter for the slider values, scaling the slider

***var*** metersPerPixel **=** simulation.bodySmall.y **/** **(**0.355 **\*** simData.getCanvases**()[**0**].**width**);** //scales the particle to the canvas by setting metersPerPixel to display

dispLargeObj**(**metersPerPixel**,**simulation**,*false*);** /\*calls the function for the large object, which displays on a seperate canvas,

this is so the large object does not have to be refreshed every time the function

moveProjectile is called.\*/

generateQuestion**(**simulation**);** /\*calls the generateQuestion function, which displays

the question decided by questionType\*/

simulationRunning **=** setInterval**(**moveProjectile**,**1000**/**300**,**simulation**,**metersPerPixel**,**simulation.questionType**,*true*);**/\*starts the simulation, sets it to call moveProjectile 300 times

every second until ended\*/

***var*** questionNumberDisplayed **=** simData.getQuestionNumber**()** **+**1**;** /\*variable questionNumberDisplayed is set one above the actual

questionNumber so it starts\*/

console.log**(**"QUESTION: " **+** questionNumberDisplayed **);** //at 1 and is used as the front display

questionCounter.innerHTML **=** "Question " **+** questionNumberDisplayed **+** " out of 10"**;**

debugData**(**simulation**);** //debug data function called for testing use

**}**

***function*** generateQuestion**(**simulation**)** **{**

***var*** questionText **=** ""**;**

***var*** questionValues **=** ""**;** //initialises all variables as empty

***var*** answerUnit **=** ""**;**

***switch*(**simulation.questionType**)** **{**

***default*:** //Given M, R, find v

questionText **=** "Given that the mass of the " **+** simulation.bodyLarge.name **+** " is " **+** simulation.bodyQuestion.mass**[**0**]** **+** //for each question, questionText is set to a

"kg, and the radius of the " **+** simulation.bodySmall.name **+** "'s orbit from the centre of the " **+** simulation.bodyLarge.name **+** //generalised question for each questionType,

" is " **+** simulation.bodyQuestion.y **+** "m, find the orbital velocity of the " **+** simulation.bodySmall.name **+** "."**;** //allowing for variables to be slotted in where

//applicable

questionValues **=** capitalizeFirstLetter**(**simulation.bodyLarge.name**)** **+** " mass: " **+** simulation.bodyQuestion.mass**[**0**]** **+** "kg<br>" **+** //The given data is written down directly here

capitalizeFirstLetter**(**simulation.bodySmall.name**)** **+** " orbit radius: " **+** simulation.bodyQuestion.y **+** "m"**;** //capitalizeFirstLetter function used here for design

answerUnit **=** "m/s"**;** //correct answer unit is specified

***break*;**

***case*** 1**:** //Given v, R, find w

questionText **=** "Given that the orbital velocity of the " **+** simulation.bodySmall.name **+** " is " **+** simulation.bodyQuestion.vx **+**

"m/s, and the radius of the " **+** simulation.bodySmall.name **+** "'s orbit is " **+** simulation.bodyQuestion.y **+**

"m, find the angular velocity of the " **+** simulation.bodySmall.name **+** "."**;**

questionValues **=** capitalizeFirstLetter**(**simulation.bodySmall.name**)** **+** " velocity: " **+** simulation.bodyQuestion.vx **+** "m/s<br>" **+**

capitalizeFirstLetter**(**simulation.bodySmall.name**)** **+** " orbit radius: " **+** simulation.bodyQuestion.y **+** "m"**;**

answerUnit **=** "m/s<sup>2</sup>"**;**

***break*;**

***case*** 2**:** //Given v, R, find a

questionText **=** "Given that the orbital velocity of the " **+** simulation.bodySmall.name **+** " is " **+** simulation.bodyQuestion.vx **+**

"m/s, and the radius of the " **+** simulation.bodySmall.name **+** "'s orbit is " **+** simulation.bodyQuestion.y **+**

"m, find the centripetal acceleration of the " **+** simulation.bodySmall.name **+** "."**;**

questionValues **=** capitalizeFirstLetter**(**simulation.bodyLarge.name**)** **+** " mass: " **+** simulation.bodyQuestion.mass**[**0**]** **+** "kg<br>" **+**

capitalizeFirstLetter**(**simulation.bodySmall.name**)** **+** " orbit radius: " **+** simulation.bodyQuestion.y **+** "m"**;**

answerUnit **=** "m/s<sup>2</sup>"**;**

***break*;**

***case*** 3**:** //Given m, v, R, find GF

questionText **=** "Given that the orbital velocity of the " **+** simulation.bodySmall.name **+** " is " **+** simulation.bodyQuestion.vx **+**

"m/s, the mass of the " **+** simulation.bodyLarge.name **+** " is " **+** simulation.bodyQuestion.mass**[**0**]** **+** "kg and the radius of the " **+**

simulation.bodySmall.name **+** "'s orbit is " **+** simulation.bodyQuestion.y **+**

"m, find the gravitational force acting between the " **+** simulation.bodySmall.name **+** " and the " **+** simulation.bodyLarge.name **+** "."**;**

questionValues **=** capitalizeFirstLetter**(**simulation.bodyLarge.name**)** **+** " mass: " **+** simulation.bodyQuestion.mass**[**0**]** **+** "kg<br>" **+**

capitalizeFirstLetter**(**simulation.bodySmall.name**)** **+** " orbit radius: " **+** simulation.bodyQuestion.y **+** "m<br>" **+**

capitalizeFirstLetter**(**simulation.bodySmall.name**)** **+** " velocity: " **+** simulation.bodyQuestion.vx **+** "m/s"**;**

answerUnit **=** "N"**;**

***break*;**

***case*** 4**:** //Given m1, m2, r, find GF

questionText **=** "Given that the mass of the " **+** simulation.bodySmall.name **+** " is " **+** simulation.bodyQuestion.mass**[**1**]** **+**

"kg, the mass of the " **+** simulation.bodyLarge.name **+** " is " **+** simulation.bodyQuestion.mass**[**0**]** **+** "kg and the radius of the " **+**

simulation.bodySmall.name **+** "'s orbit is " **+** simulation.bodyQuestion.y **+**

"m, find the gravitational force acting between the " **+** simulation.bodySmall.name **+** " and the " **+** simulation.bodyLarge.name **+** "."**;**

questionValues **=** capitalizeFirstLetter**(**simulation.bodyLarge.name**)** **+** " mass: " **+** simulation.bodyQuestion.mass**[**0**]** **+** "kg<br>" **+**

capitalizeFirstLetter**(**simulation.bodySmall.name**)** **+** " mass: " **+** simulation.bodyQuestion.mass**[**1**]** **+** "kg<br>" **+**

capitalizeFirstLetter**(**simulation.bodySmall.name**)** **+** " orbit radius: " **+** simulation.bodyQuestion.y **+** "m" **;**

answerUnit **=** "N"**;**

***break*;**

***case*** 5**:** //Given GF, m1, r, find m2

questionText **=** "Given that the gravitational force between the " **+** simulation.bodySmall.name **+** " and the " **+** simulation.bodyLarge.name **+**

" is " **+**simulation.bodyQuestion.force **+** "N, the mass of the " **+** simulation.bodyLarge.name **+** " is " **+** simulation.bodyQuestion.mass**[**0**]** **+**

"kg and the radius of the " **+** simulation.bodySmall.name **+** "'s orbit is " **+** simulation.bodyQuestion.y **+**

"m, find the mass of the " **+** simulation.bodySmall.name **+** "."**;**

questionValues **=** capitalizeFirstLetter**(**simulation.bodyLarge.name**)** **+** " mass: " **+** simulation.bodyQuestion.mass**[**0**]** **+** "kg<br>" **+**

"Gravitational Force: " **+** simulation.bodyQuestion.force **+** "N<br>" **+**

capitalizeFirstLetter**(**simulation.bodySmall.name**)** **+** " orbit radius: " **+** simulation.bodyQuestion.y **+** "m" **;**

answerUnit **=** "kg"**;**

***break*;**

***case*** 6**:** //Given GF, m2, r, find m1

questionText **=** "Given that the gravitational force between the " **+** simulation.bodySmall.name **+** " and the " **+** simulation.bodyLarge.name **+**

" is " **+** simulation.bodyQuestion.force **+** "N, the mass of the " **+** simulation.bodySmall.name **+** " is " **+** simulation.bodyQuestion.mass**[**1**]** **+**

"kg and the radius of the " **+** simulation.bodySmall.name **+** "'s orbit is " **+**

simulation.bodyQuestion.y **+** "m, find the mass of the " **+** simulation.bodyLarge.name **+** "."**;**

questionValues **=** capitalizeFirstLetter**(**simulation.bodySmall.name**)** **+** " mass: " **+** simulation.bodyQuestion.mass**[**1**]** **+** "kg<br>" **+**

"Gravitational Force: " **+** simulation.bodyQuestion.force **+** "N<br>" **+**

capitalizeFirstLetter**(**simulation.bodySmall.name**)** **+** " orbit radius: " **+** simulation.bodyQuestion.y **+** "m" **;**

answerUnit **=** "kg"**;**

**}**

document.getElementById**(**"txtQuestion"**).**innerHTML **=** questionText**;** //sets each respective HTML element as

document.getElementById**(**"txtValues"**).**innerHTML **=** questionValues**;** //the values specified above

document.getElementById**(**"answerUnit"**).**innerHTML **=** answerUnit**;**

**}**

***function*** validateUserDetails**()** **{** //validates the user data, ensures that invalid data cannot be entered

simData.setQuestionsChosen**();** //sets questionsChosen as the selected questions from the initialise function

simData.setUserID**();** //sets userID as the entered value from the initialise function

questionsChosen **=** simData.getQuestionsChosen**();** //gets the questions chosen from checkbox

***var*** userDataValid **=** ***false*;** //initialises the validity flag, to be changed from false to true

***var*** userID **=** simData.getUserID**();** //gets the userID

***for*** **(**x**=**0**;**x**<**questionsChosen.length**;**x**++){**

***if*(**questionsChosen**[**x**]** **==** ***true*){** //iterates through the checkboxes

document.getElementById**(**"questionCheckboxError"**).**innerHTML **=**"" //else, error box is emptied

userDataValid **=** ***true*;** //userDataValid is only true if at least 1 checkbox is ticked

***break*;**

**}** ***else*** ***if*(**x **==** **(**questionsChosen.length **-**1**))** **{** //if none are ticked, error message displayed and invalid flag is raised

console.log**(**"failed questionType"**);**

document.getElementById**(**"questionCheckboxError"**).**innerHTML **=** "Please select at least one question type."**;**

**}**

**}**

***if*(**isNaN**(**userID**)** **||** userID **<** 100000 **||** userID **>** 999999**){** //checks if userID is a number, and is between 100000 and 999999

console.log**(**"failed college id"**);**

document.getElementById**(**"useridError"**).**innerHTML **=** "Please enter a valid College ID."**;** //if not, error message displayed, flag raised

userDataValid **=** ***false*;**

**}** ***else*** **{**document.getElementById**(**"useridError"**).**innerHTML **=**""**}** //else, error box emptied

***if*(**userDataValid **==** ***true*){** //if flag has not been raised, run newSim

newSim**();**

**}**

**}**

***function*** capitalizeFirstLetter**(**string**){**

***return*** string.charAt**(**0**).**toUpperCase**()** **+** string.slice**(**1**);** //function used to capitalize first letter of a string

**}**

***function*** debugData**(**simulation**)** **{** //debug bar to allow for easy testing, would not be present in end user copy

console.log**(**"bodyLarge.mass: " **+** Math.floor**(**simulation.bodyLarge.mass**));**

console.log**(**"bodyLarge.radius: " **+** Math.floor**(**simulation.bodyLarge.radius**));**

console.log**(**"bodyLarge.name: " **+** simulation.bodyLarge.name**);**

console.log**(**"bodySmall.y: " **+** Math.floor**(**simulation.bodySmall.y**));**

console.log**(**"bodySmall.vx: " **+** Math.floor**(**simulation.bodySmall.vx**));**

console.log**(**"bodySmall.radius: " **+** Math.floor**(**simulation.bodySmall.radius**));**

console.log**(**"bodySmall.mass: " **+** Math.floor**(**simulation.bodySmall.mass**));**

console.log**(**"bodySmall.name: " **+** simulation.bodySmall.name**);**

console.log**(**"bodyQuestion.vx: " **+** simulation.bodyQuestion.vx**);**

console.log**(**"bodyQuestion.angularVelocity: " **+** simulation.bodyQuestion.angularVelocity**);**

console.log**(**"bodyQuestion.accel: " **+** simulation.bodyQuestion.accel**);**

console.log**(**"bodyQuestion.force: " **+** simulation.bodyQuestion.force**);**

console.log**(**"simType: " **+** simulation.simType**);**

console.log**(**"bodyQuestion.questionType: " **+** simulation.questionType**);**

**}**

***function*** dispLargeObj**(**metersPerPixel**,**simulation**,**isQuiz**)** **{** //function to display the large body

console.log**(**"Displaying large obj"**);**

canvas**=**simData.getCanvases**()[**2**];** //gets the canvases, assigns [2] to a variable which stores the large body canvas

context**=**simData.getContexts**()[**2**];** //gets the contexts, assigns [2] to a variable which stores the large body context

context.beginPath**();**

context.arc**(**canvas.width**/**2**,**canvas.height**/**2**,**simulation.bodyLarge.radius**/**metersPerPixel**,**0**,**2**\***Math.PI**);**

context.fillStyle **=** simulation.bodyLarge.colour**;** //Displays the large object using canvas inbuilt functions

context.fill**();**

context.font **=** "12px Times New Roman"**;**

context.fillStyle **=** "white"**;**

context.textAlign **=** "left"**;**

context.fillText**(**simulation.bodyQuestion.radius**[**0**]** **+** "m"**,** canvas.width**/**2 **+** //Displays the radius on the edge of the object

simulation.bodyLarge.radius**/**metersPerPixel**,**canvas.height**/**2**);**

***if*** **(**simulation.questionType **!=** 6**)** **{**

context.fillText**(**simulation.bodyQuestion.mass**[**0**]** **+** "kg"**,** canvas.width**/**2 **+** //Only displays the mass if the question isn't to find the mass

simulation.bodyLarge.radius**/**metersPerPixel**,**canvas.height**/**2 **-** 15**);**

**}**

**}**

***function*** moveProjectile**(**simulation**,**metersPerPixel**,**questionType**,**isQuiz**)** **{**

***var*** dt **=** difTime.value**;** // time step in seconds

simulation.bodySmall.r **=** Math.sqrt**(**simulation.bodySmall.x**\***simulation.bodySmall.x **+** //gets the magnitude of the radius, by performing pythagoras on the x and y co-ordinate

simulation.bodySmall.y**\***simulation.bodySmall.y**);**

***var*** accel **=** newtonG **\*** simulation.bodyLarge.mass **/** **(**simulation.bodySmall.r **\*** simulation.bodySmall.r**);**//acceleration found from GM/R^2 equation

***var*** ax **=** **-**accel **\*** simulation.bodySmall.x **/** simulation.bodySmall.r**;** //both ax and ay contain the compartmentalised acceleration, circular motion equations used

***var*** ay **=** **-**accel **\*** simulation.bodySmall.y **/** simulation.bodySmall.r**;**

simulation.bodySmall.vx **+=** ax **\*** dt**;** //v set to v+a\*dt, which is v=u+at SUVAT

simulation.bodySmall.vy **+=** ay **\*** dt**;**

simulation.bodySmall.x **+=** simulation.bodySmall.vx **\*** dt**;** //s set to v\*dt, which is s=ut SUVAT

simulation.bodySmall.y **+=** simulation.bodySmall.vy **\*** dt**;**

drawProjectile**(**simulation**,**metersPerPixel**,**questionType**,**isQuiz**);** //calls drawProjectile to display new values

**}**

***function*** drawProjectile**(**simulation**,**metersPerPixel**,**questionType**,**isQuiz**)** **{**

canvas **=** simData.getCanvases**()[**0**]** //sets canvas to [0] of the canvas array, contains the particle canvas

context **=** simData.getContexts**()** //gets all contexts

***var*** pixelX **=** canvas.width**/**2 **+** simulation.bodySmall.x**/**metersPerPixel**;**

***var*** pixelY **=** canvas.height**/**2 **-** simulation.bodySmall.y**/**metersPerPixel**;** //scaling using metersPerPixel to ensure all of simulation displays properly

context**[**0**].**clearRect**(**0**,** 0**,** canvas.width**,** canvas.height**);** //clears canvas

context**[**1**].**beginPath**();** //draws the particle trail in red

context**[**1**].**arc**(**pixelX**,** pixelY**,** 1**,** 0**,** 2**\***Math.PI**);**

context**[**1**].**fillStyle **=** "red"**;**

context**[**1**].**fill**();**

context**[**0**].**beginPath**();**

context**[**0**].**arc**(**pixelX**,** pixelY**,** 5**,** 0**,** 2**\***Math.PI**);** //draws the particle

context**[**0**].**fillStyle **=** simulation.bodySmall.colour**;**

context**[**0**].**fill**();**

context**[**0**].**font **=** "12px Times New Roman"**;**

context**[**0**].**fillStyle **=** "white"**;**

context**[**0**].**textAlign **=** "left"**;**

//if the question is not asking for the mass of the small object, display the mass

***if*(**questionType **!=** 5**)** **{**

context**[**0**].**fillText**(**simulation.bodyQuestion.mass**[**1**]** **+** "kg"**,** pixelX **+** 5**,** pixelY **+** 5**);**

**}**

context**[**0**].**fillText**(**simulation.bodyQuestion.radius**[**1**]** **+** "m"**,** pixelX **+** 5**,**pixelY **-** 10**);** //displays the radius right next to the object as it's moving

**}**

***function*** submitAnswer**(**answerGiven**){** //called when answer is submitted using the submit button

***var*** questionNumber **=** simData.getQuestionNumber**();** //gets the questionNumber

***if*((**answerGiven **==** ""**)** **||** isNaN**(**answerGiven**)){** //if answerGiven is empty or not a number, display error and wait until valid answer is given

document.getElementById**(**"txtAnswerError"**).**innerHTML **=** "Please enter a valid answer."**;**

**}*else*** ***if*(**simData.getAlreadyClicked**()** **==** ***false*){** //if the button has already been clicked, disables the button being pushed again

document.getElementById**(**"txtAnswerError"**).**innerHTML **=** ""**;**

simData.setAlreadyClicked**(*true*);** //sets alreadyClicked to true

***var*** simulation **=** simData.getCurrentSimulation**();** //gets the simulation

***var*** dataSent **=** ""**;**

***switch*(**simulation.questionType**)** **{** //switches what the correct answer is depending on the question type

***default*:**

dataSent **=** simulation.bodyQuestion.vx**;**

***break*;**

***case*** 1**:**

dataSent **=** simulation.bodyQuestion.angularVelocity**;**

***break*;**

***case*** 2**:**

dataSent **=** simulation.bodyQuestion.accel**;**

***break*;**

***case*** 3**:**

***case*** 4**:**

dataSent **=** simulation.bodyQuestion.force**;**

***break*;**

***case*** 5**:**

dataSent **=** simulation.bodyQuestion.mass**[**1**];**

***break*;**

***case*** 6**:**

dataSent **=** simulation.bodyQuestion.mass**[**0**];**

***break*;**

**}**

simData.setQuestionsAnsweredCorrect**(**questionNumber**,**isCorrect**(**answerGiven**,**dataSent**),**simulation.questionType**);**//sets the questionsAnswered and questionsCorrect variables to be sent, calls

//isCorrect and compares answerGivenv with the dataSent from the actual simulation

console.log**(**"simulationRunning: " **+** simulationRunning**);**

setTimeout**(**endSim**,**2000**);** //calls endSim after 2000ms, to allow for correct/false messages to be displayed

//before the next question

**}**

**}**

***function*** isCorrect**(**answerGiven**,**answerTrue**)** **{** //called to determine if the value is correct or not

***if*** **(**Math.abs**((**answerGiven**-**answerTrue**)/**answerTrue**)** **<=** 0.05**)** **{** //evaluates the answer, if it is more than 5% off the true value, it is incorrect

document.getElementById**(**"txtQuestion"**).**innerHTML **=** **(**"Correct! <br>" **+** //correct response

document.getElementById**(**"txtQuestion"**).**innerHTML**);**

***return*** ***true*;**

**}*else*{**

document.getElementById**(**"txtQuestion"**).**innerHTML **=** **(**"Wrong, keep going! <br>" **+**

document.getElementById**(**"txtQuestion"**).**innerHTML**);** //incorrect response

***return*** ***false*;**

**}**

**}**

***function*** endSim**()** **{** //ends the simulation

simData.setAlreadyClicked**(*false*);** //sets alreadyClicked to false

simData.incrementQuestionNumber**();** //increments QuestionNumber

console.clear**();** //clears the console (debug purposes)

clearInterval**(**simulationRunning**);** //ends the simulation

simData.clearAll**();** //clears all the canvases

***if*(**simData.getQuestionNumber**()** **===** 10**)** **{** //if questionNumber = 10, end simulation fully

document.getElementById**(**"userForm"**).**innerHTML **=** "<input type=\"button\" value =\"Go Again?\" onclick=\"initialise(false)\" />" //easy reset button

postUserData**();** //posts the userData to the server

**}** ***else*** **{**

//else, start the next simulation question

newSim**();**

**}**

**}**

***function*** postUserData**(){**

***var*** userID **=** simData.getUserID**();** //gets UserID

***var*** quizScore **=** 0**;** //initialises score variable

***var*** questionsAnswered **=** simData.getQuestionsAnswered**();** //gets the QuestionsAnswered and QuestionsCorrect variables

***var*** questionsCorrect **=** simData.getQuestionsCorrect**();**

***for*** **(**x **=** 0**;** x **<** questionsCorrect.length**;** x**++){** //for every question correct, add to the score variable

***if*(**questionsCorrect**[**x**]** **==** 1**)** **{**

quizScore **+=**1**;**

**}**

**}**

createCookie**(**"CollegeID"**,**userID**,** 1**);** /\*create cookies for all postable data, userID, quizScore, and

questionsAnswered and questionsCorrect as an array \*/

createCookie**(**"QuizScore"**,**quizScore**,** 1**);**

createCookie**(**"QuestionsAnswered"**,**questionsAnswered.join**(**','**),**1**);**

createCookie**(**"QuestionsCorrect"**,**questionsCorrect.join**(**','**),**1**);**

$**.**post**(**"/test/phpServerContent.php"**,**"function(result){})"**);** //posts to phpServerContent using AJAX

**};**

***function*** createCookie**(**name**,** value**,** days**)** **{** //create cookie function

***var*** expires**;** //initialise expiry

***if*** **(**days**)** **{**

***var*** date **=** ***new*** Date**();** //sets it so that the cookie takes long to expire so it can be read fully

date.setTime**(**date.getTime**()** **+** **(**days **\*** 24 **\*** 60 **\*** 60 **\*** 1000**));**

expires **=** "; expires=" **+** date.toGMTString**();**

**}** ***else*** **{**

expires **=** ""**;**

**}**

document.cookie **=** escape**(**name**)** **+** "=" **+** escape**(**value**)** **+** expires **+** "; path=/"**;** //adds the cookie to the document cookies

**}**

</script>

</body>

</html>

#### path\_to\_db\_connect.php

<?php

$dns = "mysql:host=localhost;dbname=userdatagravsim";

$user ="root";

$password = "";

$connect = **new** PDO($dns, $user, $password);

?>

#### phpServerContent.php

<?php

**require\_once** 'path\_to\_db\_connect.php';

$CollegeID = $\_COOKIE["CollegeID"];

$QuizScore = $\_COOKIE["QuizScore"];

$DateTestTaken = **date**("Y-m-d");

$QuestionsAnswered = **explode** (',',$\_COOKIE["QuestionsAnswered"]);

$QuestionsCorrect = **explode** (',',$\_COOKIE["QuestionsCorrect"]);

$requestTests = $connect->prepare('INSERT INTO tests(CollegeID,QuizScore,DateTestTaken) values(?,?,?);');

$requestTests->execute(**array**($CollegeID,$QuizScore,$DateTestTaken));

$requestTestID = $connect->prepare("SELECT `TestID` FROM `tests` WHERE `CollegeID` = ?;");

$requestTestID->execute(**array**($CollegeID));

$TestID = **implode**($requestTestID->fetch(PDO::FETCH\_ASSOC));

**echo** **var\_dump**($TestID);

//The ? will be replace when query will be executed

$requestTestQuestions = $connect->prepare('INSERT INTO `testquestions`(`QuestionNumber`, `TestID`, `TypeID`, `Correct`) VALUES (?,?,?,?);');

**for**($x = 0; $x<**sizeof**($QuestionsAnswered);$x++) {

**echo** $x + "<br>";

**echo** $TestID;

**echo** $QuestionsAnswered[$x]+ "<br>";

**echo** $QuestionsCorrect[$x]+ "<br>";

$requestTestQuestions->execute(**array**($x,$TestID,$QuestionsAnswered[$x] + 1,$QuestionsCorrect[$x]));

}

**if**(!$requestTests **or** !$requestTestQuestions){

**echo** 'Error with INSERT';

}

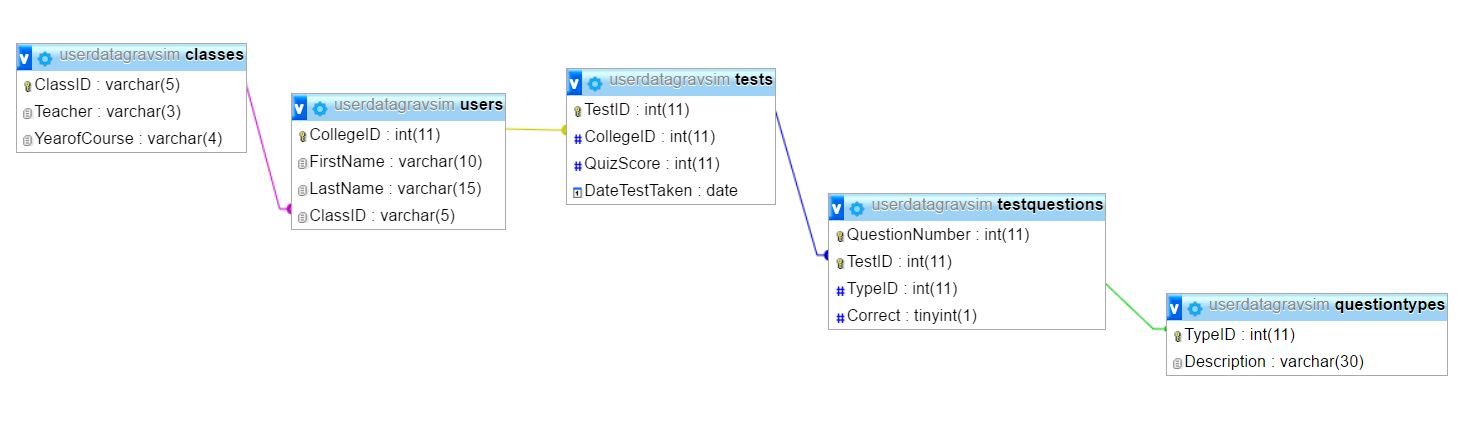
**else** {

**echo** 'INSERT success';

}

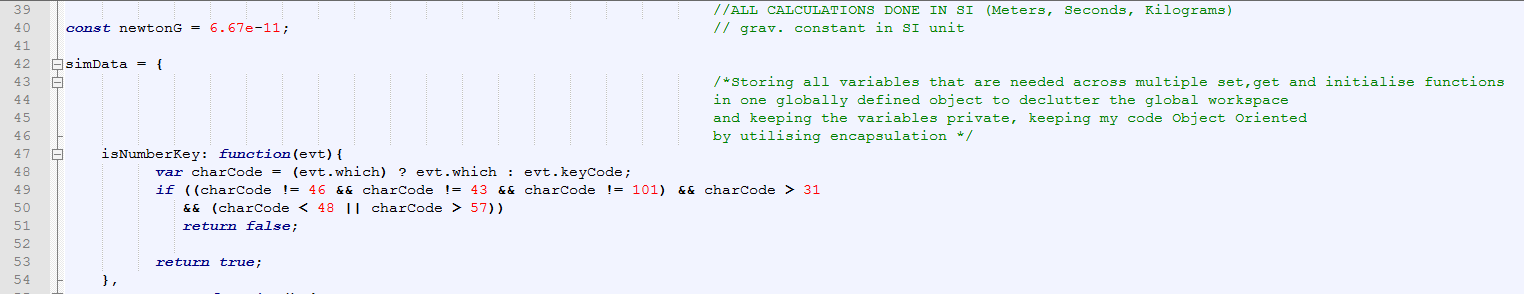
?>

### Table Relationships



### Detailed Code Analysis

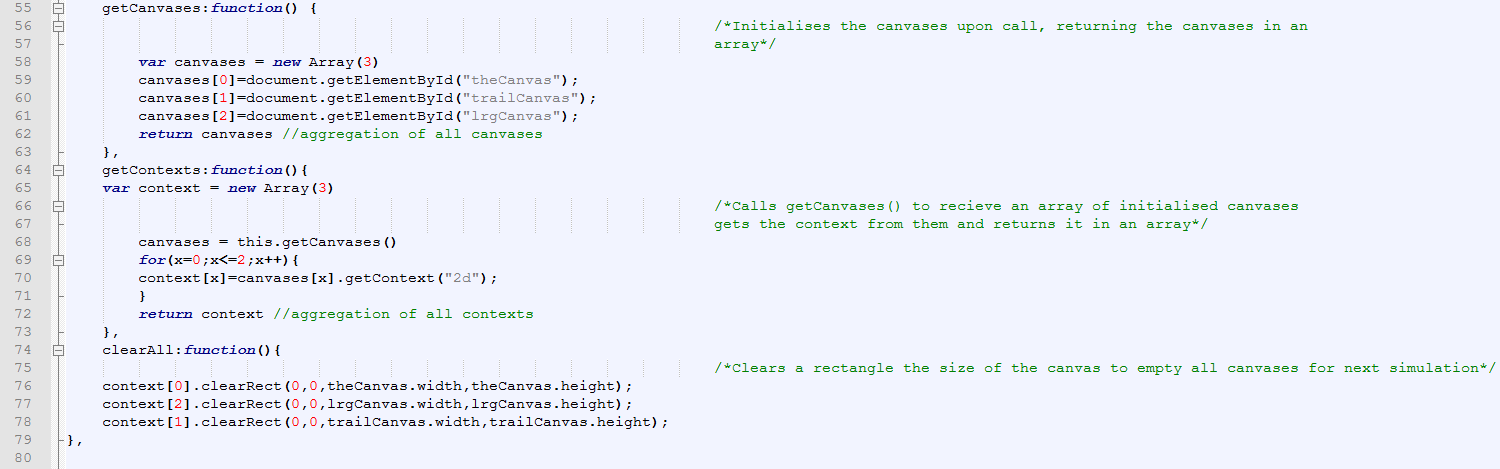
##### SimData start and isNumberKey



This is the start of my code. I open it with the creation of simData, which stores all of my setter/getter functions. This preserves the Object-Oriented nature of my code, as each piece of data is stored within a function, and within a large object. This ensures no variable is accidentally accessed, as I have hidden the variables inside a function, and in an object, which reduces the chance of data corruption.

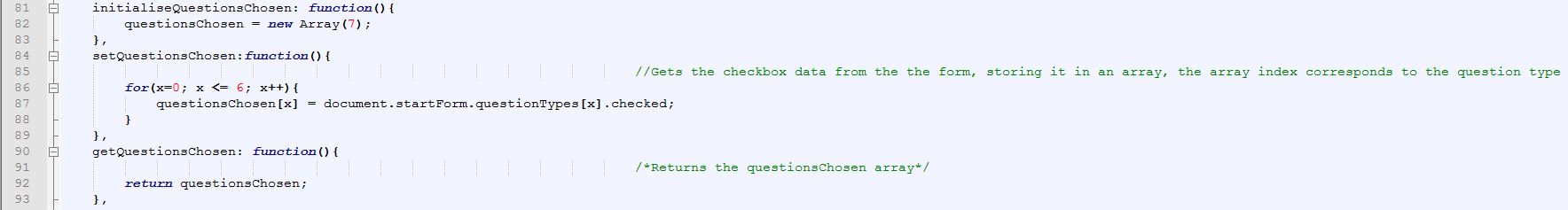
The first function in this large simData object is isNumberKey, which takes the keystroke input from pressing the submit button, and will only display the character associated with the keystroke if it is either numeric, ‘e’, or ‘+’. This allows exponent form to be entered, but does not allow someone to type in invalid data, such as letters or symbols.

##### getCanvas and getContexts function



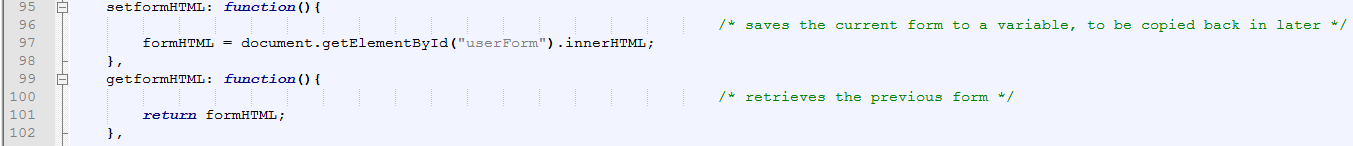
Both functions are getters, meaning they retrieve the data from somewhere. They retrieve the data of the 3 canvases that they are displayed on. getContext calls getCanvases, as getContexts retrieves the context of each canvas, which is a required attribute of the canvas that is edited to paint shapes onto the canvas. This is called before displaying an object on any of the canvases.

##### Initialise,set,getQuestionsChosen



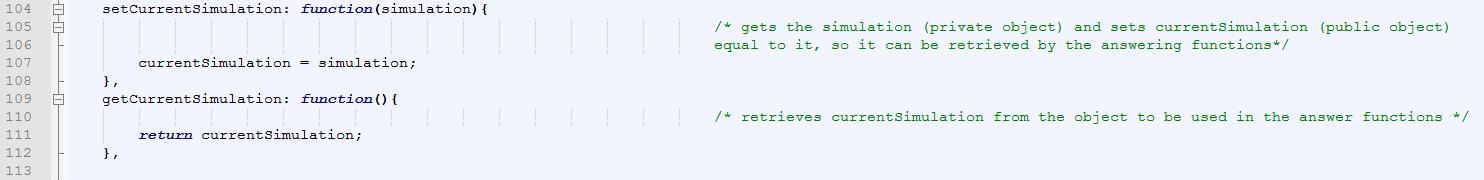
These functions deal with the checkboxes in the userform at the start of the program. At initialisation, an array of size 7 is created. When set, it is given all of the .checked values of the checkboxes, using a for loop to iterate through them, creating an array full of Boolean values. This is the returned upon calling the get function.

##### Set,get FormHTML



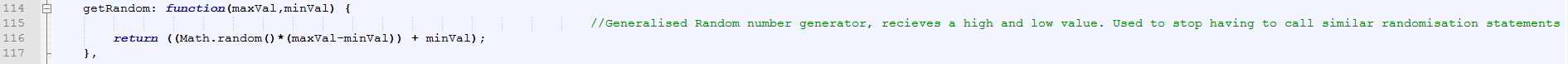
This stores the html for the quiz. It sets it at the start of the program before it is changed for the userform, and then when the userform is complete, the get function of this is called to return the html for the quiz. This is then displayed, and overwrites the html for the userform.

##### Set,get currentSimulation



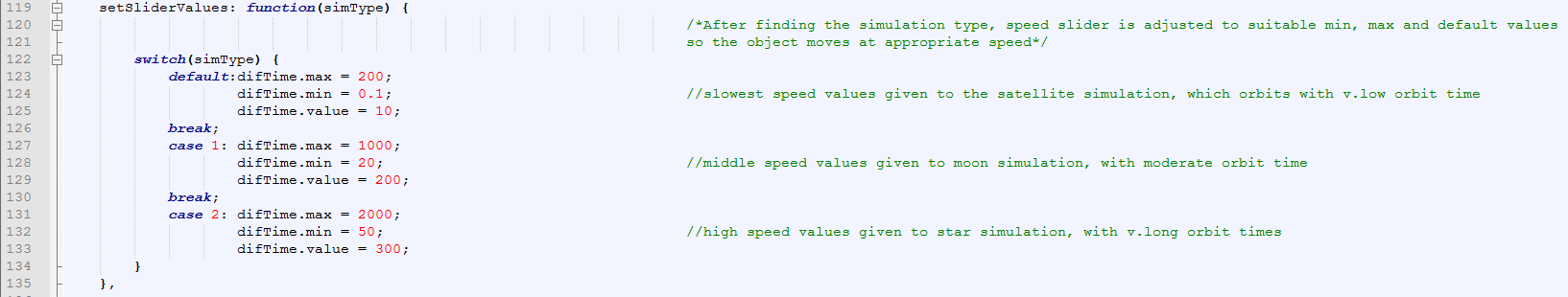
These functions deal with the currentSimulation variable. This takes in the contents of the simulation object, sets it equal to currentSimulation, and then the get function returns the data. This is used so the simulation data can be accessed from other functions in the program that do not directly follow the newSim(), primarily the submitAnswer() function, whilst preserving the Object-Oriented nature of the code and not placing any variables within the global workspace.

##### getRandom function



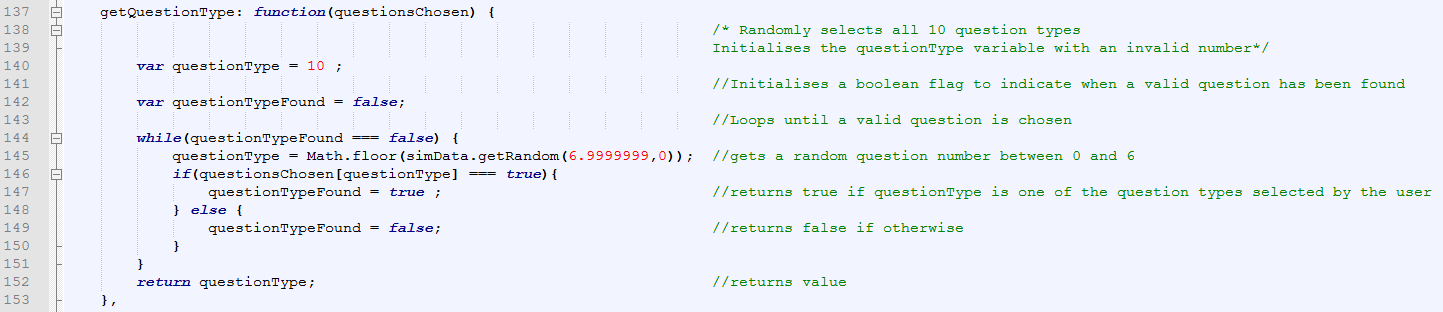
The getRandom function is a generalised random number generator in my function, taking in inputs of the maximum and minimum number. This then uses the Math.random() inbuilt function to generate a random number between these two values. This creates a general function for my randon number generator, and is therefore easily called multiple times instead of being repeated multiple times.

##### getSliderValues function



This get function sets the maximum and minimum, as well as the default position for the speed slider in my simulation. This is because my 3 different simulation types all have different appropriate speed of orbit ranges, and therefore require different maxima and minima for each simulation type. Therefore, when called, and simType passed in, it returns appropriate values for that simulation type.

##### getQuestionType function



The getQuestionType function is used to randomly select the questions, from the choices of the questions selected. It does this by having questionsChosen passed into it, and then questionType is set to a random number between 0 and 6 using the .getRandom() function. If the questiontype is found in the array to be true, it breaks out of the while loop and the value is used. If not, it does not break the while loop and questionType is given another random question until it is one of the questions selected. This questionType is then returned, now guaranteed to be one the user selected at the start of the program.

##### Set,get UserID



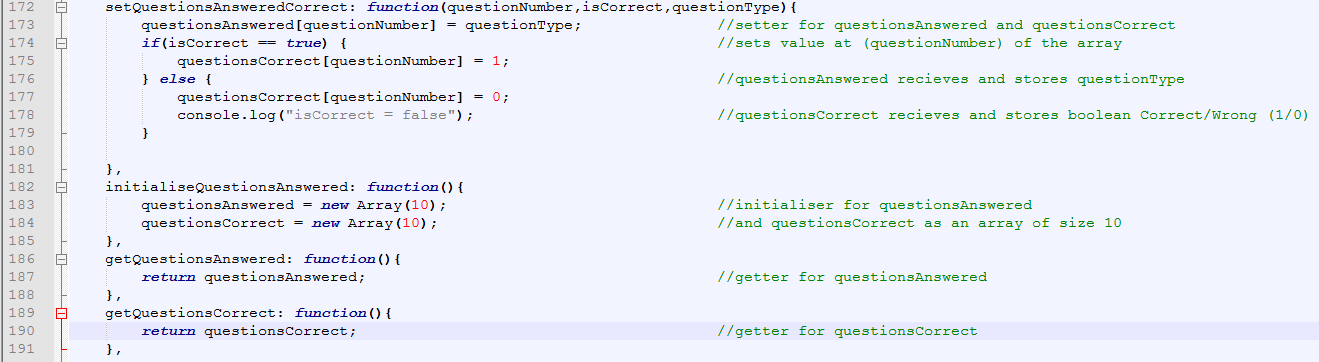
The setter for userID is called at the end of the userForm, when the submit button is pressed. It sets userID equal to the contents of the College ID textbox. userID is then easily returned when getuserID is called. The setter can be called multiple times to overwrite the value stored in userID, as would need to occur if the userID failed validation.

##### Set,get,and increment QuestionNumber



The setter for questionNumber gives it a default value of 0. This initialises questionNumber at 0, as in no other case is it needed to be distinctly set to a value, only added to. IncrementQuestionNumber however is called at the end of every question. This adds one to the value of questionNumber and nothing more. The getter function returns questionNumber.

##### Set questionsAnsweredCorrect, initialiseQuestionsAnswered, get questionsAnswered and questionsCorrect

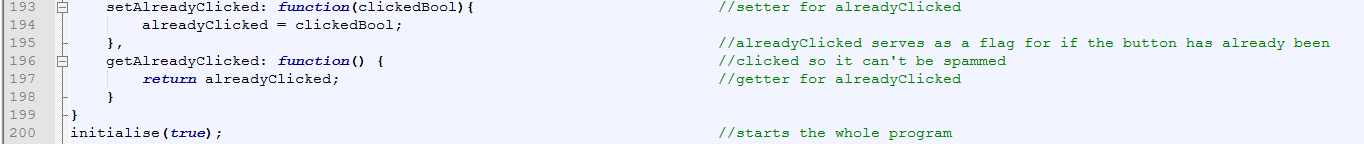


The initialiser function initialises both arrays, at size 10. This means it has exactly enough storage to contain the questionType and Boolean answer for the entire quiz, as the quiz has exactly 10 questions. This creates some amount of storage efficiency, as there are no unused spaces in this array, as it is given a definite size, as I know it will not contain any more than 10 values at any time.

The setter function sets one element of each of the arrays questionsAnswered and questionsCorrect to their respective values when this function runs. The number of the element is equal to questionNumber, and is called once for each question, leading to 10 entries into both arrays. questionsAnswered stores the questionType of the question entered, whilst questionsCorrect stores a Boolean indicating whether the question was marked as correct or not.

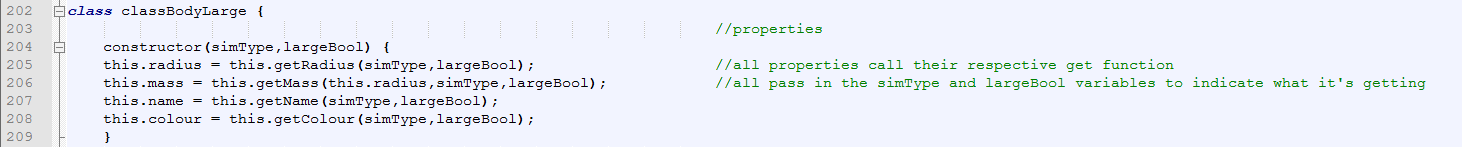
The getter functions are simple and just return each respective array. This preserves the OOP of the code.

##### Set, get AlreadyClicked



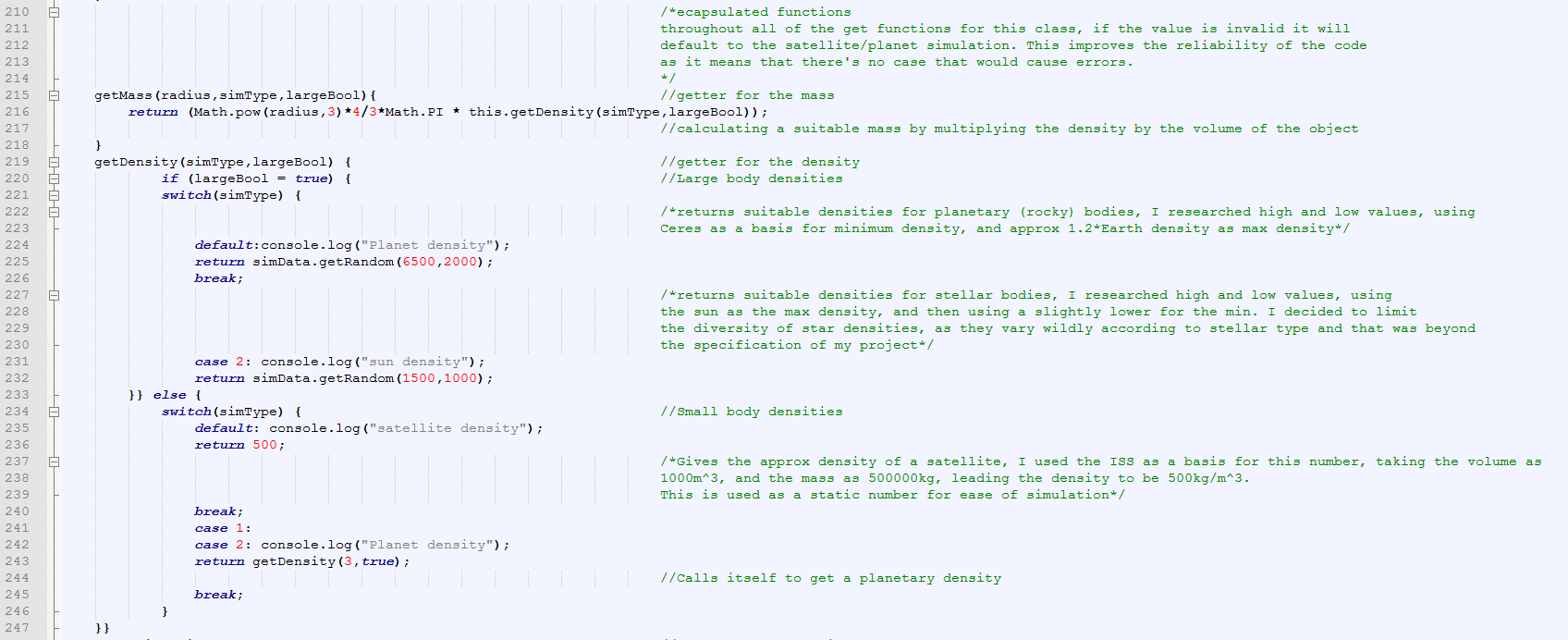
This is a normal setter function, which gets a variable passed into it when it’s called. alreadyClicked is then set to this value. The purpose of this variable is to serve as a flag for if the submit button has already been clicked, so it can’t be tapped repeatedly and questions skipped without an answer. The get function returns this variable.

##### ClassBodyLarge properties



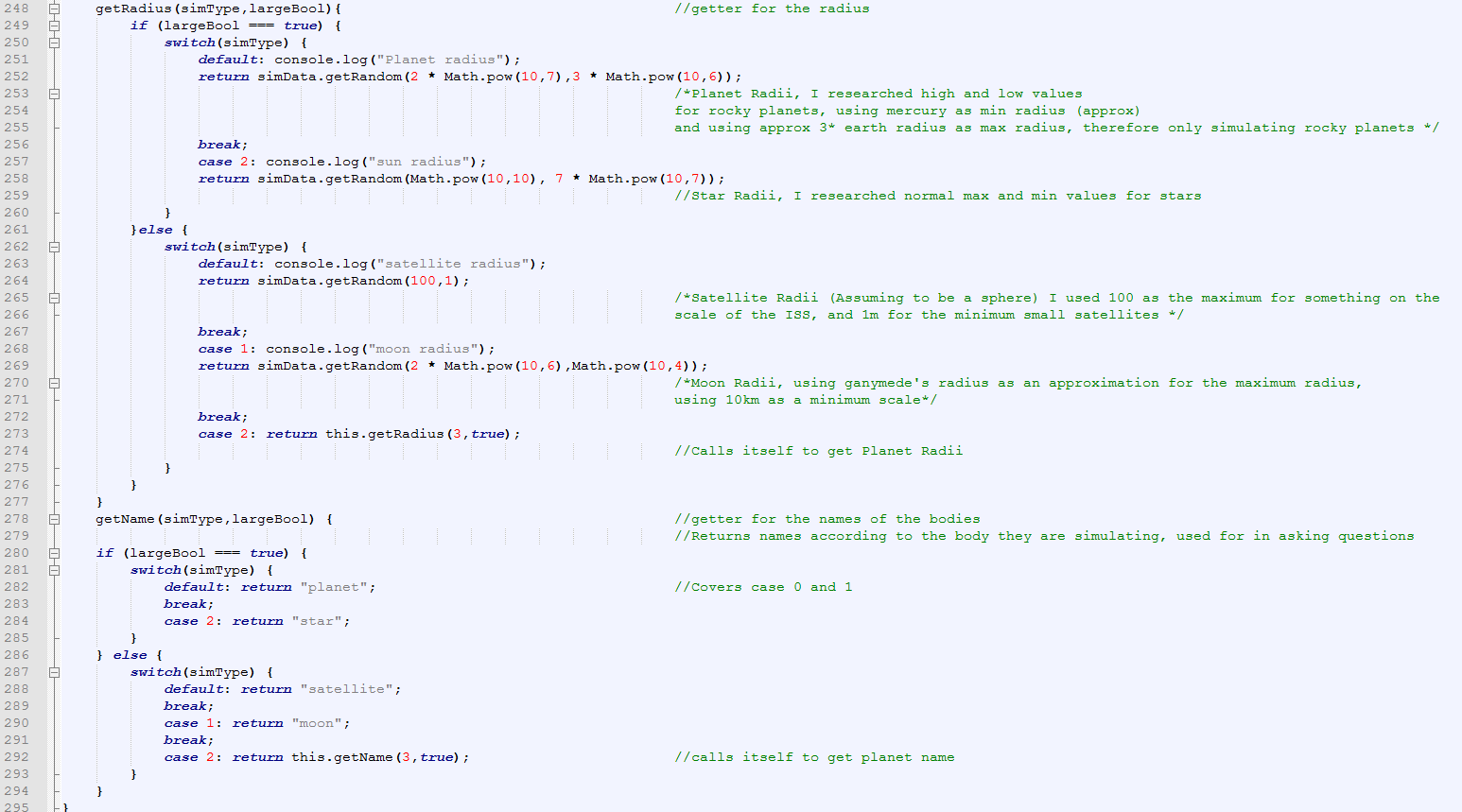
This opens the class ClassBodyLarge, which is the basis for the object bodyLarge, and also the parent of ClassBodySmall. Here I have displayed the properties of the class, all of which call their respective get functions to retrieve them from the encapsulated functions within ClassBodyLarge. The class can only be created when the variables simType and largeBool are passed into it, as this dictates the ranges in which the radius, mass and density can be between, as well as what the name and colour are equal to.

##### ClassBodyLarge functions



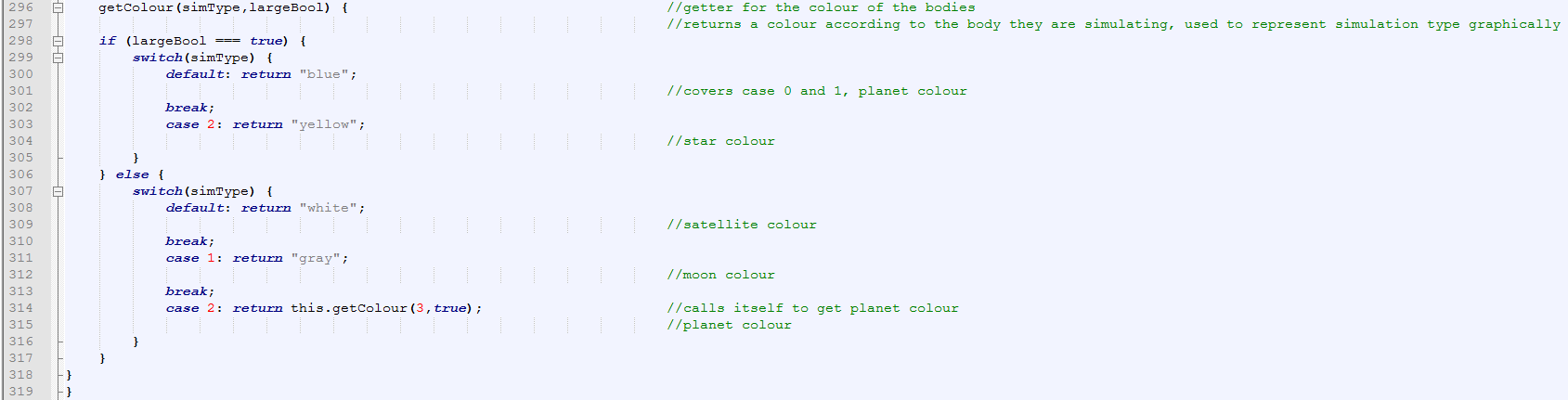
Getmass is a simple function, calling getDensity, and getting the property radius of the classBodyLarge. This performs a simple calculation, calculating the volume from the radius and then multiplying it by the density. This calculates the mass, and returns the value. getDensity, on the other hand, takes the variables passed in – simType, and largeBool, and uses them in a combination of two selection statements – an if statement to determine whether it’s a large or small body, and then a switch statement depending on what simType is selected. This allows it to narrow down what type of object is being created, and then returns the appropriate density. Each maximum and minimum value has been researched to ensure it is accurate.

##### getRadius and getName



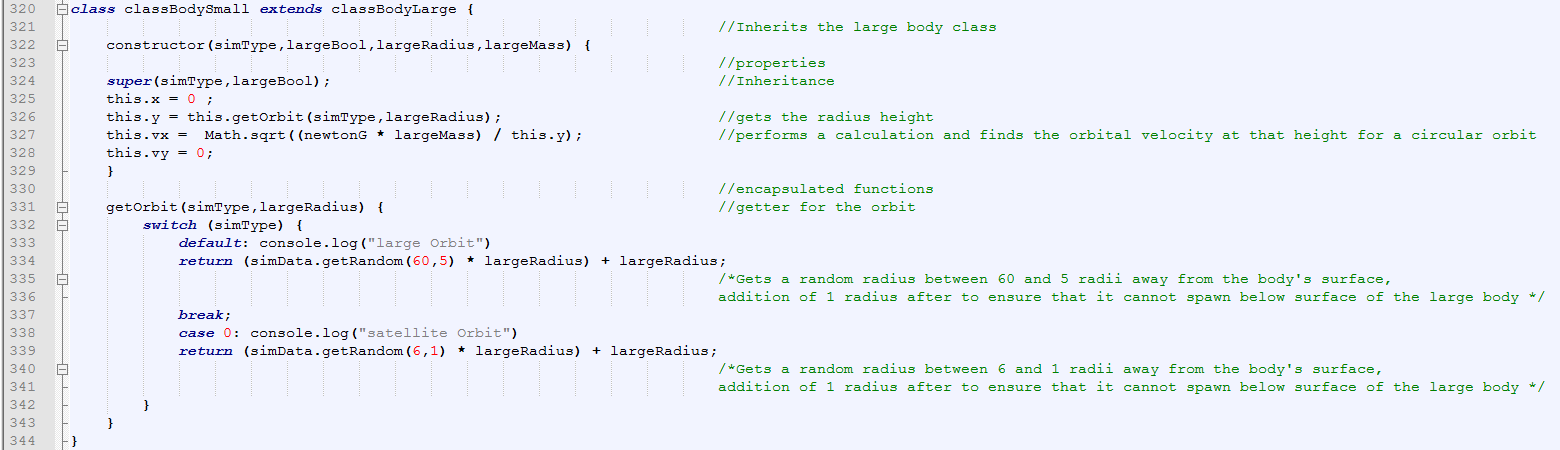
getRadius performs a similar sort of selection statement to that of getDensity. It performs an if selection based on the value of simType, and then a switch statement based on the value of simType. This then returns the radius, which has been randomised inbetween the maximum and minimum values decided by the selection. Again, all values have been researched, and this adds an extra level of realism for my simulation. For example, the planet radii has been given a maximum value of 3 Earth radii, and a minimum value equivalent to mercury’s radius.

##### getColour



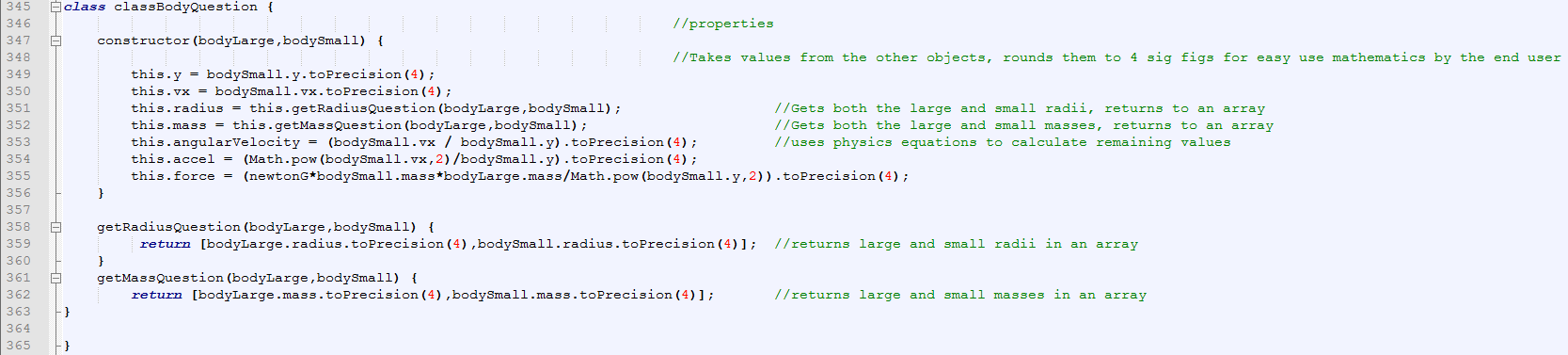
Again, getcolour is based upon 2 selection statements to work out the type of object being created. Once each selection statement has been passed through, it returns the display colour for the object.

##### classBodySmall



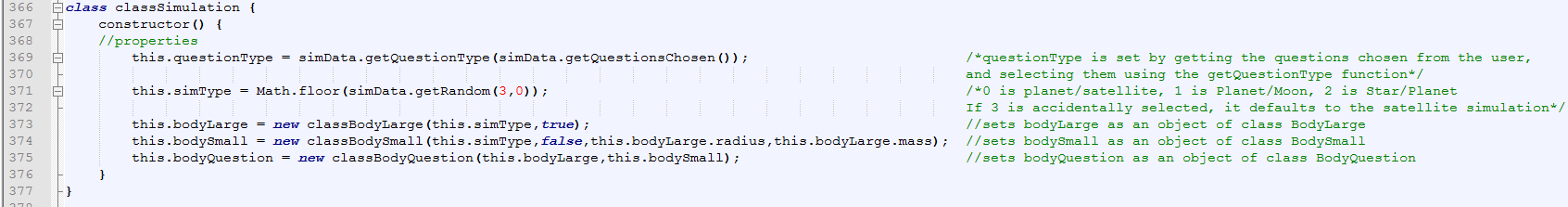
ClassBodySmall is the class that the object BodySmall is derived from. It inherits the class classBodyLarge using the extends function, and then inherits its properties using the super() function. This means that the BodySmall object that is created from this class also has all the properties that the BodyLarge has, which means I can give colour, name, mass, and radius to bodySmall without having to write the same function twice. This means inheritance is very good for code efficiency. BodySmall is also given the properties x, y, vx, and vy. These allow for me to give the object movement using the moveProjectile function. It also adds on one encapsulated function – getOrbit, which randomly generates the height of the object, scaled by the radius of the large object.

##### classBodyQuestion



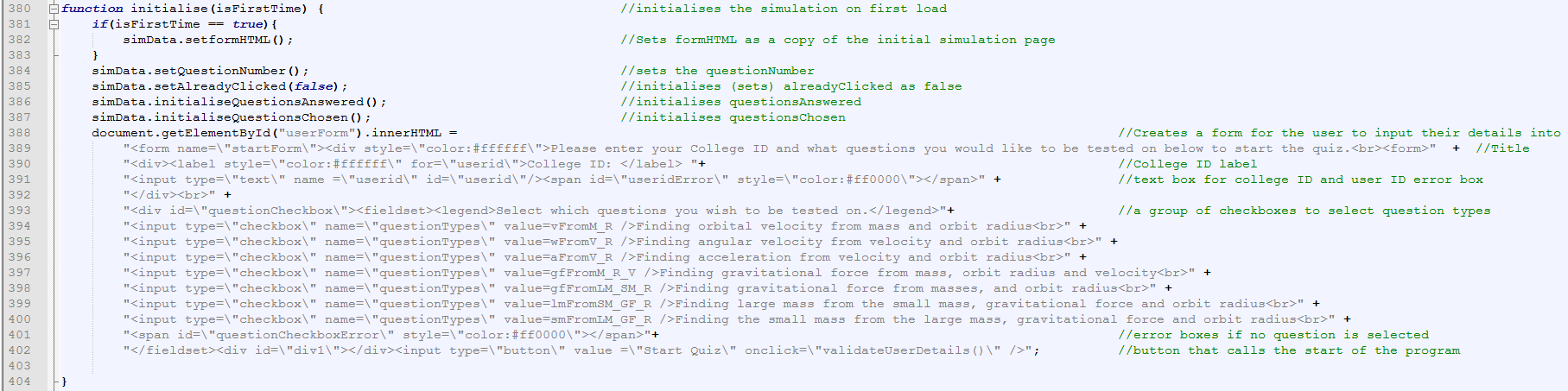
classBodyQuestion is the class that BodyQuestion is derived from. It’s constructed from the properties of bodyLarge and bodySmall, using them in its own properties. The purpose of BodyQuestion is to store the data to be used in the question. This means all of the rounded values are stored here, as are answer values, such as the gravitational force, or the angular velocity, which would otherwise never be created.

##### classSimulation



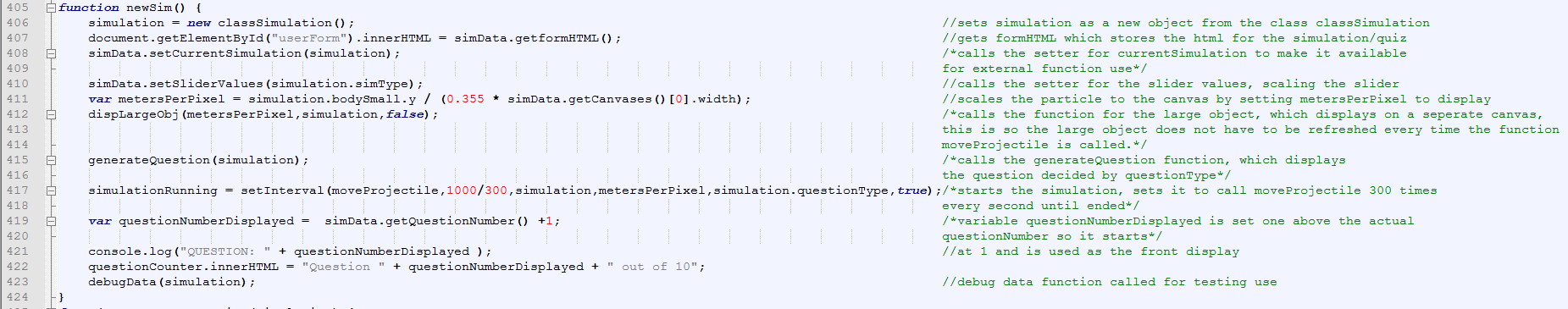
classSimulation is the basis for the object ‘simulation’. This class consists of 2 normal properties, simType and questionType, which are randomised integers between 2 and 0 and 6 and 0 respectively. These define the form of question, and the simulation displayed to highlight this question. The other 3 properties are objects in themselves, bodyLarge, bodySmall and bodyQuestion made up of the 3 above classes- classBodyLarge, classBodySmall, and classBodyQuestion. Together, this makes up the class, and provides all the data for a question in the program. 10 simulations are created throughout the course of the program, 1 for each of the questions.

##### Initialise



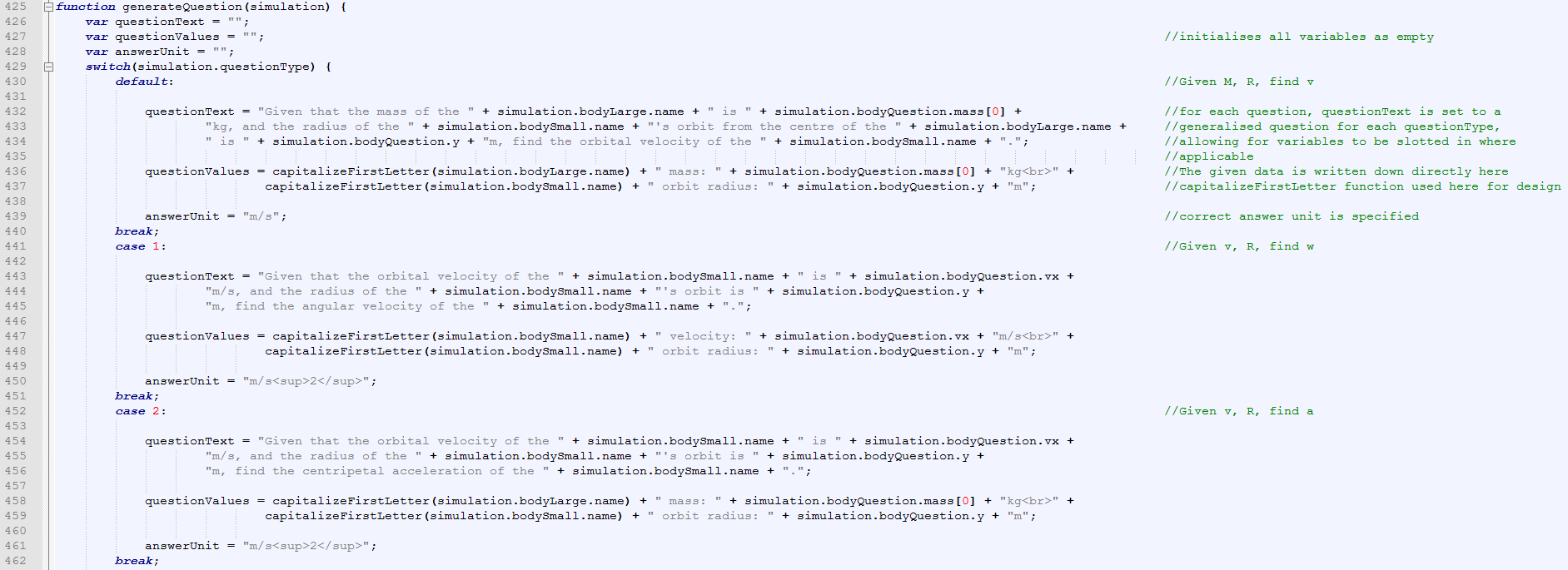
The initialise function is called when the program first loads, or when the program is told to go again at the very end of the program. It has a variable passed in , ‘isFirstTime’, to work out whether this is the first run of the program or not. If it is true, as should be the case whenever the program is loaded, it saves the current form to formHTML using the setter function. This saves the HTML from being permanently lost when it is overwritten. It then initialises and sets the initial values of some variables, including AlreadyClicked, QuestionsAnswered and QuestionsChosen. This ensures that all variables are ready to have their setter and getter functions called without an error occurring. After these have been initialised, the userForm is changed to resemble an input form. This was done by directly overwriting the html in a form. This includes a textbox for input of college ID, and a group of checkboxes to allow a user to pick and choose what question types they wish to be tested on. Both of these also are created with empty error boxes, which can be filled if an issue arises with the input. Finally, a button called submit is put at the bottom of the form, which when pressed finds if the data entered is valid. If this is not the case, the form stays until valid data is entered. If the data is valid, newSim is called.

##### NewSim



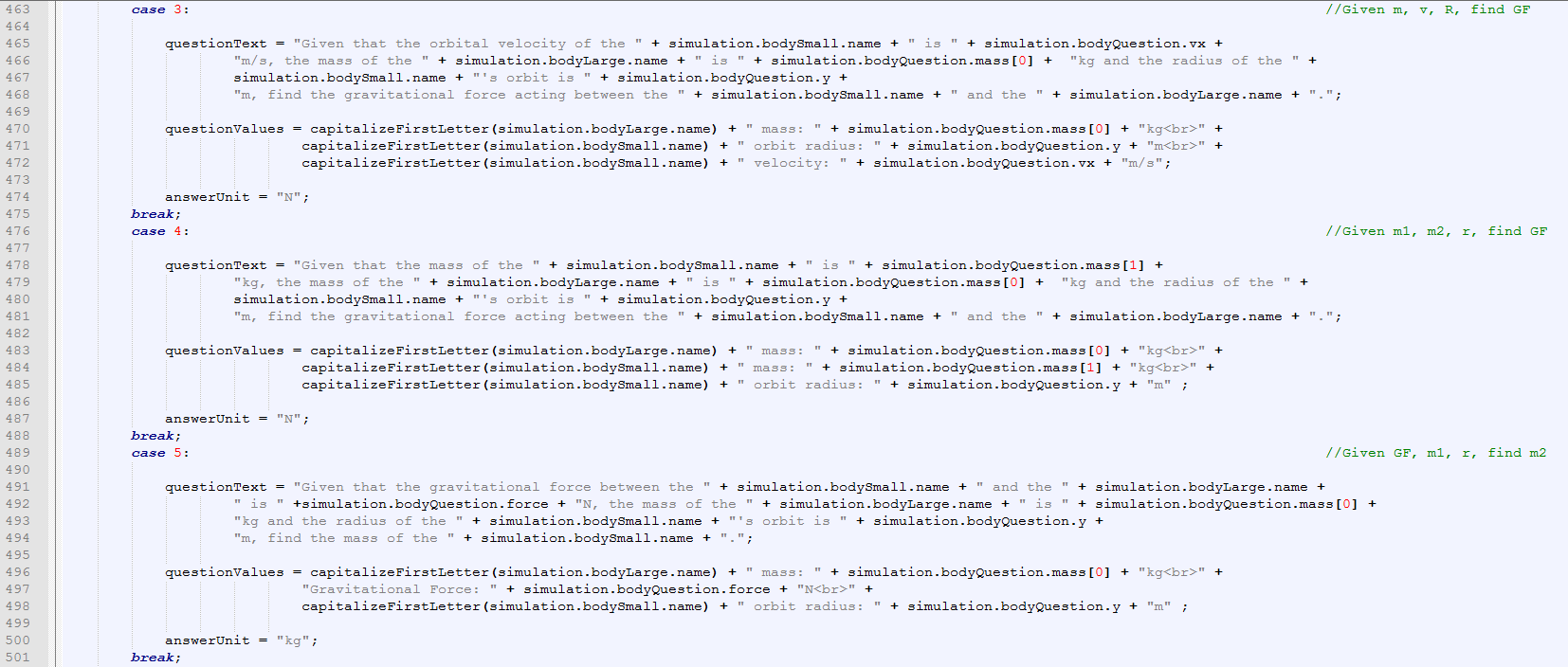
Newsim when run will firstly create a simulation of class classSimulation. This generates the question, as well as the data for the simulation. Following this, it uses the getter for formHTML to set the html back to the quiz screen. Furthermore, it also calls setCurrentSimulation to put simulation into the variable currentSimulation so it can be recovered from other parts of the program. It then calls setSliderValues. It then sets metersPerPixel, which is a scaling unit I have used to ensure that all of the simulation is present on the canvas, and does not fly off the canvas. DispLargeObj is called, which displays the large body in the middle of the canvas. GenerateQuestion is called, with the simulation object passed in. This displays the question for that simulation. SimulationRunning is set as an object of an interval moveProjectile, which means that it calls moveProjectile 300 times every second. The questionNumberDisplayed variable is set to equal one more than the actual questionNumber, since the quiz should start from 1, but the array starts from 0. The questionNumber is then displayed where necessary. Finally, it calls debugData, which outputs the data to the console. This would not be present in the end user copy.

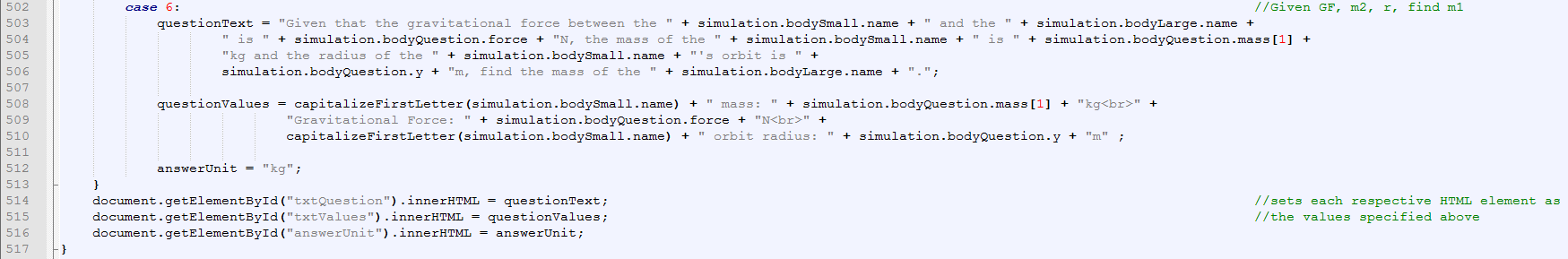
##### GenerateQuestion



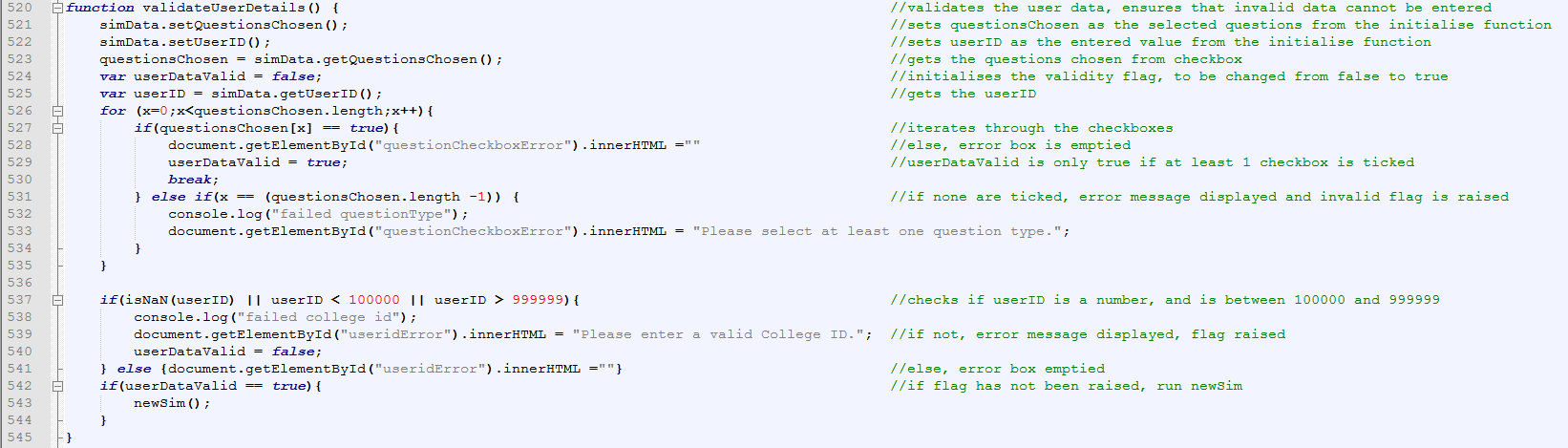
This function creates and displays the question. It does this by forming a switch statement, which chooses from the different questionTypes. Once the questionType has been selected, a question is built up using a template into which the data is inserted into. This also sets the unit that’s displayed next to the textbox, and the summary data. Once the data has been set, it outputs it onto the page.

**GenerateQuestion Continued**





##### ValidateUserDetails



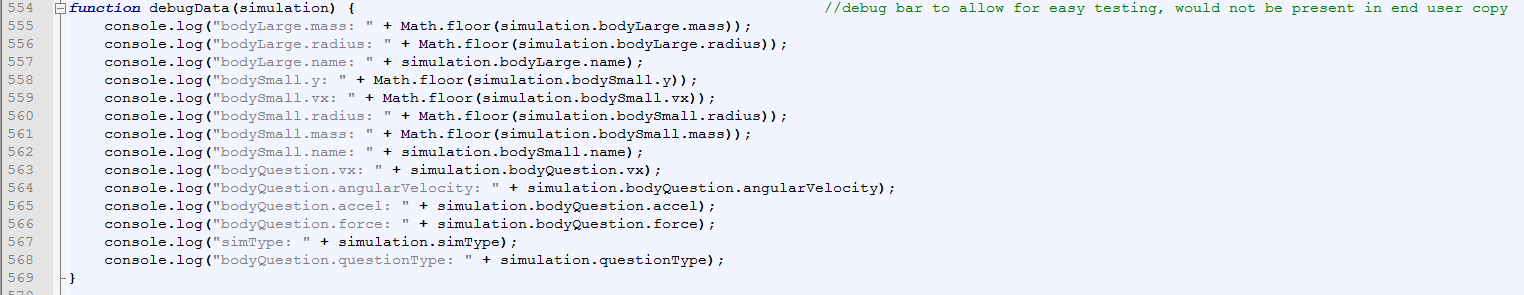
ValidateUserDetails is used as a validation check for the user form before the quiz starts. It recovers the variables from the userform using the setter and getter functions, and sets the UserDataValid flag to false. The fields are then tested. It checks questionsChosen, and iterates through all the checkboxes to see if any are checked. If one is checked, the flag is set to true and it breaks out of the loop. If it finds no checked box, the flag is kept as false, and the error box is displayed with a message asking for at least one question type to be selected. It then checks userID, and checks with an if statement that it is numeric, and 6 digits long. If it is not, an error message is displayed, and the validity flag is set to false.

##### CapitalizeFirstLetter

C:\Users\ciara\AppData\Local\Microsoft\Windows\INetCache\Content.Word\CapitalizeFirstLetter.png

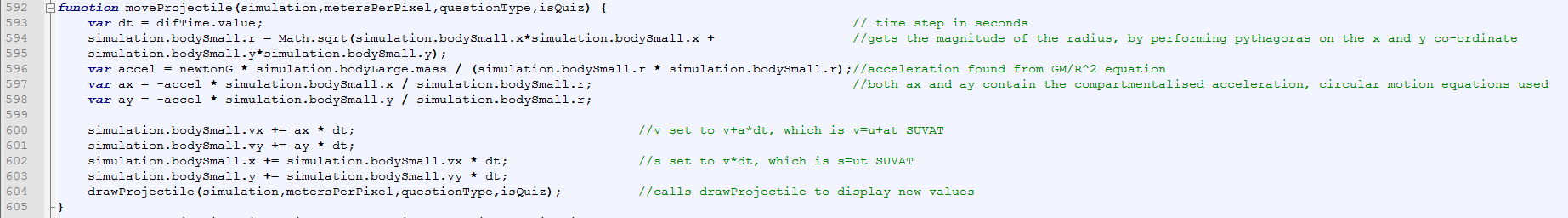
CapitalizefirstLetter is a simple function which returns the string it is given but with the first letter capitalized. It does this by splitting the string, setting the first letter to uppercase and then splicing the string together again.

##### DebugData



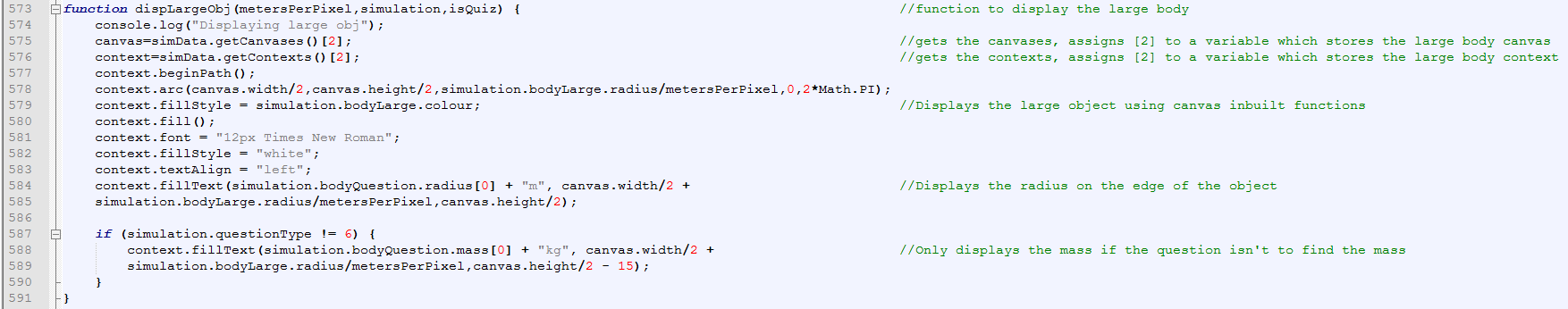
DebugData is a function that would not be present in the end user copy. It outputs the contents of the simulationData into the debug console for me to test. It will be removed in the end user copy so people cannot access the debug console and cheat.

##### MoveProjectile



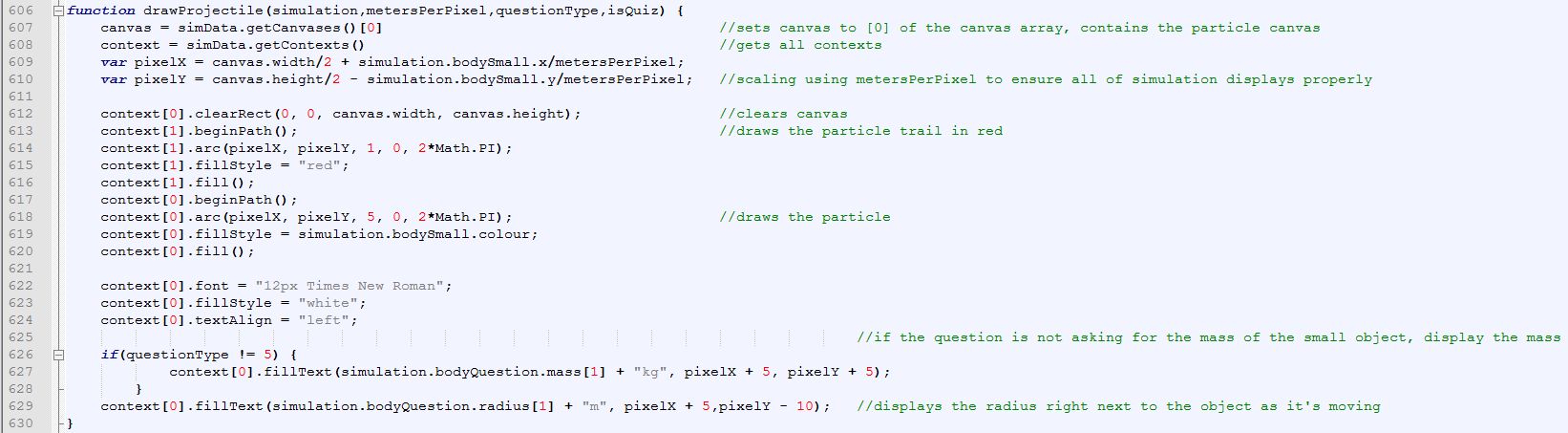
MoveProjectile is the basis for my simulation, as it performs all the necessary calculations. The variable dt which stands for the difference in time is set to the value of the slider. It then sets values in the simulation using mathematical equations, such as pythagoras’ theorem to calculate the radius from the x and y co-ordinates. Accel is calculated from the a = GM/R2 equation. This accel variable is then broken down into ax and ay, which is the compartmentalised acceleration. These are then applied to the vx and vy values using suvat, and the vx and vy values alter the x and y co-ordinates.

##### dispLargeObj



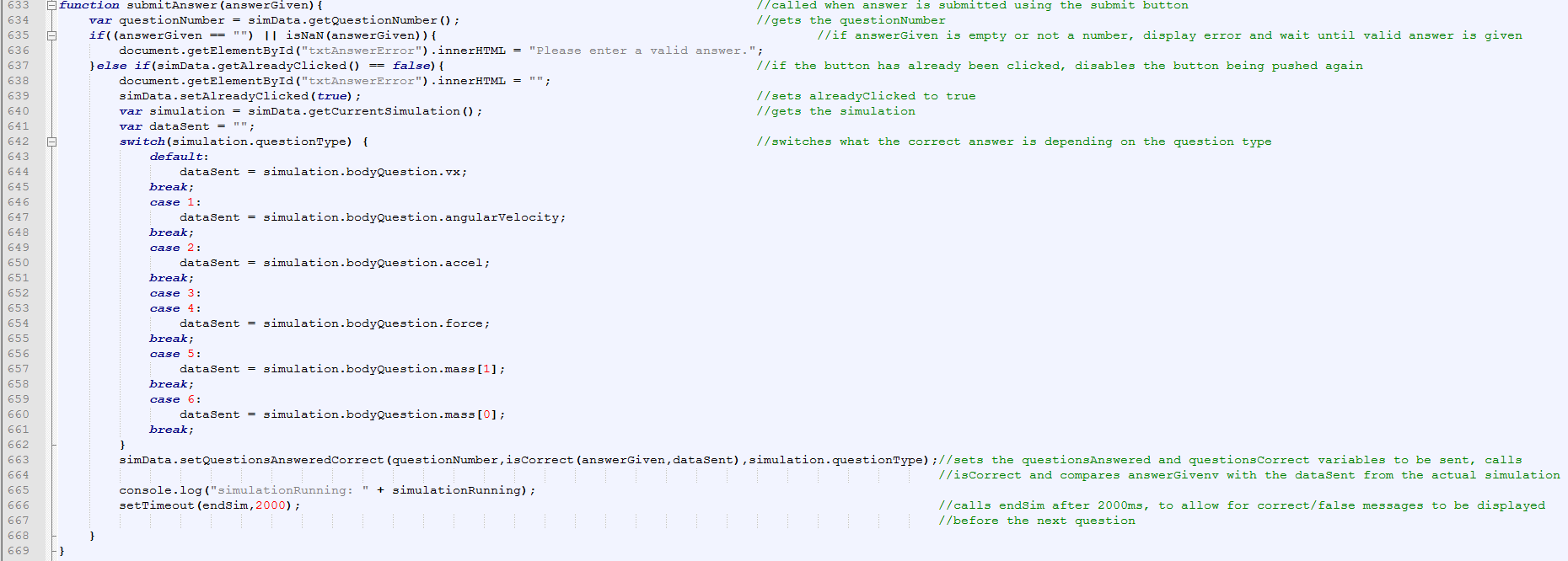
DisplayLargeObject will display the large body in the simulation when called. This is called on a separate canvas to the particle, so it does not have to be constantly applied to the canvas when the canvas is cleared to display the next position of the particle. It does this by using the getter functions for the canvas, and the context, of which is saves the canvas and context in array position 2, the large canvas. It then uses canvas inbuilt functions to display a circle of proportionate radius. It labels these with the radius, and the mass, but only if the mass of the large object isn’t the answer to the current question.

##### DrawProjectile



DrawProjectile displays both the path and the small body. It does this by firstly recovering the canvas and the contexts of both the small body canvas and the trail canvas. This is then drawn on using inbuilt functions of the canvas. The smallbody is given a default radius of 5 pixels, so that it is visible when displayed on the same scale as the large body, and the distance inbetween. X and y are used as raw values for the location of the object, however these values are divided by metersPerPixel to scale it to the canvas.

##### SubmitAnswer



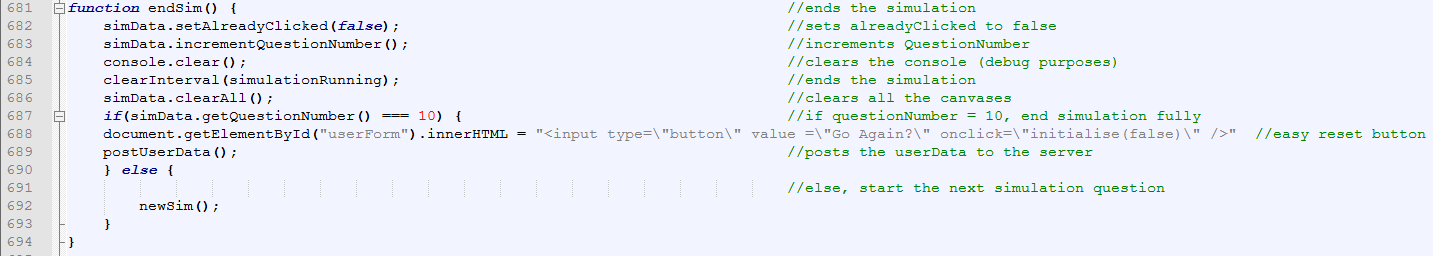
The submitAnswer function is called when the submit button is pressed. If the answer given by the user is valid, and it has not been clicked already, it sets alreadyClicked to true using the setter function. This stops the button from being rapidly clicked, and skipping forward multiple questions at once. It then finds the correct answer for the specified question using a switch statement on the questionType, and puts that into the function IsCorrect as well as the variable of the answer given by the user. This is then saved using the setter function for questionsAnswered and QuestionsCorrect.

##### IsCorrect



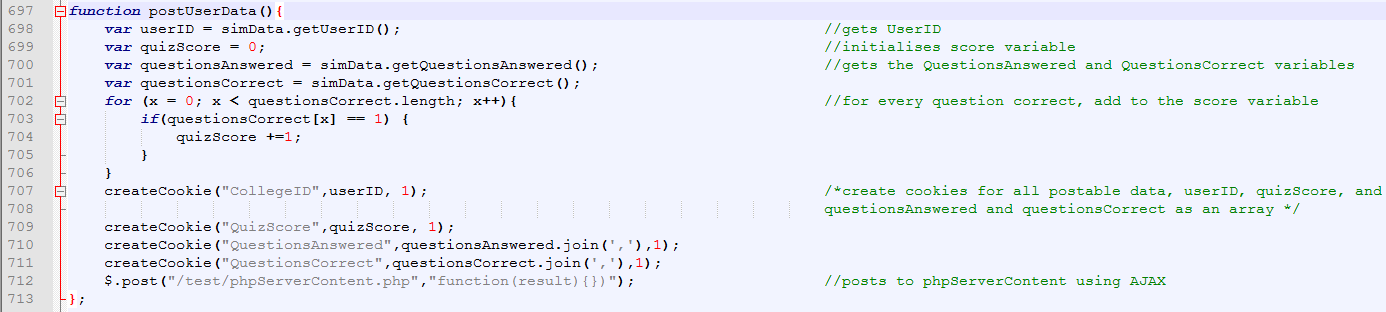
The isCorrect function has the true answer and the given answer passed into it. These are then put in to a percentage error calculation, and if the value is less than 5% then true is returned. It also sets a small message on the page to display “Correct”. If otherwise, incorrect is displayed, and false is returned.

##### EndSim



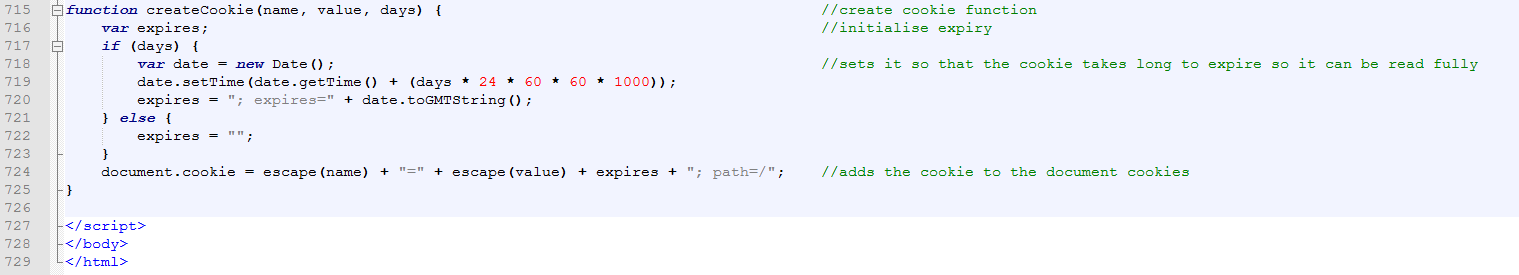
The endism function is called once the submit button has been pressed and the submitAnswer function has returned if it is correct or false. This increments questionNumber, resets variables using the set function of AlreadyClicked. The console is cleared, as is the simulation, and the clearAll function is called. This clears all the canvases. If the questionnumber is equal to 10, the quiz is over, and the postData function is called. If this is not the case, it moves onto the next question.

##### PostUserData



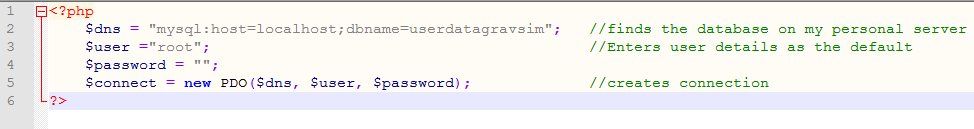
The PostUserData function posts the data to the database. It gets UserID using its get function, and quizscore is defined. The questionsAnswered and questionsCorrect functions are returned using their respective get functions. QuizScore is set to equal the number of correct questions. Finally, it puts all this data into cookies, and calls the phpServerContent php script.

##### CreateCookie



CreateCookie creates a cookie with the data that is inputted into it. It sets the time of expiry to many days in the future, and then creates the cookie using document.cookie.

##### Php\_to\_database\_connect



This php script is called once. It establishes a connection with the server, using the default username of root, and no password. The variable $connect then scores the connection.

##### PHPServerContent



This script is called at the end of the program. This recovers the cookies from the document, saving them into variables. It splits the array string that was sent across in a cookie back into an array, using the explode function. The connection is established, and an INSERT sql statement is called, and then executed using the variables given. A select statement is then called to recover the TestID given to the test, in the table ‘tests’. This is because it is an autoincrement number, and not created by me. Therefore to use the same TestID in testQuestions, it has to be found using the SELECT statement to ensure any connection made using foreign keys is correct. After inserting into tests, it sets up another INSERT SQL statement, and then iterates through the array of QuestionsAnswered. This will post all the questions individually to the table testquestions.

# Testing

### Youtube link

https://youtu.be/vblCQAvTOlk

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No. | What is being tested for | Expected Result | Given Result | Notes |
| 1 | Is userForm displayed and understood | Userform is displayed in a way that can be easily read | 0:13 | Passed |
| 2a | Is collegeID validated against letters being entered | Submit button does not let simulation continue, error message displayed | 0:38 | Passed |
| 2b | Is collegeID validated against letters and numbers | 0:51 |
| 2c | Is collegeID  Validated against numbers and symbols | 1:00 |
| 3a | Is collegeID validated against entries below 100000 | Submit button does not let simulation continue, error message displayed | 1:07 | Passed |
| 3b | Boundary case (outside range) | 1:34 |
| 3c | Boundary case (inside range) | 1:45 |
| 4a | Is collegeID validated against entries above 999999 | Submit button does not let simulation continue, error message displayed | 1:10 | Passed |
| 4b | Boundary case (outside range) | 2:05 |
| 4c | Boundary case (inside range) | 1:55 |
| 5 | Is collegeID validated against null entries? | Submit button does not let simulation continue, error message displayed | 2:20 | Passed |
| 6a | Is the number of questions selected validated against /no questions being selected | Submit button does not let simulation continue, error message displayed | 2:25 | Passed |
| 6b | /all questions selected | 2:40 |
| 6c | /some questions selected | 2:30 |
| 7 | When both fields are valid, does the quiz start? | Submit button, when pressed starts the quiz, question 1 displayed | 3:00 | Passed |
| 8 | Is questionnumber displayed? | QuestionNumber out of 10 displayed at the top of the form | 3:15 | Passed |
| 9 | Is the question displayed? | Question displayed below questionNumber | 3:20 | Passed |
| 10 | Is the question in an understandable format? | Values are rounded to 4 sig figs, comprehensible question | 3:40 | Exponent form, instead of standard form |
| 11 | Is the data recognisable in the question? | Values in the question to 4 sig figs, important data is repeated in note format | 4:15 |
| 12 | Is the entire simulation visible? | Particle is always visible on screen, does not fly off it | 4:35 | Particle remains on canvas, but can go off the edge of the page occassionally |
| 13 | Are the large and small bodies displayed? | Both the large and the small object are visualised, and appear | 4:55 | Large body is displayed to scale, small body is not as it would often be too small to be visible |
| 14a | Are the bodies identifiable for each simulationType? /Sun/Planet | Each body is named in the question, and specific colours are used to signify that object | 5:00 | Appropriate colours are used for each object, but pictures of each object would likely be better suited |
| 14b | /Planet/Moon | 12:50 |
| 14c | /Planet/Satellite | 10:15 |
| 15 | Does each body have important data highlighted about them? | The body will have information such as mass and radius displayed next to them on the simulation | 5:15 | Passed |
| 16 | Body does not detail information that answers the question | The information for the answer is removed from the simulation | 5:30,  8:00 | Passed |
| 17a | The speed slider has a reasonable scale for  /Sun/Planet | When put at maximum and minimum values, the simulation still runs at a reasonable speed | 5:50 | For planet/satellite, the simulation gets jittery at higher slider settings. This is because it is travelling large amounts of its orbit per frame, which will also cause the orbit to falter a bit from its main course due to it presenting an orbit that is not circular. |
| 17b | /Planet/Moon | 13:05 |
| 17c | /Planet/Satellite | 10:30 |
| 18 | The answer box is validated so only valid numbers are allowed | Error message displayed, answer not accepted | 7:10 | Passed |
| 19 | The answer box allows exponent form | Number is accepted, treated as any other number would be | 7:40 | Passed |
| 20 | The answer box does not allow non-numeric characters to be typed | Only the following values can be typed in 1234567890e+ | 6:15 | Passed |
| 21 | The answerbox cannot be left empty | Error message displayed, answer not accepted | 6:50 | Passed |
| 22 | Does the program advance to the next question if a valid answer is given | Correct/false message displayed, after a time the next question is displayed | 7:50 | Passed |
| 23 | Answer is still correct if it is a little amount off (+/-5%) | Correct message is displayed even if value is slightly off | 9:30 | Passed |
| 24 | Does the program recognise an incorrect answer? | Incorrect Message Displayed | 12:05 | Passed |
| 25 | Is a path drawn behind the moving body? | A red path should be drawn behind the moving body | 13:00 | Passed |
| 26 | Is an answer not involving exponent form still valid? | The response is treated as a normal number, and correct/wrong response is given accordingly | 14:25 | Passed |
| 27 | Is the data of the test stored? | The test is stored, with quizScore, CollegeID and date of test. TestID is used as a unique identifier. | 16:55 | Passed |
| 28 | Is the TestID of the test table the same as the TestID of the testquestions table? | The data should be stored under the same TestID. TestID is the primary key in tests, and part of the composite key in testquestions. | 17:40 | Passed |
| 29 | Has another user been created with the collegeID entered? | No new user should be created. | 19:15 | Passed |
| 30 | Are all 10 questions from the test stored? | All 10 questions should be stored in table testquestions. They will be stored under a composite key made from their college ID and the questionNumber. | 19:25 | Passed |
| 31 | Is the database connected with relationships? | All 5 tables should be connected with relationships made by foreign keys. | 20:00 | Passed |
| 32 | Are the questions stored with questionType and if they got it correct? | Each individual question is stored, containing questionNumber, TestID, correct?, and questionType, with questionNumber and TestID forming a composite key. | 17:20 | Passed |

# Evaluation

Andy’s Response

On the plus side the simulator is very easy to use, with associated selection boxes to allow for a bespoke approach to questioning students on various topics. The answers can be expressed in terms of proper answers or answers which have an associated rounding error of 5%. The questions are clear and concise and aid in learning. The processed data which can be drawn from the simulator is also impressive, allowing the teacher to track students via their college based student numbers to ascertain their understanding.

From analysing Andy’s response, I can see that I have performed well in the quiz assessment area of the program. The variety of questions available as well as the error margin of 5% provide ease of use for the student using the program. One of my requirements was that each question was easily comprehensible for the student, and I feel that from this response I have achieved that. Furthermore, the ability to select specific questions at the start of the simulation allows for each test to be directed manually towards areas of focus. Finally, he has complimented the amount of data that can be recovered from each test. This, when implemented into the college database, could be collated with other scores in other areas of physics, which would allow a teacher to gauge the ability of the student. I feel that therefore my program meets the intended purpose, especially when the test data that I have created is actually replaced with the college database.

When a student wishes to use the simulation, there is no option for ‘free play’ meaning that the students will be required to answer the questions from the start, and those data regarding the answering of the questions will be supplied to the teacher.

What is being stated here is that I have not provided an ability to let the students mess around with the values from the simulation, and see the effects of having a sub orbital speed for that velocity, or even having above escape velocity. This means the simulation is only useful for the quiz section, which, while useful, does not provide a firm “sandbox” for the teacher to demonstrate in front of the students. I have taken this into consideration, and I feel that this addition would be easily possible if I had more time. This would have been a suitable extension to my project, as it provides a firm benefit to my end user.

The final improvement thought would be in relation to the simulator itself. Being a simulator of sorts, it would be more appropriate to call it a question generator, so to make it a true Astrophysical simulator, try to add a model environment which can be used adapted to aid in learning e.g. input data fields for two orbiting bodies – the simulation can run then has a question related afterwards such as “why did the satellite crash into the planet?” etc.

This follows on from andy’s last point, in that the simulation could be used for more than a simple physics quiz. The sandbox option would allow users to interact with sliders to change the initial conditions and how that affects the simulation. His other suggestion is one that would be harder to implement – A qualitative quiz answer. Whilst what the question would be testing (collision) is easy to create, to provide a question for it would provide some difficulty. This is because a question such as “why did the satellite crash?” does not have a numeric answer, and would therefore mean examining strings. However, to work out as to whether the string contains the question or not would be a near impossible task. This is because there are so many ‘correct’ statements that could be made concerning this question, and therefore a Correct/False marking would not be suitable for this. I would consider this a long term extension goal.