BOUNCING BALL SIMULATOR: DOCUMENTATION

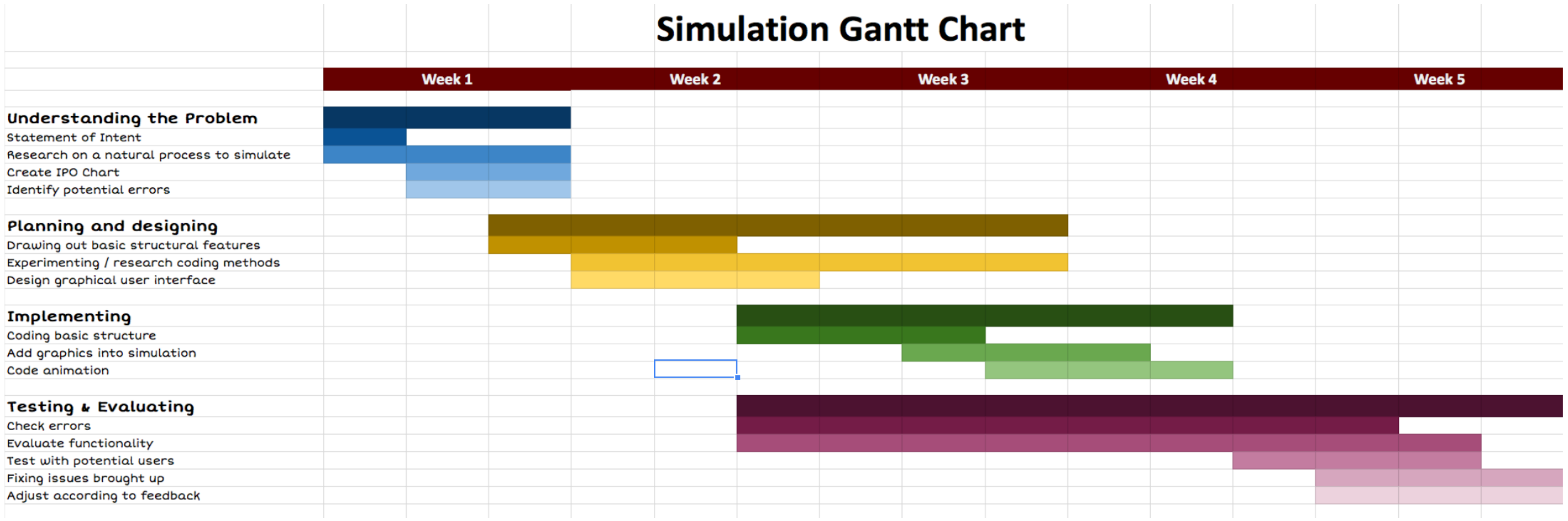
Software Design and Development Year 11 Task 4: Group Project

Alexandra Cruz
Tate Adams
Eric Pullakaran
Lavanya Sood

Statement of intent:

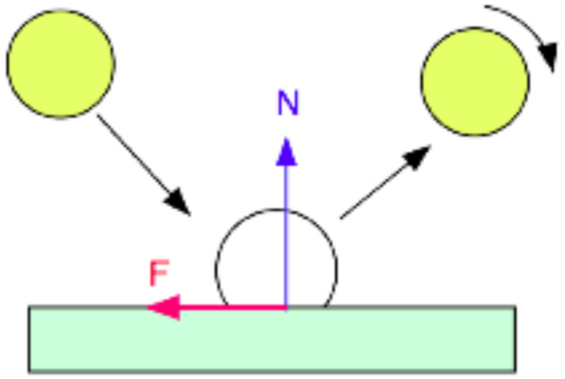
Reason for developing the simulation of physics forces (gravity, mass, acceleration) is to further develop coding skills. This will also help our group project management skills by carrying out a formal development approach (that incorporates more structure). This project will be executed using the Javascript developing skills learnt through Khan academy and through help with teachers and fellow students. The simulation that will be created is the bouncing ball, that simulates gravity and frictional forces, the ball will react to these forces resulting in it following in a path of a bouncing ball. The GUI design will be carried out using Photoshop and Illustrator and will follow GUI aspect guidelines, e.g. white space, colour schemes etc. Features that will be implemented include, user interactions that allow for inputs. These inputs will allow for the user to change mass, colour, position etc., thus changing the simulation. There will also be an option for changing gravities to again create different instances of simulations. The position input will allow for the user to click with the mouse to change its start location.

Gantt Chart:



Research:

1. A ball falls vertically downward under the influence of gravity (*g*). The velocity *V* points downward. The acceleration *a* also points downward. The magnitude of *a* is equal to *g*, in the absence of air resistance.
2. Then the ball begins to make contact with the surface. It continues to fall vertically downward under the influence of gravity. The velocity (*V)* and acceleration (*a)* (equal to *g*) both continue to point downward.
3. The velocity *V* is still pointing downward. However, the ball has deformed sufficiently such that the acceleration *a* is now pointing upward. This means that the ball has deformed enough such that it's pushing against the surface with a force greater than its own weight. As a result, the acceleration *a* is pointing upward.
4. the ball has reached its maximum deformation. As a result, the acceleration *a* is still pointing upward, and the velocity *V* is zero.
5. the ball's velocity *V* is increasing and pointing upward since the ball is now in the rebounding stage. As a result, the ball is less deformed than in the previous stage, but is still deformed enough such that it's pushing against the surface with a force greater than its own weight. This means that the acceleration *a* is still pointing upward.
6. The ball now barely touches the surface. The velocity *V* is still pointing upward since the ball is still in the rebounding stage. However, since the ball is no longer deformed it has essentially zero contact force with the surface. This means that the only force acting on the ball is gravity. As a result, the acceleration *a* is now pointing downward, and the upward velocity *V* is now decreasing.
7. the ball has fully rebounded and has lifted off from the surface. The velocity *V* is still pointing upward, and the acceleration *a* is still pointing downward since the only force acting on the ball in this stage is gravity.



The formulas that will be used are the frictional force and the f =ma formula. The formula works by stating that the mass is proportional to the force while the acceleration is inversely proportional to the mass. The frictional formula is Frictional force = -1vnormal where mu is the frictional coefficient and V is the velocity and the normal force is perpendicular to the interface.

IPO chart:

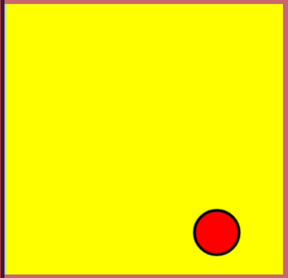
Input	**Process**	**Output**
The user will be able to input the mass of ball. (keyboard input range limit)	1. Enter mass 2. Enter gravity 3. Get position 4. Calculate object mass into F = ma equation through pVectors 5. Display ball bouncing 6. End	Displays the size of the variables that the user inputted, and speed at which the ball bounces.
The user will be able to input different strengths of gravity through buttons and a custom gravity through a keyboard input. (different planets)	1. Enter gravity 2. Enter mass 3. Get position 4. Calculate gravity and object mass into F = ma equation through pVectors. 5. Display ball bouncing 6. End	Displays the ball bouncing at a speed according to gravity input.
The user will be able to input the starting position of the ball by clicking a position in the screen.	1. Get position 2. Translate into x and y coordinates 3. Display ball in the x and y coordinate 4. End	Displays the initial position of the ball.
The user will be able to change the colour of the ball through buttons.	1. Get colour 2. Display ball with chosen colour 3. End	Displays the ball with the chosen colour.

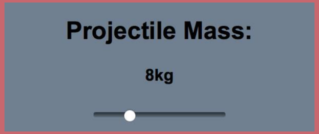
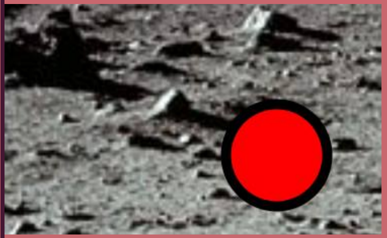
Logbook of programming:

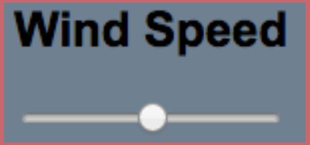
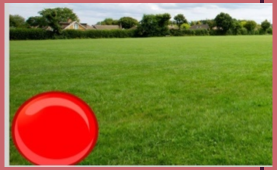
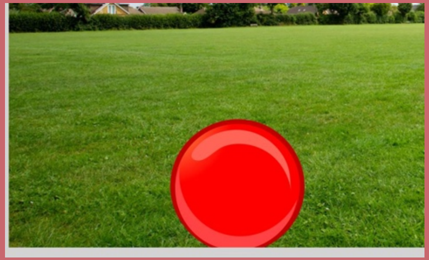
	Plan to achieve in the code	**Issues Encountered**	**How these issues were solved**
Date: 30/7/16 **Coder(s):** Eric	Create a PVector function to hold the position/location of the object at all times by adding velocity to the object in both x and y coordinates to allow further development with forces and formulas.	I was successful in creating the PVector function but experienced errors around the addition of velocity to the object.	I fixed the error by calling the variables for individual (x,y) by this.y and this.x
Date: 4/8/16 **Coder(s):** Eric	Make the ball visible on click.	For this I didn't know why the ball would increase in speed when clicked	To solve this a bit of code rearrangement was required to allow for more object oriented programming eliminating the layering of instances which was the issue
Date: 8/8/16 **Coder(s):** Lavanya	Make the user input a value for the mass and to be able to change the look of the ball.	Plan was successful as there were no issues encountered.	

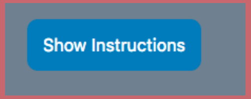
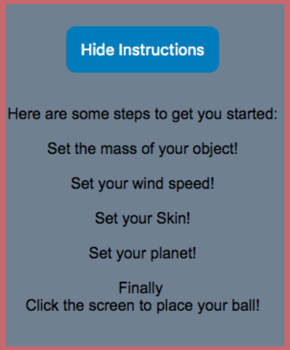
Date: 10/8/16 **Coder(s):** Lavanya	To change the color of the ball and the background of the interface by creating a drop down menu for the user to pick their desired color of the ball and the background.	Plan was successful as there were no issues were encountered.	
Date: 12/8/16 **Coder(s):** Alex	Add an input for the user to change the force of gravity acting on the ball.	It was unsuccessful as the input did not change the velocity at which the ball was bouncing	A variable was created to change the value of acceleration/gravity into the equation that changes the ball's velocity when a value is input.
Date: 13/8/16 **Coder(s):** Alex	Create a drop down menu for the user to choose which planet the ball is placed, to change the force of gravity acting on the ball.	Plan was successful as there were no issues encountered.	
Date: 15/8/16 **Coder(s):** Eric	Fix the ball falling off the screen.	Plan was successful as there were no issues encountered.	
Date: 17/8/16 **Coder(s):** Alex	Add instructions in the GUI on how to use the simulation	Plan was successful as there were no issues encountered.	

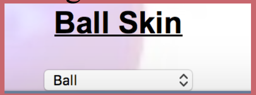
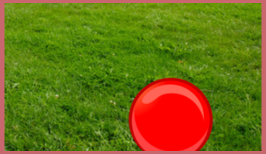
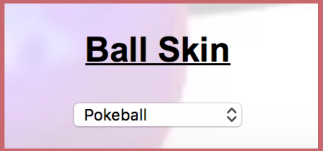
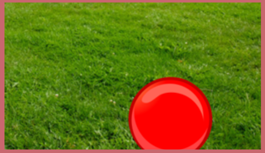
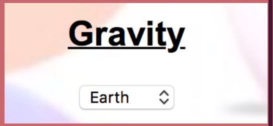
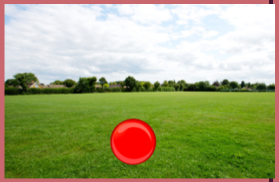
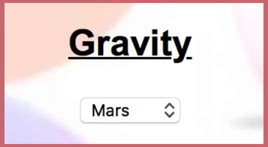
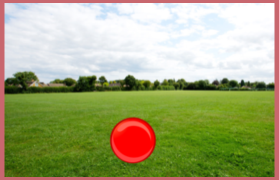
Testing Documentation:

Module	Input	Expected Output	Output	Solution
Creating input for ball position	User mouse click	Translate mouse click to X and Y coordinates and change ball position to suite location then add effects such as gravity etc.	 Not clearly seen in the image but as user repeatedly clicks the mouse the ball function is redrawn even faster incrementing by every mouse click	To fix this issue I had to rearrange my code to allow for more object oriented programming since, when the set-interval function was called it, would create an instance of it, as it was called multiple times it would layer these instances resulting in a faster animation of the ball being displayed. After changing some code we were able to call only the update function upon mouse click meaning that after every click it would just use the same interval function. Additionally the ball variable was initially defined as a “this.” variable meaning it could only be accessed through the specific function, a simple change of this into a standard variable allowed for the ability to access the function through different functions, thus fixing the issue.

Change the mass of the object	User inputs a desired value	Changes the mass and the size of the ball according to input	The mass stays the same no matter what the value is:  Projectile Mass: 8kg	To fix the error the mass was placed in the update function and then a new function named 'edit()' was created when the mass inputted by the user was converted into an integer and then updated, thus the mass of the ball changed.
Change the gravity / acceleration of the object	User selects a celestial object (each with different gravities)	Velocity of the ball changes	Ball bounces at the same velocity no matter what celestial object was selected 	To fix the issue a separate function was created to pass these values of alternate gravities. This successfully fixed the issue allowing for the ball to fall at different rates to the ground.

Changing the wind speed	User inputs wind speed by adjusting the bar of wind speed input (either negative or positive)	Ball moves to the direction of the wind Ball changes speed according to the effect of the change in the wind speed	Issue was the ball was displaying to be moving in accordance to the wind but having no display of the extent at which it was being moved 	To fix this a new h2 title was created which stored the default value of 0 m/s, with an Id i then accessed it through the javascript and changed the value to the value of the slider
Checking the edges to allow the ball to bounce	When a ball bounces the edges of the SVG are checked so that if the ball is in contact with the edge of the SVG it bounces back	As soon as the ball touches the edge it bounces back 	The ball bounces back before touching the edges 	To fix this issue whenever the ball was reaching the left edge originally it was checking the radius of the ball and thus was leaving the space between the actual edge and the edge set because the the radius thus while checking the left edge instead of setting the variable 'this.position.x' to an integer and thus we are able to overcome the edge problem

Creating button	User clicks and instructions are displayed	The instructions are displayed with correct sizing	No instructions were displayed even after button click  Show Instructions	To fix this issue I had to create a on-click function and change the display type  Hide Instructions Here are some steps to get you started: Set the mass of your object! Set your wind speed! Set your Skin! Set your planet! Finally Click the screen to place your ball!
Ball speed	User click	Ball will obey newton's laws and bounce normally	Ball would continually get faster and faster as it fell (screenshot wouldn't be beneficial in this situation)	To resolve this issue I multiplied the acceleration with 0 upon every update function to stop the ball adding incrementing acceleration

Changing the skin of the Ball	The user selects a skin from a drop down menu	The skin of the ball changes  	The skin of the ball wouldn't change while the ball was bouncing  	To fix the problem I had to change the style of the simulation to only support one ball at a time during the simulation.
Changing the backgroun d according to the selected planet	The user selects a planet gravity from a drop down menu	The background changes to the background of the planet selected  	Background does not change when a planet was selected  	To fix the problem a grav function was created to get the id of the photo depending on the user input of what planet gravity they selected

L.S. \|E.P.\| A.S.\| T.A.\|

Creating air friction/resistance for the ball	The ball is placed by a click,	the air resistance is a force that is applied to the ball while the ball is in motion	The force wasn't applied to the ball while in motion	To fix the problem of air resistance the function had to be moved to a location where the air resistance would be applied properly to the ball.

Evaluation:

There were many ups and downs throughout this whole project. However, the effective time management and productivity ensured the successful development of the physics simulator. The tasks were allocated appropriately according to time constraints and executed depending upon the difficulty of the task at hand. Although this project was executed well, there were issues that arose due to understanding of some coding aspects. These issues were investigated and fixed, when we decided to undertake the khan learning course that taught coding fundamentals to allow for the ability to successfully create code. As mentioned throughout the report all GUI elements in the plan were in the end successfully implemented and even some new features were added into the simulation, such as sliders for wind speed etc. for gravity selection, originally it was an entry box but to improve the user interactions, and for efficiency the buttons were created. In the end we finished the project according to the GUI plans etc. When in creating the simulation time constraints helped but the development process was faster than anticipated thus leading to the project finishing earlier than planned and allowing for further implementation for additional features into the game such as web layout (adding buttons for instructions, changing colors etc.). We think we effectively used the Gantt chart and other planning methods to successfully create the simulation.

Self Evaluation:

Eric Pullakaran:

Throughout the project I developed features that include the mouse positioning, which consisted of just pressing the mouse and for the ball to display at that location, also programed the gravitational forces, etc. **My percentage 30%**.

Lavanya Sood:

Throughout the project I created various smaller functions for the program to function, I created the mass input function for the user. I also added the different ball skins and the background of the HTML document, I worked on the visuals of the simulation. **My percentage is 27%**

Tate Adams:

Throughout the project i assisted with code concepts and code segments of different modules, and i also assisted with creating the visuals of the project. I also helped with formatting and creating the documentation. **My percentage 20%**

Alexandra Cruz:

I helped in the research for a natural simulation and the physics behind the simulation. For the code, I added the drop down menu that allows the user to select a planet to change the force of gravity acting on the ball, changing the velocity of the ball as well. Also, I contributed to the design of the GUI of the simulation by selecting appropriate background colours, and aided in designing some of the visuals in the html and added instructions on how to use the simulation. For the documentation, I greatly contributed to the testing documentation, and in organising the logbook. **My percentage 23%**