University of Utah

Madadasa

Principia: Design Document

Authors:
Sam Olds
Daniel Peterson
Matthew Turner
Dalton Wallace

Supervisor:
H. James
de St. Germain

February 21, 2016

Contents

1 Executive Summary

Physics is one of the core subjects that college-bound high school students will explore, especially for those that plan to pursue a degree in a scientific field. However, many instructors have doubts about how well high school courses are preparing students for their college experience. In the Third International Mathematics and Science Study (TIMSS), 18 participating countries tested student populations from elementary, middle, and high school on a broad range of physics concepts. The average result was that students were only able to answer 35% of the questions correctly [1]. For U.S. students, the average dropped to 24%.

Clearly, there is a need for improvement in this area. Our team believes that we can provide a web-based solution that would have the power to make a difference both for high school students and for students taking college level physics for the first time. Our project will be called Principia, named after *Philosophiæ Naturalis Principia Mathematica*, the collection of books written by Newton that serves as the foundation of classical mechanics.

We envision Principia as something that is in equal parts an engaging community experience and a powerful tool to model physical systems. Students will have a support network to help them overcome challenges and a chance to share their knowledge with others. Instructors will have a website they can use to cover material in a deeper and more captivating way than a traditional lecture that uses static images.

The core feature will be an interactive sandbox for designing systems. It will include a toolbox that contains components like point masses, springs, and pulleys. Users can drag and drop these components into a central canvas. Within the canvas, any instance of a component can be selected and its properties will be displayed on the side of the window. See Section 3.3 (System Features) for an enumeration of the components we will support and Appendix A for additional figures demonstrating our plans for the UI.

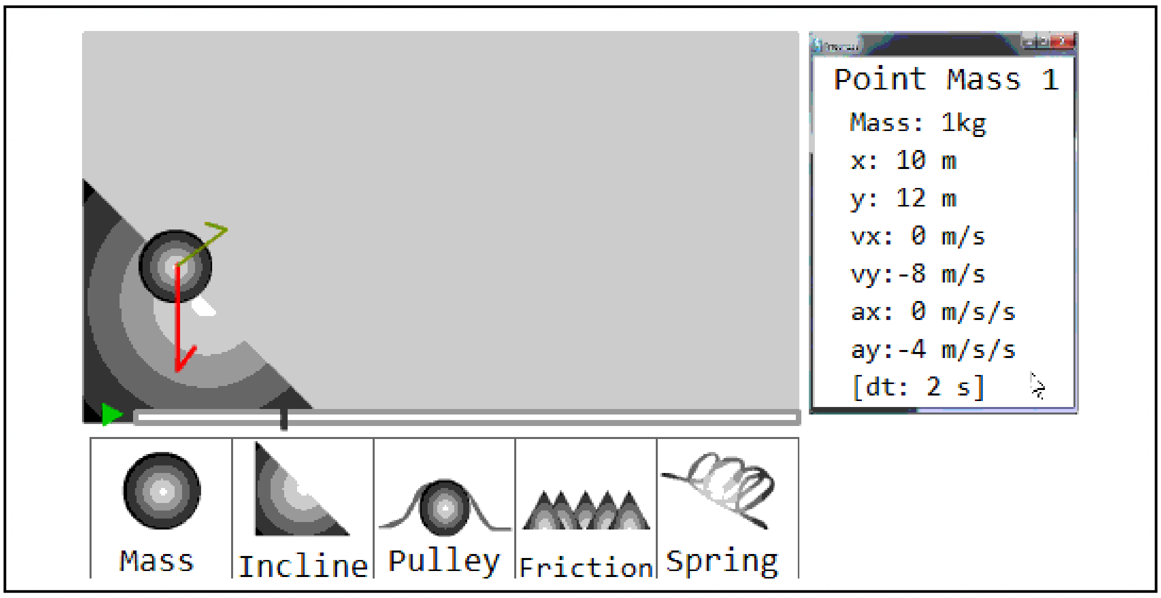

Point Mass 1
Mass: 1kg
x: 10 m
y: 12 m
vx: 0 m/s
vy:-8 m/s
ax: 0 m/s/s
ay:-4 m/s/s
[dt: 2 s]

Figure 1: Concept for interactive sandbox

Once the components are in place, users can select play and the canvas will come to life. The components will move over time according to their properties, allowing students to visualize the system. At any point, users can pause the simulation and select a component to view its properties, allowing them to get quantitative results in addition to the visualization. We believe that by combining animated graphics with numeric descriptions, students will be able to develop an intuition for what kind of behaviors are associated with otherwise incomprehensible equations and formulas.

Though the sandbox is the core feature and an ambitious project in its own right, it is far from Principia's only selling point. To enable users to connect with and support one another, the website will allow them to create accounts. A registered user can save and load his creations, share them with others, browse simulations, provide comments and annotations, and even create quizzes or walkthroughs. Imagine a future where instructors integrate Principia into their course. No longer will students work in isolation, writing out equations based on a fixed image or an ambiguous word problem. Instead, students will have the chance to have a dialogue with both the instructor and their peers, exploring problems interactively or even inventing their own systems.

We see the final version of our project as a top site for physics education and a powerful supplement to material covered in the classroom. Teachers, students, and the growing community of individuals interested in online learning at their own pace will all be able to take advantage of Principia.

2 Background

2.1 Idea Space

We believe that the most memorable physics lessons were the ones with demonstrations. The best lecturers came to class with props and experiments used to reinforce the lesson's material. However, students don't always have the means to recreate the demonstrations to get a better understanding of the underlying concepts. Furthermore, not every instructor has the resources or time to prepare practical demos.

Principia would solve both of these problems. Students would be able to easily build their own simulations of the in-class demonstrations and tweak components to see how their changes affect the end result. This would make learning physics more visual and interactive, and ultimately more fun! Instructors would also be able to share simulations, eliminating the need to purchase special equipment or to spend precious class time setting up an experiment.

The fundamental problem we wish to address, however, is the poor state of physics education itself. Dr. John W. Brelsford studied the effects of virtual reality specifically within the context of physics education. His found that "As a training method, virtual reality was superior to the [lecture-based] control condition at the four-week retention period. Such a finding supports cognitive theorists who argue that the lack of opportunities for hands-on, manipulation of objects in the physical world is one of the reasons children are often poor at intuitive physics. Virtual reality provides them the opportunity to develop manipulational skills they did not previously possess" [2].

Principia's sandbox is precisely the kind of tool that improved physics knowledge retention in Dr. Brelsford's study. The addition of support from an online community will make for even better software. Principia will improve the state of physics education on a worldwide scale.

2.2 Similar Ideas

While we assert that there is significant room for innovation and improvement in the area of physics education, we acknowledge several existing educational offerings.

2.2.1 Commercial Software

There is a wide array of commercial software for physics. Some are focused on a very specific domain for professionals, such as molecular dynamics or high-energy physics. This kind of software would not be useful to the audience we are targeting. Other examples that do focus on high school or college physics, like Stewart Software's Creative Physics® 5.0, cost as much as $149.99 for a single-user license [3]. We want to provide software that has a lower barrier for entry and that puts improving the state of physics education above financial gain.

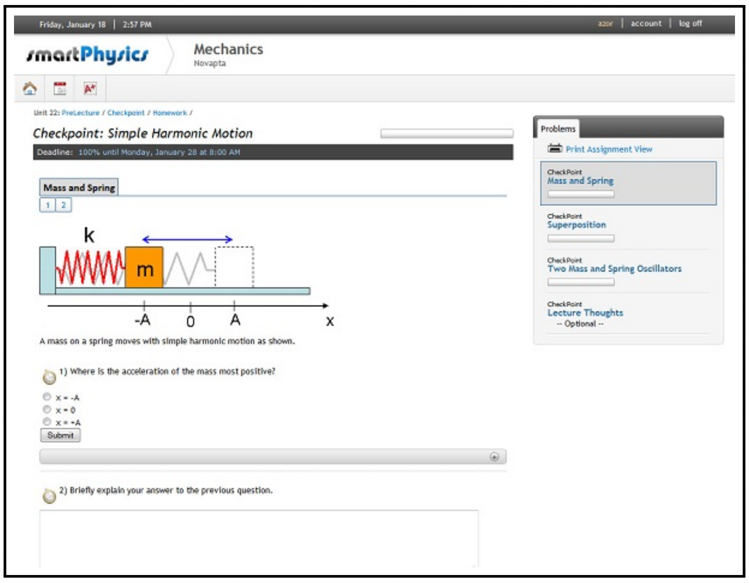
2.2.2 Physion

Physion is the closest existing software to what we want to provide. It is highly interactive, free, and has a modest online community. However, we believe we can improve on it by having Principia's sandbox run in-browser instead of as a separately downloaded executable and by providing greater support for getting quantitative data out of the simulation.

2.2.3 smartPhysics

Arguably the most prominent offering on the current market, smartPhysics provides students with additional examples and resources to help solidify concepts learned in the classroom. This is primarily accomplished through online “pre-lectures” and homework material. In our experience with smartPhysics, we have found that it lacks interactivity, relegating users to working on practice problems involving systems that the smartPhysics developers have already created. For instance, the following figure is a sample image of a checkpoint concerning simple harmonic motion. This checkpoint will always involve a single spring attached to a single mass. The practice problem can be varied by adjusting the values associated with the amount of mass or the spring constant, but there is no way to see the system in motion or to experiment with how additional components like a second spring or surface friction would affect its behavior.

smartPhysics is a great way of introducing concepts, but it is possible to do much more. Our goal with Principia is to provide a new level of interactivity, allowing a user to explore concepts and ideas specific to her needs by building her own systems or modifying existing ones rather than being limited to “canned” systems as in this example.



Unit 22: PreLecture / Checkpoint / Homework /

Checkpoint: Simple Harmonic Motion

Deadline: 100% until Monday, January 28 at 8:00 AM

A mass on a spring moves with simple harmonic motion as shown.

1) Where is the acceleration of the mass most positive?

- x = -A
- x = 0
- x = +A

2) Briefly explain your answer to the previous question.

Figure 2: Smart Physics checkpoint [4]

2.2.4 PhET

PhET is an open-source collaboration developed by a group of researchers from UC Boulder. The project aims to provide “fun, free, interactive, research-based science and mathematics simulations”. While PhET offers a measure of interactivity, the number of physics topics it covers is relatively limited. It is more dynamic than smartPhysics, as should be apparent in the following figure, but it is still limited by the fact that users are tweaking the properties of an existing system instead of building their own in a less restrictive environment.

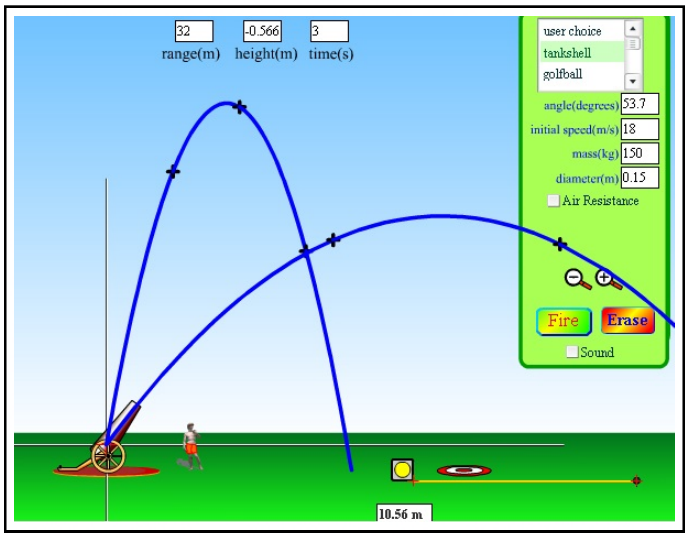
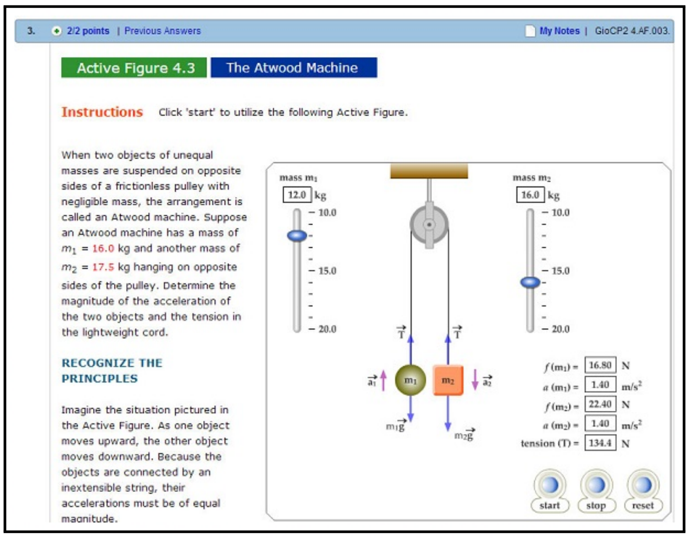


Figure 3: Projectile motion in PhET [5]

2.2.5 WebAssign

WebAssign allows educators to set up online environments designed to test student knowledge and offer additional instruction. Principia would expand upon this idea by not only providing an environment for instructors to engage their students but also allow students to query their classmates with specific questions and concerns. In addition, we feel that Principia would provide more freedom when creating quizzes or exams. For example, with Principia teachers could pose questions that would require students to modify a specific portion of a simulation and explain how their change affected the simulation's outcome.



3. 2/2 points | Previous Answers My Notes | GioCP2 4.AF.003.

Active Figure 4.3 The Atwood Machine

Instructions Click 'start' to utilize the following Active Figure.

When two objects of unequal masses are suspended on opposite sides of a frictionless pulley with negligible mass, the arrangement is called an Atwood machine. Suppose an Atwood machine has a mass of m_1 = 16.0 kg and another mass of m_2 = 17.5 kg hanging on opposite sides of the pulley. Determine the magnitude of the acceleration of the two objects and the tension in the lightweight cord.

RECOGNIZE THE PRINCIPLES

Imagine the situation pictured in the Active Figure. As one object moves upward, the other object moves downward. Because the objects are connected by an inextensible string, their accelerations must be of equal magnitude.

mass m_1 12.0 kg

mass m_2 16.0 kg

$f(m_1)$ = 16.80 N

$a(m_1)$ = 1.40 m/s^2

$f(m_2)$ = 22.40 N

$a(m_2)$ = 1.40 m/s^2

tension (T) = 134.4 N

Figure 4: Active figure in WebAssign [6]

2.3 Required Technology

Principia will require the following technologies:

- A web server where the application can be hosted.
- A database to hold information about users and their saved simulations.
- A simulation window nested in an HTML page that allows for interactivity.
- A physics engine that will accurately update the simulator's physics model and render the results.

2.4 Assets and Engines

To avoid rebuilding the wheel for every component, we will be using the following languages, services, and libraries:

- Go will be the language used on the server and back end.
- Google App Engine will be used to host the web application.
- HTML, CSS, and JavaScript will all be languages used on the front end.
- The HTML <canvas> element will be used for the simulator window.
- To help with some of the difficult physics components, we will utilize the open source JavaScript library, PhysicsJS. It has in-built support for rendering into an HTML canvas and will allow us to write high-level code for the physics model instead of creating it from scratch.
- To help provide a consistent theme and feel across browsers, Twitter's Bootstrap CSS library will be used.
- AngularJS will be used to make the front end more user friendly and responsive.

2.5 Software Requirements

In order to fully use Principia, the user will need a computer with access to the Internet and a web browser. We plan to support all of the mainstream browsers, including Google Chrome, Safari, Mozilla Firefox, and Internet Explorer. Since it is a web application, the site will be available on mobile devices; however, we don't plan to explicitly support mobile devices. We will make a "best-effort" approach analogous to "best-effort" packet delivery, i.e. we make no functional guarantees! No separate download or special hardware will be required to use the simulator. It will be easier to build a community around our project if it has a low barrier for entry.

3 Requirements Analysis

3.1 System Architecture

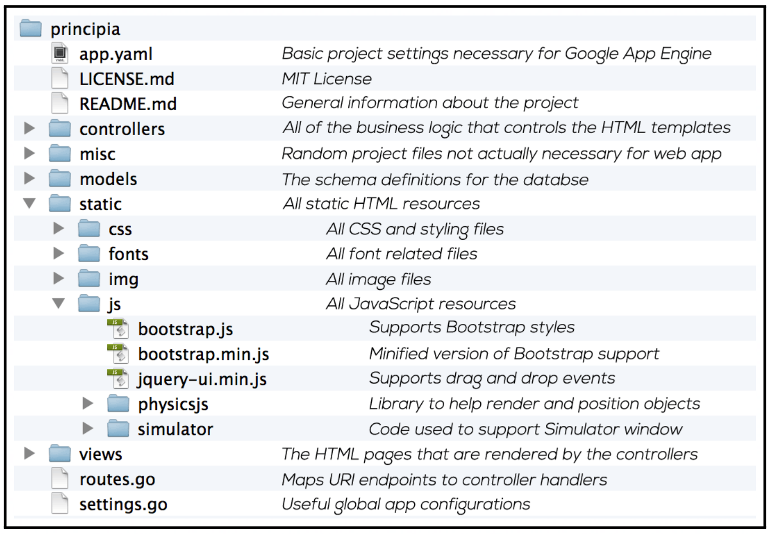
Back End:

Our web server is responsible for routing requests and putting together the HTML templates that will end up as a complete page in the user's browser. It is also responsible for allowing users to log in to access MyPrincipia (see section 3.2.2) and persistent data storage.

- Google App Engine (G.A.E.): The platform we are using to build Principia
- Using the MVC Pattern
  - Models - Google App Engine's Big Table Database: Our means of persistent data storage
  - Views - HTML files rendered with Go's template package: Our means of displaying our application
  - Controllers - Written in Go: Our means of routing requests and assembling templates into an HTTP response
    - Simulator
    - User
    - Comment
    - Group
    - Assignment
- Unit Testing with Go

Front End: The simulator will be written in JavaScript and run on the font end within the user's browser.

- HTML pages constructed from server templates
- Twitter Bootstrap
- JavaScript
  - Base Simulation Module: Logic shared by all simulators and event handler for user interactions
  - Physics Simulator Modules: Logic specific to a particular physics concept
    - 1D Kinematics
    - 2D Kinematics
    - Statics
    - Work and Energy
    - Angular Motion
  - jQuery and jQuery-ui
  - IntroJS/JoyRide for tutorials
  - Jasmine for unit testing



principia	
app.yaml	*Basic project settings necessary for Google App Engine*
LICENSE.md	*MIT License*
README.md	*General information about the project*
controllers	*All of the business logic that controls the HTML templates*
misc	*Random project files not actually necessary for web app*
models	*The schema definitions for the databse*
static	*All static HTML resources*
css	*All CSS and styling files*
fonts	*All font related files*
img	*All image files*
js	*All JavaScript resources*
bootstrap.js	*Supports Bootstrap styles*
bootstrap.min.js	*Minified version of Bootstrap support*
jquery-ui.min.js	*Supports drag and drop events*
physicsjs	*Library to help render and position objects*
simulator	*Code used to support Simulator window*
views	*The HTML pages that are rendered by the controllers*
routes.go	*Maps URI endpoints to controller handlers*
settings.go	*Useful global app configurations*

Figure 5: Server Directory Structure

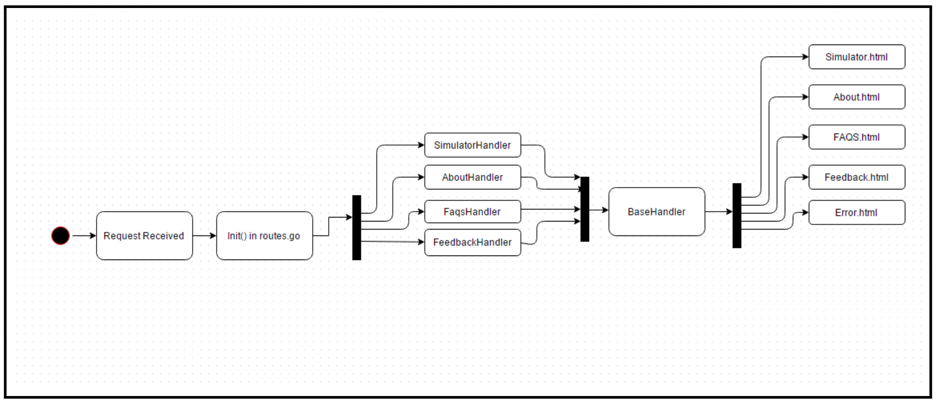


Figure 6: Activity Diagram: Making a Request

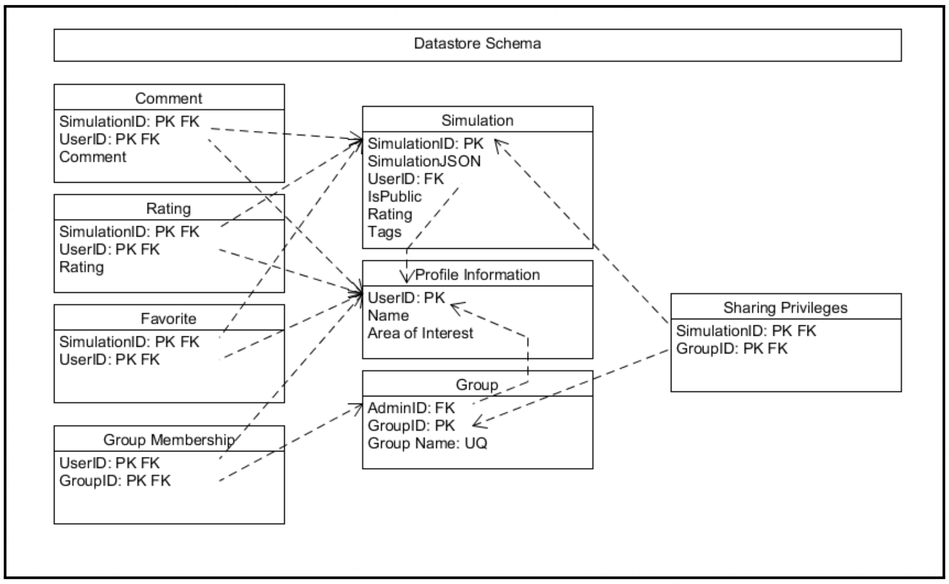


Figure 7: Entities for Datastore

3.2 Personnel Responsibilities

- **Matthew Turner**: Matthew's interest in Computer Science began after a high school physics teacher showed him a lecture on *A New Kind of Science*, by Stephen Wolfram. His main responsibility will be working on the model underlying the simulator. He has prior experience working with physics models from another project as well as having performed well in an undergraduate physics course.
  - Lead Simulation Programmer
    - Work on base simulator JavaScript code
    - Understand and utilize the PhysicsJS library
  - Secondary Tester
- **Dalton Wallace**: Dalton has gained industry experience working as a full stack engineer for multiple companies using frameworks like ASP.net and AngularJS. He enjoys tinkering with personal projects and keeping up with the latest web technologies and design. He is passionate about web development and the creative freedom it allows and looks forward to contributing to project principia.
  - Lead Datastore Programmer
    - Work on Go models
    - Understand Datastore fundamentals
  - Secondary Front End Design
- **Daniel Petersen**: Danny has web software coursework experience in which he worked with a group to help implement the current teaching assistant application for the computer science department. He enjoys learning new technologies on his own using online learning platforms.
  - Lead Front End Design
    - Focus on look and feel of website
    - Understand and implement Google users API
    - Work on tutorial walk through
  - Lead Tester
  - Secondary Datastore programmer
- **Sam Olds**: Sam has web development industry experience and has deployed a handful of side projects in his free time. He is familiar with a variety of different web frameworks and how to manage these kinds of projects. Because of his industry experience he is also accustomed to working with a team using management tools such as Git, Jira, and Slack.
  - Lead Back End Architecture
    - Set up server in Go and deploy with G.A.E.
    - Manage application structure and data flow
    - Git, Go, and G.A.E. wrangler
  - Secondary simulation programmer

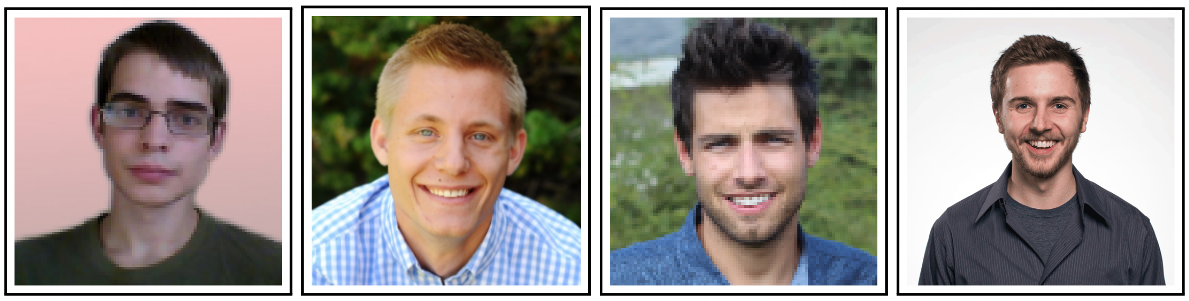


Figure 8: The wonderful members of Madadasa

3.3 System Features

3.3.1 Base (Rank 1)

- Web Server: Integral to the success of our project, this component will serve site pages and facilitate data storage.
- Database: Utilizing datastore we will persist user data for those who have registered.
- Simulator: The core of our project, a collection of JavaScript files to handle modeling physical systems, simulating their behavior over time, and providing means for visualizing that simulation as well as extracting numeric data at any point.
  - Modes of Operation
    - Standard Mode: Allow users to drag and drop components into an initial frame and simulate up to 1000 additional frames with a limited set of options for the time-step between each frame.
    - Keyframe Mode: Allow users to add keyframes that define the state of the simulation at different points in time and possibly including components with unknown properties. The simulator will attempt to solve for unknowns and then fill in the timelines between each keyframe.
  - Elements
    - Viewport: HTML element where simulation is displayed.
    - Toolbox: Container for physics components that a user can drag and drop onto the viewport.
    - Timeline: Allows users to run the simulation or to select a particular frame and query the current state of each component
    - Properties window: Allows users to edit global simulation settings (e.g. constant for gravity, distance per pixel) and settings for a component selected in the viewport (e.g. mass, position).
- Registration: Users will be given the opportunity to register for a Principia account
- Browse Simulations: The site will feature a search function, allowing visitors to browse public simulations

3.3.2 Planned (Rank 2)

- MyPrincipia: A collection of features to foster a sense of community and aid in getting effective simulations into the hands of other users.
  - Groups: Users can form groups to more easily manage permissions
  - Commenting: User can comment on simulations to either provide feedback or questions
  - Sharing: Users can release their saved simulations publicly or to a specific group
  - Rating: Users can rate simulations to help others find the most informative examples for a particular topic
- Advanced Simulation Settings: Allow users to model more complex scenarios by providing additional settings in the properties window
  - Change of reference frame (relative to origin, relative to point, relative to another component)
  - Change of coordinate frame (Polar or Cartesian)
  - Change of units (Base application will use Metric units, planned support will include Imperial units)

3.3.3 Advanced (Rank 3)

- Change physics components to render as whatever image you upload (accessible through properties window)
- Annotated Walkthroughs:
  - Add clarity to a given simulation
  - Highlight key takeaways
  - Accessible through toolbox - drag and drop “annotation” elements with a text property and a property for how many frames it is displayed
- Canvas Integration: Allow students to submit simulations to be graded via Canvas by Instructure.
- Quiz Generation: Create questions associated with a given simulation
- Property Visualization: Allow users to create charts for properties like position and velocity of a particular component
- Viewport Navigation: Provide support for panning around the viewport and resizing it while maintaining the state of the simulation.
- Collaborative Simulation Creation: Multiple users can edit one simulation simultaneously.

4 Techniques

4.1 Development Methodology

We strongly believe in iterative development. Very rarely does a completed project resemble what the design documents described. The best method to build a desirable product is through iterations of development and feedback, so we will be practicing an Agile process. We will have two week sprints with sprint planning, sprint review, and sprint retrospective meetings on the first Monday, second Monday, and first Monday of the following sprint respectively.

During sprint planning we will address all of the work we would like to complete during the sprint and who will "own" each task. During the sprint review, we will work together to help anyone who is blocked and reassess the assigned tasks of the current sprint. During sprint retrospectives, we will examine how the sprint went and try and make changes to the next sprint to ensure success.

4.2 Versioning

Git is becoming an industry standard for version control. We will use Git to manage our distributed development process, simultaneous feature construction, and product releases. Our plan is to have each team member checkout a new branch to work on a feature. At the end of a sprint, we will resolve any conflicts and merge complete features into the master branch, which can then be pushed to our website. We are also interested in making our work open source, so our repository will be publicly available on GitHub (see [7]).

4.3 Bug Tracking

Since we are already utilizing GitHub, we want to take full advantage of the features it offers. GitHub has a inbuilt set of tools for creating bug tickets ("issues", see [8]). Every issue will be appropriately labeled to indicate whether an issue is part of an upcoming enhancement, or is a bug with some existing feature. Tickets can be prioritized, assigned to specific developers, and reference each other using markdown. We considered using some of the alternative tools available, such as Jira, but decided that a "depth not breadth" approach would allow us to have a deeper understanding of our toolset and a centralized place to manage our project.

4.4 Testing

We are planning on using the behavior-driven development framework Jasmine for testing front-end JavaScript code. In addition to JUnit/Jasmine, Go's native "testing" package will be used to unit test logic in the back-end. We will be continuously testing as we use agile development practices to add new features to our project. The results and tracking of tests will also be accounted for using Github tickets.

4.5 Documentation

Meeting minutes have been taken during every meeting and stored in the "Wiki" section of our GitHub repository. This Wiki is also used to document general things the team should stay up to date with, such as API endpoints, technologies being used, and overall architecture. Naturally, we fully intend to follow industry standards in terms of documentation within our source code as well, so we will provide inline comments and method headers. The JSDoc project (see [9]) will allow us to produce HTML files containing an API reference for our project as well as tutorials associated with specific functions.

4.6 Team Communications

Our primary means of communication (other than talking in person) is Slack. Slack allows us to have a permanent record of group discussions as well as providing neatly organized channels so we know where to look for records concerning a particular topic. We will use email and group text messaging for transitory messages.

4.7 Team Meetings

In person team meetings will be held regularly, three times weekly. We will spend the first five minutes coming up with an agenda. In general this will include going over progress made for the current sprint, refining or setting sprint goals to accomplish before the next meeting, and then using the remaining time for pair programming. In addition, we will meet over Slack on Sundays to review progress for the week as a whole and focus on any outstanding issues. In addition to our regularly scheduled meetings, additional meetings may be held as deemed necessary. We will use an online calendar to organize

5 Timeline

We will use this timeline as our guideline for what we hope to accomplish for Principia through Spring 2016. At the end of each production phase, we will perform a code review, fix outstanding bugs, and make enhancements to existing features instead of working on new ones. We will also ensure our documentation and wikis are up to date at that point.

Week	Matthew Turner	Daniel Petersen	Dalton Wallace	Sam Olds
Alpha				
1	Keyframe Support	New CSS And Layout	Persistence Robustification	New CSS and Layout
2	Coordinate Systems	Homepage/signup	Profile Page	Server Error Handling
3	Specify Units	User groups	User groups	API Robustification
4	UI Additions/Refinement	Comments and upvote/downvote	Unit Testing	JavaScript Cleanup
5	Stabilize/Bug Fixes	Stabilize/Bug Fixes	Stabilize/Bug Fixes	Stabilize/Bug Fixes
Beta				
6	Component: Ropes/Pulleys	Multiple simulation pages	Route Refactor	Component: Ropes/Pulleys
7	Component: Inclines	Search Simulations	Search Simulations	Component: Inclines
8	Component: Friction	Component: Friction	Sharing Simulations	Showcase Page
9	Component: Levers	Showcase Page	Component: Levers	Unit Testing
10	Stabilize/Bug Fixes	Stabilize/Bug Fixes	Stabilize/Bug Fixes	Stabilize/Bug Fixes
Production				
11	Statics Module	New User Tutorial	Statics Module	New User Tutorial
12	Work/Energy Module	Annotated Walkthroughs	Annotated Walkthroughs	Statics Module
13	Viewport Navigation	Viewport Navigation	Canvas Integration	Quiz Pages
14	Flex Week	Flex Week	Flex Week	Flex Week
15	Stabilize/Bug Fixes	Stabilize/Bug Fixes	Stabilize/Bug Fixes	Stabilize/Bug Fixes

insta

6 Unregistered User

6.1 Register

Description:

An unregistered user will be able to register for a user account, user registration will be exclusively through Google's Users API

Steps:

1. Anytime an unregistered user attempts to perform an action requiring an account, the user will be prompted to register for a free Principia account
2. Registration will take place with Google's User API and will require an active gmail account
3. Upon successful registration, the user will be redirected to their user profile where they can customize their account settings.

7 Registered and Unregistered Users

7.1 View Public Simulations

Description:

An unregistered user will be able to browse public simulations of interest via a search function.

Steps:

1. User navigates to the search bar and inputs topic of interest
2. Screen displays public simulations created by other Principia users relating to user search keywords
3. User can view/run specific simulation
4. User can view comments but may not add comments
5. User can copy simulation and modify, however cannot persist forked simulation without registering

7.2 View Other User's Profiles

Description:

An unregistered user will be able to browse public user profiles via a search function, the unregistered user will also be able to view the author's user profile from his/her public simulation.

Steps:

1. While browsing simulations, the user will have the ability to view the author's user profile
2. User profile will display the following user data:
   (a) Join Date
   (b) Public Simulations
   (c) Areas of Interest
   (d) Groups
   (e) Comments
   (f) Liked Simulations
3. Search function will also allow users to search for specific users by username

7.3 Using the Simulator: 1D Kinematics

Description:

A student is working on an assignment and needs to model a kinematics problem: A car is traveling at 72 km/h when the brakes are applied, providing a uniform acceleration of -2m/s^2. How long does it take the car to stop? How far does it travel?

Steps:

1. The user navigates to the simulator page
2. The 1D Kinematics module is selected
3. When prompted for global settings, he selects ”keyframe mode” and otherwise uses default values
4. From the toolbox, the user drags a point mass anywhere into the simulator window
5. The user selects the point mass to edit its properties:
   (a) The user clicks ”select image” and chooses a car from the default options
   (b) The user sets the value of ”x” to 0 m.
   (c) The user sets the value of ”v” to 72 km/h
   (d) The user sets the value of ”a” -2 m/s$\hat{2}$
6. The user creates a new keyframe on the timeline and when prompted for a time selects ”unknown”
7. The user continues using the properties window of the selected point mass:
   (a) The user sets the value of ”x” to ”unknown”
   (b) The user sets the value of ”v” to 0
8. The user selects ”solve”; the unknown values are filled in with ”10s” and ”100m”
9. The user can run the simulation to see the car in motion

7.4 Using the Simulator: 2D Kinematics

Description:

A professor is teaching about 2D kinematics and uses the following problem as a lesson: A boat detects an unknown ship 19.4 km away in the direction 15.6 degrees east of north. The unknown ship is traveling 30 km/h in the direction 42.3 degrees east of north. If a speedboat that can travel 52.2 km/h is sent to intercept it, what direction should it go in degrees east of north?

Steps:

1. The professor navigates to the simulator page
2. The 2D Kinematics module is selected
3. When prompted for global settings, she selects ”keyframe mode” and otherwise uses default values
4. From the toolbox, the user drags a point mass anywhere into the simulator window
5. The user selects the point mass to edit its properties
   (a) The user clicks ”select image” and chooses a boat from the default options
   (b) The user sets ”x” and ”y” to 0 km
   (c) The user sets ”v” to 52.2 km/h in the direction ”unknown”
6. From the toolbox, the user drags a point mass anywhere into the simulator window
7. The user selects the second point mass to edit its properties
   (a) The user clicks ”select image” and chooses a boat from the default options
   (b) The user changes the position options from ”relative to origin” to ”relative to object” and selects the original point mass
   (c) The user changes the coordinate system from ”Cartesian” to ”Polar”
   (d) The user sets ”theta” to 15.6 degrees from north and ”x” to 19.4 km.
   (e) The user sets ”v” to 30 km/h in the direction 15.6 degrees east of north
8. The professor creates a new keyframe on the timeline and when prompted for ”t”, selects ”unknown”
9. The user selects the second point mass and changes ”x” to 0 (i.e. 0 km from the first)
10. The user selects ”solve”, which solves for the unknown values of ”direction” and ”t”
11. The user can run the simulation to see the boats in motion up to the point they meet
12. The professor confirms the result is what she expects, then deletes the second keyframe so students can set it up for themselves.
13. See C.7 for how the professor could allow students to fork her simulation and C.6 for how she could view her students' progress.

7.5 Using the Simulator: Statics

Description:

A user is modeling the following problem: A sign 0.5m across has one rope holding it up at the very end and another rope holding it up 0.375m from the end.

Steps:

1. The user navigates to the simulator page
2. The Statics module is selected
3. The user selects a 1D mass and drags it anywhere into the simulator window
4. The user selects the 1D mass and sets the length to 0.5m
5. The user drags in a tension object and attaches it at 0m of the mass
6. The user drags in a tension object and attaches it at 0.375m along the mass
7. The user selects the first tension object and sets "T" to "unknown"
8. The user selects the second tension object and sets "T" to "unknown"
9. The user selects the mass and sets it to "1kg"
10. The user selects "solve" and the unknown tensions are filled in

Additional Notes: There is no timeline for the statics module

7.6 Using the Simulator: Work & Energy

Description:

A user is modeling the following problem: A student is pulling on a rope to drag his sled across the ice. He pulls upwards and rightwards with a force of 25 Newtons at an angle of 40 degrees above the horizontal to drag his sled a distance of 100 meters. Determine the work (in Joules) done upon the backpack.

Steps:

1. The user navigates to the simulator page
2. The Work & Energy module is selected
3. When prompted for global settings, he selects "keyframe mode" and otherwise uses default values
4. The user selects a point mass and drags it anywhere into the simulator window
5. The user drags a force vector onto the mass
6. The user selects the force vector to edit its properties
   (a) Its magnitude is set to 25N
   (b) Its direction is set to 40 degrees above horizontal
7. The user inserts a new keyframe
8. In the new keyframe, the user selects the point mass and sets "x" to 100m.
9. After running the simulation, the user selects the force vector on the final frame and checks the "work done" property

7.7 Using the Simulator: Angular Motion

Description:

A user is modeling the following problem: A mass attached to a 1m rope does 2 rotations in one second. Find the period, tangential velocity, and angular velocity.

Steps:

1. The user navigates to the simulator page
2. The Angular Motion module is selected
3. When prompted for global settings, she selects ”standard mode” and otherwise uses default values
4. The user drags an axis into the simulator window
5. The user drags a mass into the simulator window and places it 1m from the axis
6. The user selects the mass and sets the ”frequency” property to 2 rotations/second. The other properties are automatically solved for.
7. The user runs the simulator to see the rotating mass in motion
8. To observe how it affects the problem, the user adjusts the ”period” property. The other properties are automatically solved for.
9. She runs the simulator again to see the new result

8 Registered Users

8.1 Create Profile Page

Description:

Upon registering, the new user will then be given the opportunity to set up their profile.

Steps:

1. User will be able to view their list of simulations or create a new simulation
2. User will be able to accept group invitations or create new groups
3. User will be able to view past comments and favorited simulations
4. User will also be able to modify account settings which will include:
5. Setting a username
6. Setting account to public/private
7. Modifying their "Areas of Interest"

8.2 Save Simulation

Description:

A user can persist changes they've made to any simulations they have created, they will be able to choose visibility privileges, these privileges will dictate who is allowed to view, fork and directly modify their simulation.

Steps:

1. User creates a simulation
2. Upon completion or exit, they can save the simulation to their list of simulations to be modified or view later.
3. User can save simulations with the following privileges:
4. Public: Any registered user can fork the simulation, anyone can view it
5. Private: Only owners can view, edit and delete.
6. User will specify who is allowed owner privileges (the ability to modify the simulation as if it were their own)
7. User will also specify who is allowed to view the simulation (cannot directly modify simulation but can fork and modify copy)

8.3 Sharing Your Own Simulation Privately with Other Users

Description:

A registered user might want to share their saved simulation with people. But they still might want to keep the simulation private. A user will be able to specific other users to privately share their simulation with.

Steps:

1. Sign in
2. Create simulation and save it as private
3. Click share button at bottom of simulator window
4. New modal window pops up to allow user to specify list of other users to share it with
5. Click “Share!” button in modal window

8.4 Sharing Your Own Simulation Privately with Someone as Another Owner

Description:

A registered user might want to collaborate with another user on a simulation. They can share their simulation with another user and gave them Owner permissions.

Steps:

1. Sign in
2. Create simulation and save it as private
3. Click share button at bottom of simulator window
4. Enter username to share with
5. Click “Owner” to give that user owner permissions
6. Click “Share!”

8.5 Simulation Owners Can Create Associated Assignments

Description:

Any user with "Owner" permissions of a simulation will be able to create or modify assignments that are associated with the simulation. Then, any user that the simulation is shared with (as a Forker or Viewer) will be able to see the assignment, but not edit it.

Steps:

1. Sign in
2. Create simulation and save it as private
3. Create associated assignment
4. Click share button at bottom of simulator window
5. Sharing with users as "Forkers" or "Viewers" will be able to complete assignment, but not edit it.
6. Click Share!

8.6 Simulation Owners Can View Any Simulations if Forked from Their Simulation

Description:

Simulation Owners who have shared with users as Forkers may want to view changes made to the duplicated simulation. Any fork of a simulation will be viewable by the original owner, even if private.

Steps:

1. Sign in
2. Create simulation and save it as private
3. Click share button at bottom of simulator window
4. Enter username to share with
5. Click "Forker" to give that user forker permissions
6. Click "Share!"
7. View new instance of simulation shared with user as Forker in a list of Forked simulation under original simulation.

8.7 Sharing Your Own Simulation Privately with Someone as a Forker

Description:

A registered user might want to share their simulation with another user and allow that user to “finish from where was left off” without modifying the original simulation. The owner’s simulation will be copied in it’s current state and shared with the forker, who will then become the owner of the duplicate.

Steps:

1. Sign in
2. Create simulation and save it as private
3. Click share button at bottom of simulator window
4. Enter username to share with
5. Click “Forker” to give that user forker permissions
6. Click “Share!”

8.8 Sharing Your Own Simulation Privately with Someone as a Viewer

Description:

A registered user might want to share their simulation with a user but doesn’t want to allow that user to modify the simulation. The “viewer” user will be able to make changes and play with the shared simulation, but none of their changes will be saved and won’t modify the original simulation.

Steps:

1. Sign in
2. Create simulation and save it as private
3. Click share button at bottom of simulator window
4. Enter username to share with
5. Click “Viewer” to give that user viewer permissions
6. Click “Share!”

8.9 Create Walkthrough

Description:

A user will create walkthrough annotations for a saved simulation that can then be shared with other users.

Steps:

1. User will select a simulation to create walkthrough for.
2. Enter summary of walkthrough.
3. Navigate to desired key-point of the walkthrough.
4. Add annotation at key-point.
5. Repeat steps 3-4 for as many key-points as desired.

8.10 Rate a Simulation

Description:

A user who is logged in and viewing a simulation will rate the simulation out of 5 stars and provide an optional comment.

Steps:

1. While viewing a specific simulation user clicks ‘rate this sim’ located by the simulation’s 5 star rating under the lower right hand corner of simulation.
2. Pop-up window with rating dialogue appears.
3. A user selects rating out of 5 stars.
4. Under the star gui, user can enter optional text comment.
5. User selects ‘submit rating’.

8.11 Create User Group

Description:

A logged in user will create a user group with which to share simulations with.

Steps:

1. User selects ‘my account’ in top left corner.
2. User selects ‘my groups’.
3. In my groups users selects ‘add new group’.
4. User enters in group name.
5. User can search for users by Name or email.
6. Select which users to add to group.
7. Select ‘Create group’
8. New group is not listed under ‘my groups’.

8.12 Favorite a simulation

Description:

While viewing a simulation a logged-in user favorites a simulation to be listed under the user’s ‘my favorites’.

Steps:

1. User opens a simulation to view.
2. User clicks favorite ‘heart icon’ located under bottom right corner of simulation.
3. If heart icon is not filled in, it is not selected as a favorite. If it is filled in, it is selected as a favorite.
4. User can continuously click heart icon to mark or unmark current simulation as a favorite.

8.13 Comment on a Simulation

Description:

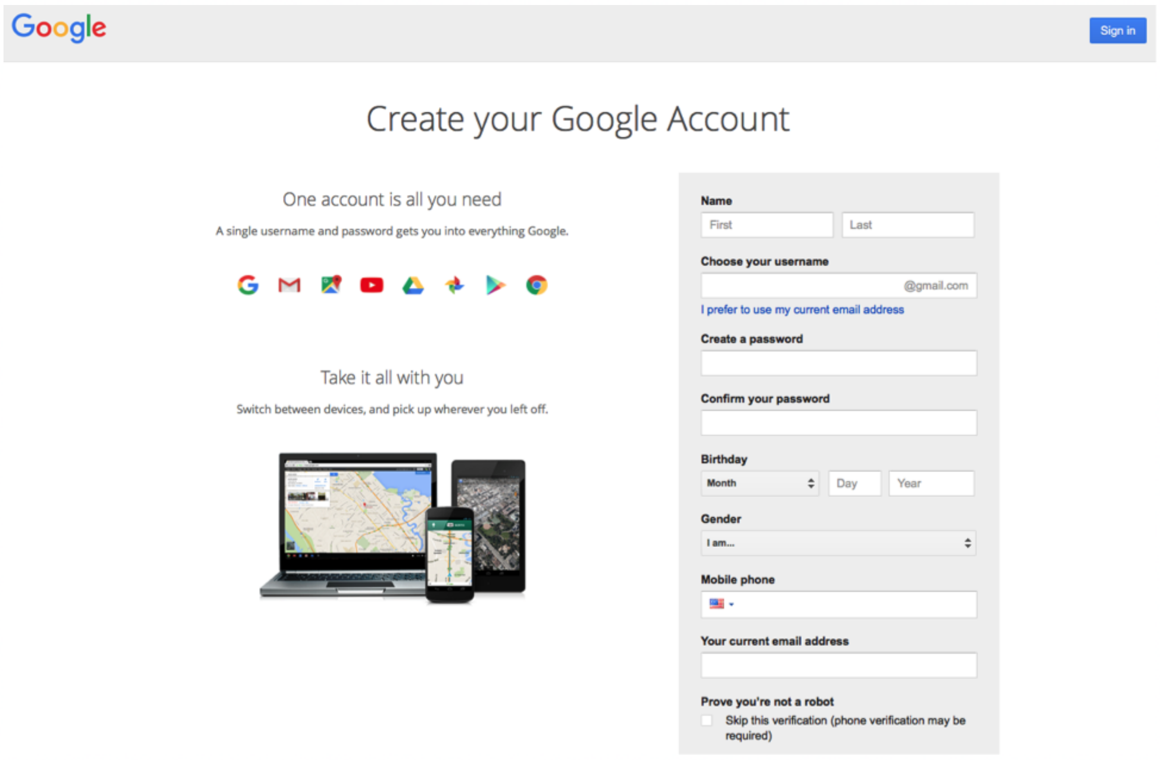
While viewing a simulation a logged-in user comments on the current simulation.

Steps:

1. User opens a simulation to view.
2. User navigates under simulation to ‘comments’ section
3. User enters their comment into text field.
4. Press Enter to submit comment.

Appendix: UI Sketches

9 Unregistered Users - UI Sketches



Google Sign in

Create your Google Account

One account is all you need

A single username and password gets you into everything Google.

Take it all with you

Switch between devices, and pick up wherever you left off.

Name

First | Last

Choose your username

@gmail.com

I prefer to use my current email address

Create a password

Confirm your password

Birthday

Month | Day | Year

Gender

I am...

Mobile phone

Your current email address

Prove you're not a robot

Skip this verification (phone verification may be required)

Figure 9: A.1: Registering a User

10 Registered and Unregistered Users - UI Sketches

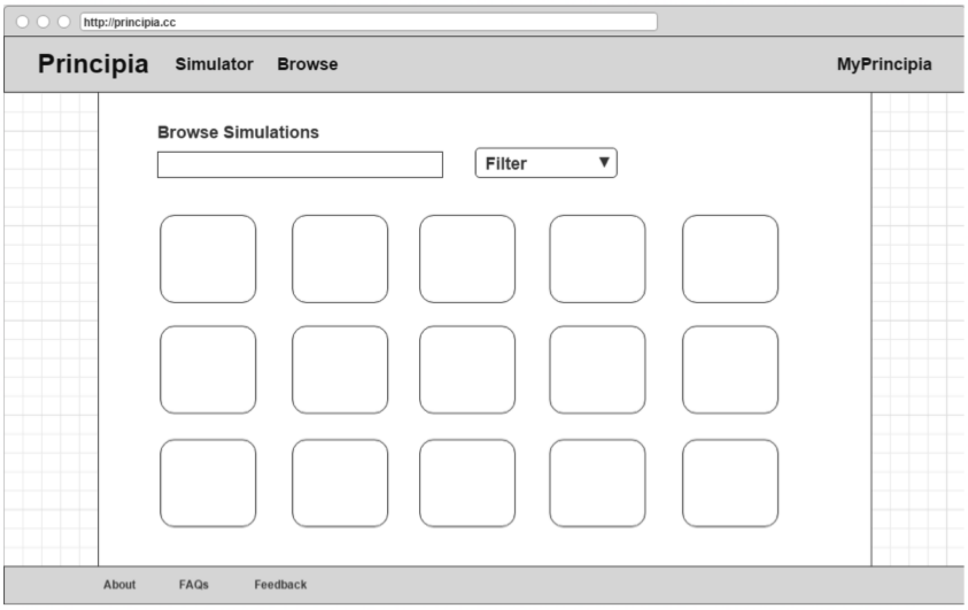


Figure 10: B.1: View Public Simulations

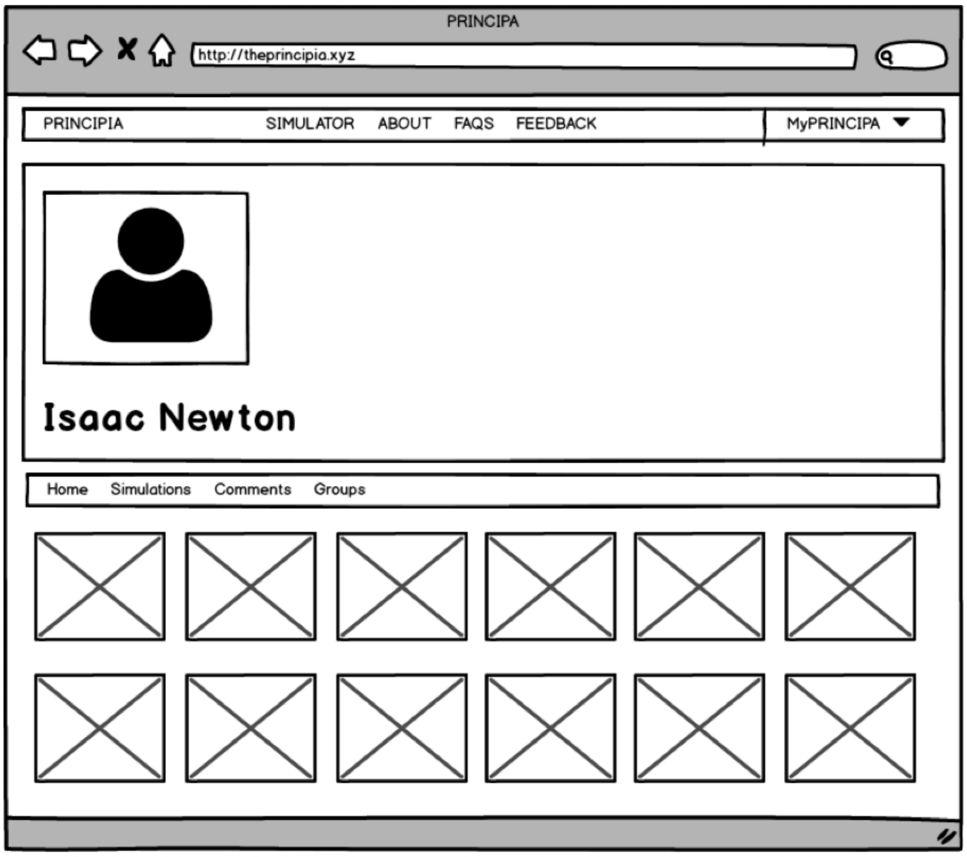
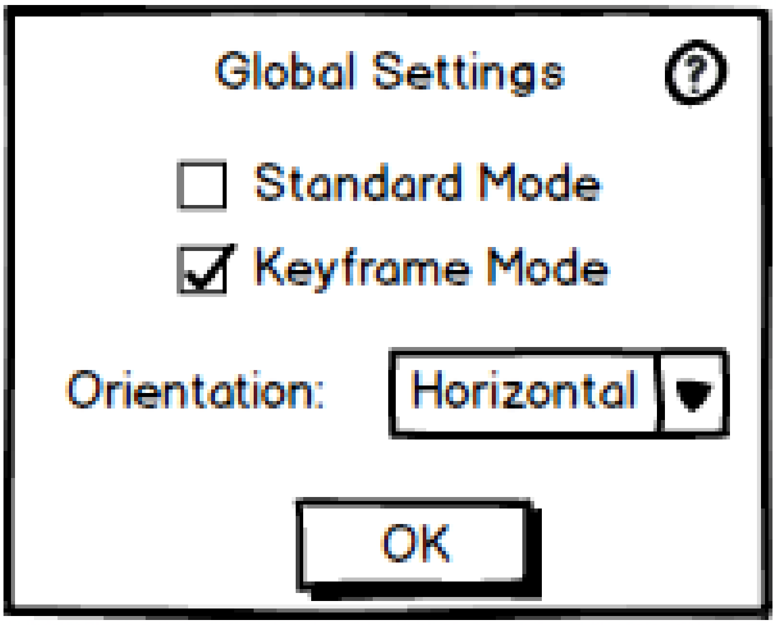
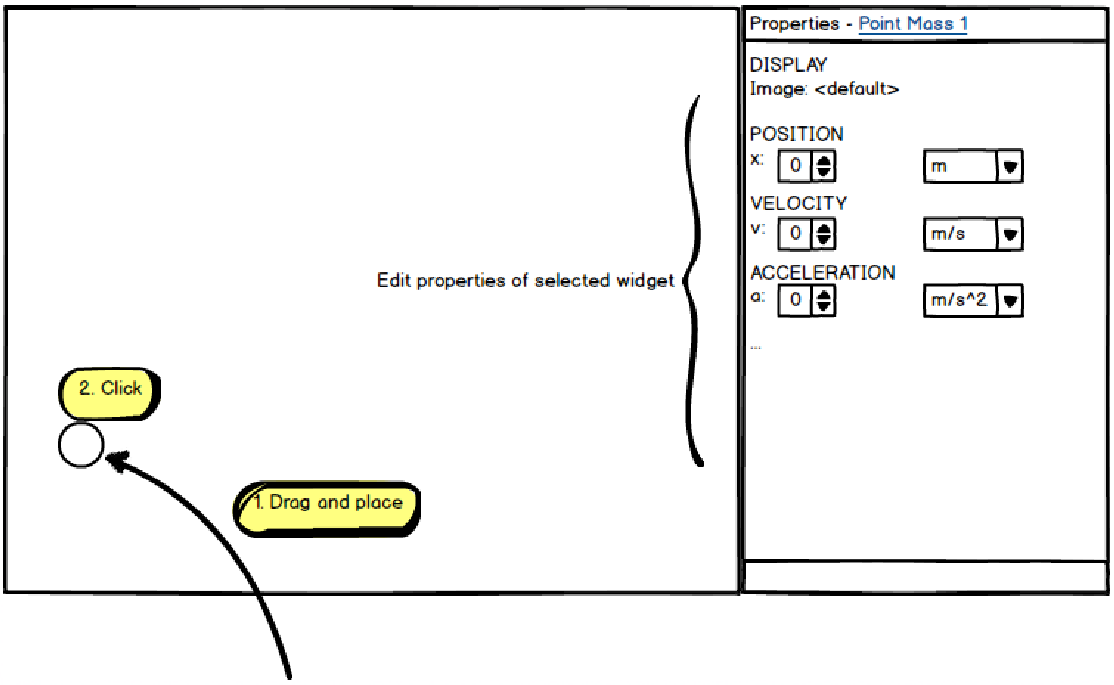


Figure 11: B.1: View Other User's Profile



☐ Standard Mode
☑ Keyframe Mode
Orientation: Horizontal

Figure 12: B.*: Global Settings Dialog



Properties - Point Mass 1
DISPLAY
Image: <default>
POSITION
x: 0 m
VELOCITY
v: 0 m/s
ACCELERATION
a: 0 m/s^2
...

Figure 13: B.3: 1D Kinematics

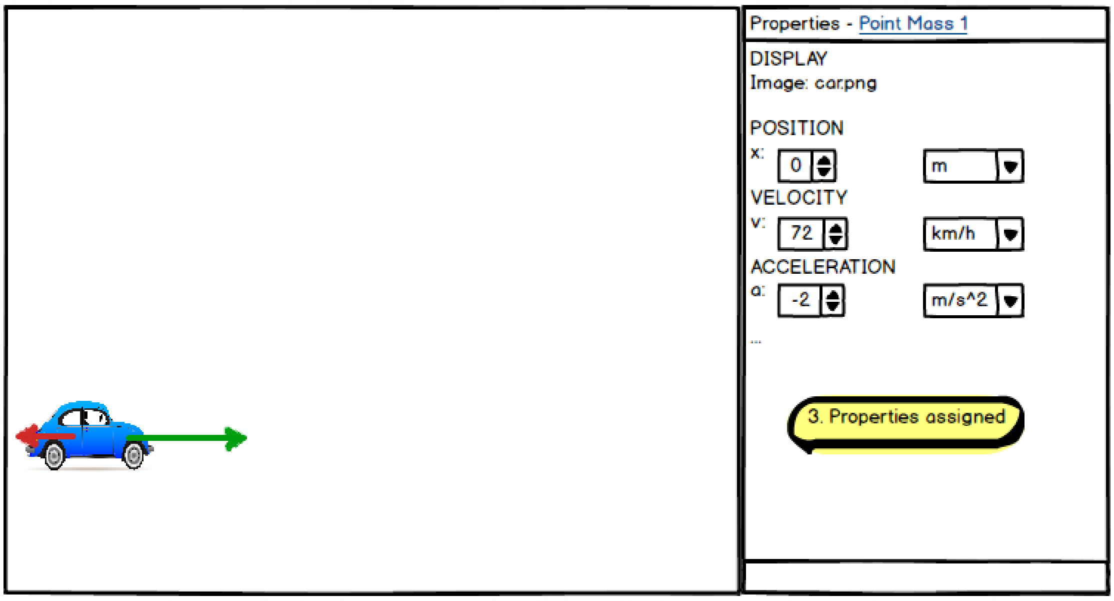


Figure 14: B.3: 1D Kinematics

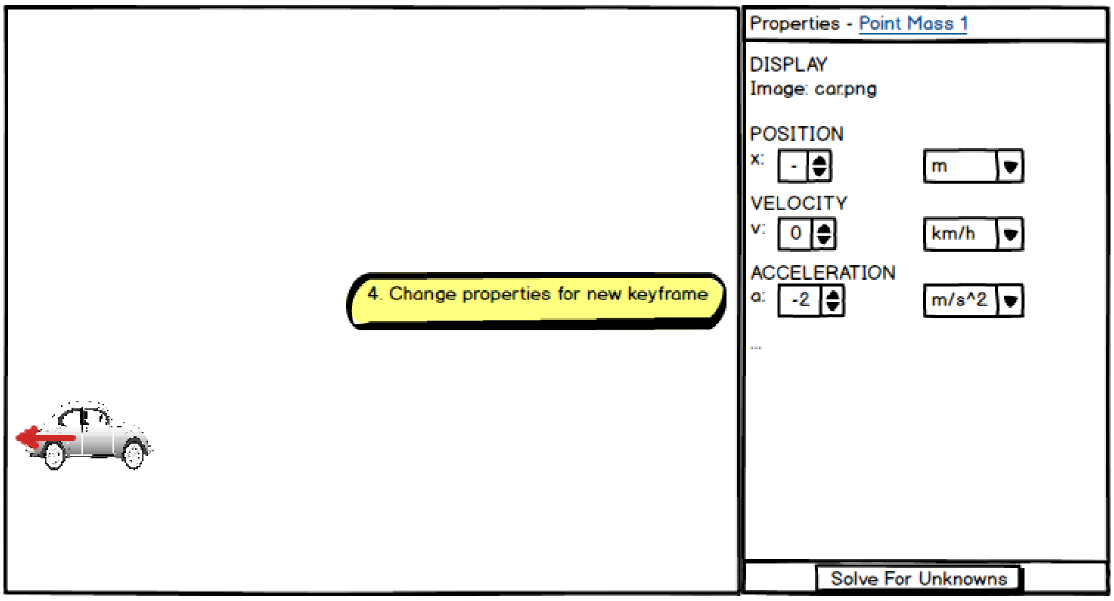


Figure 15: B.3: 1D Kinematics

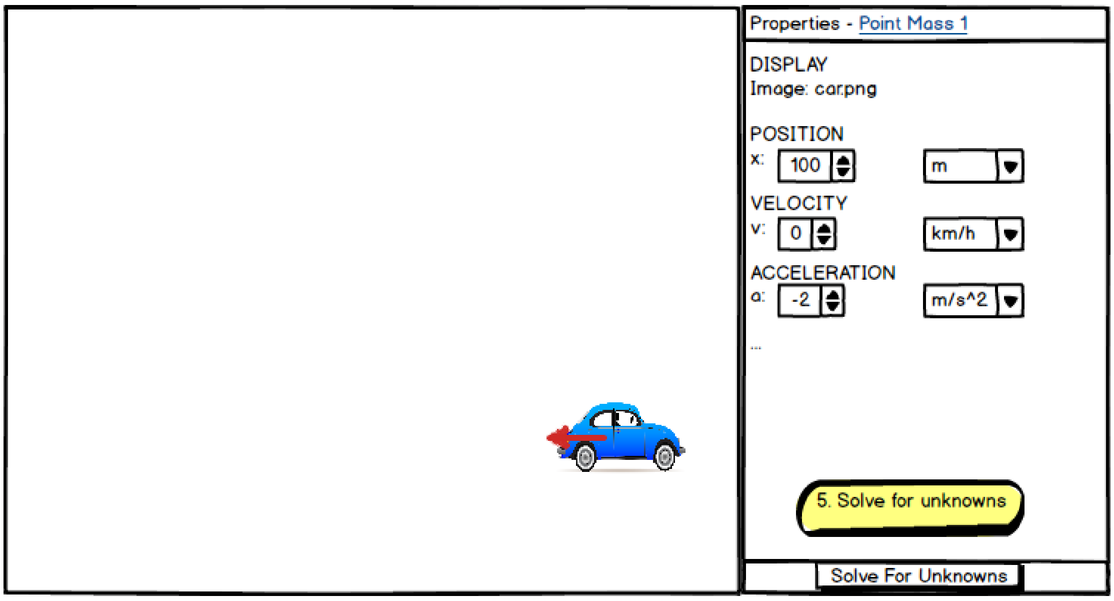


Figure 16: B.3: 1D Kinematics

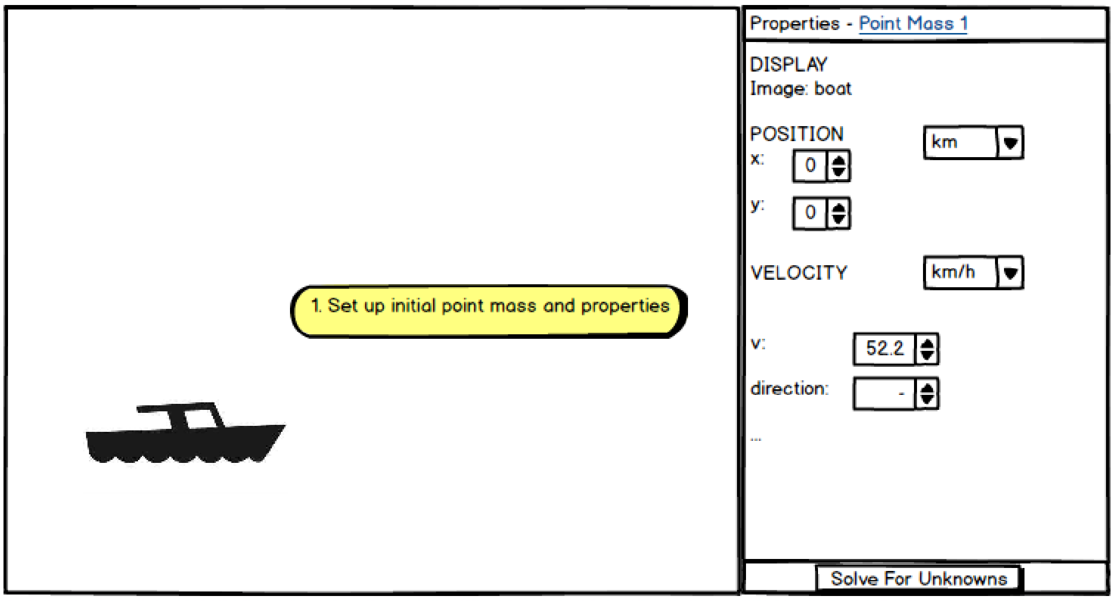


Figure 17: B.4: 2D Kinematics

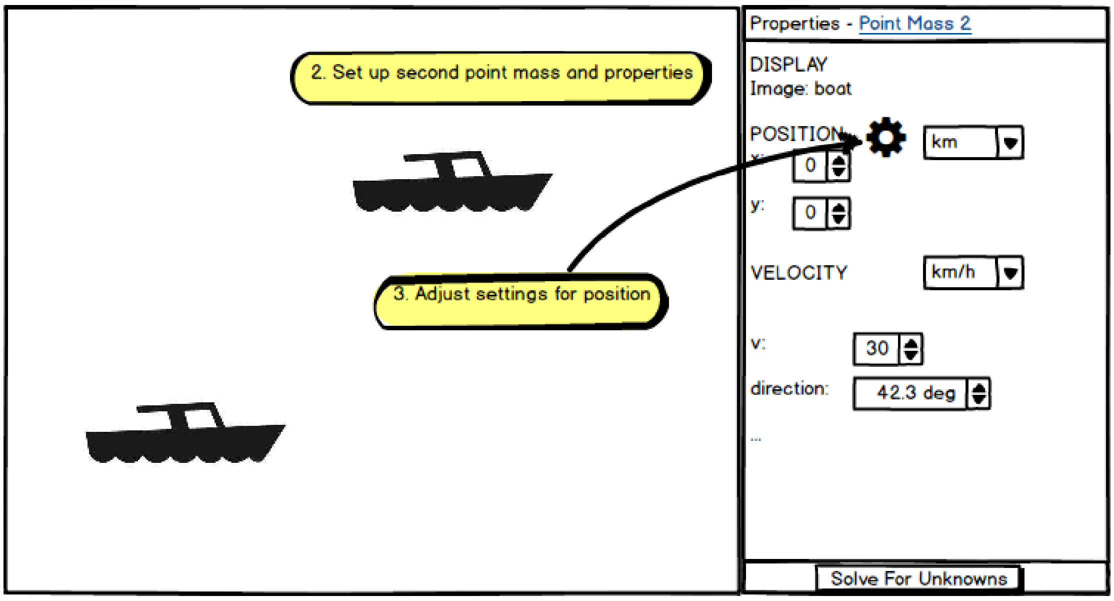

2. Set up second point mass and properties
3. Adjust settings for position
Properties - Point Mass 2
DISPLAY
Image: boat
POSITION
km
x: 0
y: 0
VELOCITY
km/h
v: 30
direction: 42.3 deg
...
Solve For Unknowns

Figure 18: B.4: 2D Kinematics

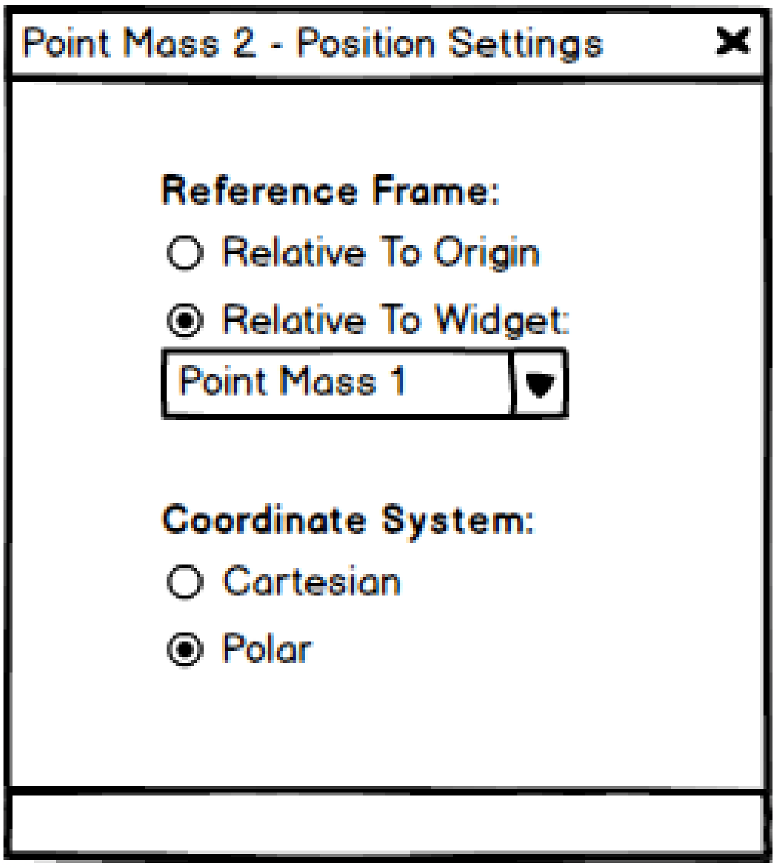
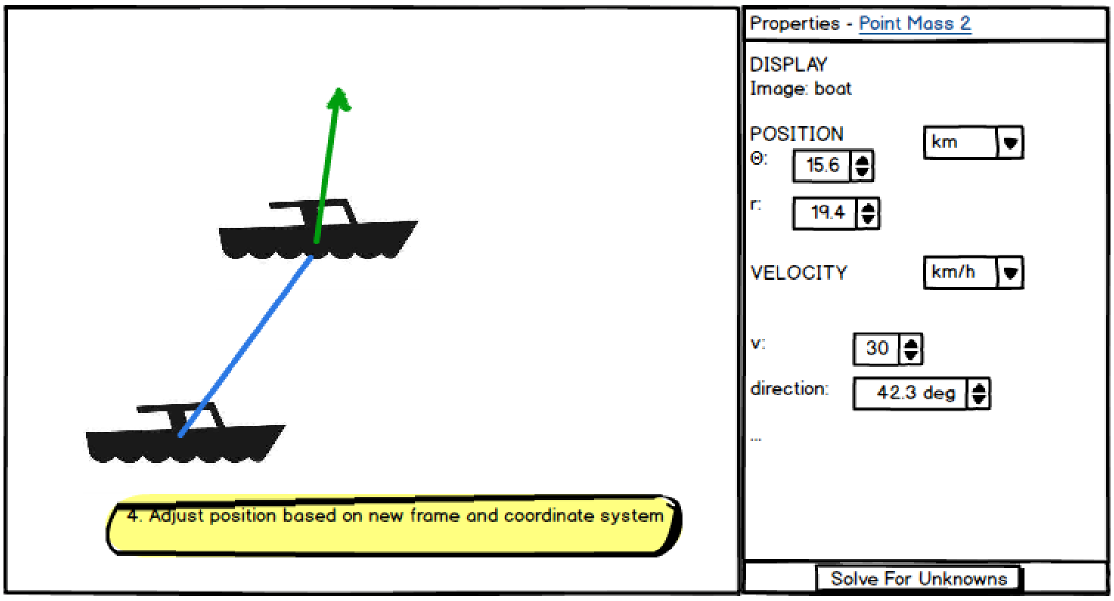


Figure 19: B.4: 2D Kinematics



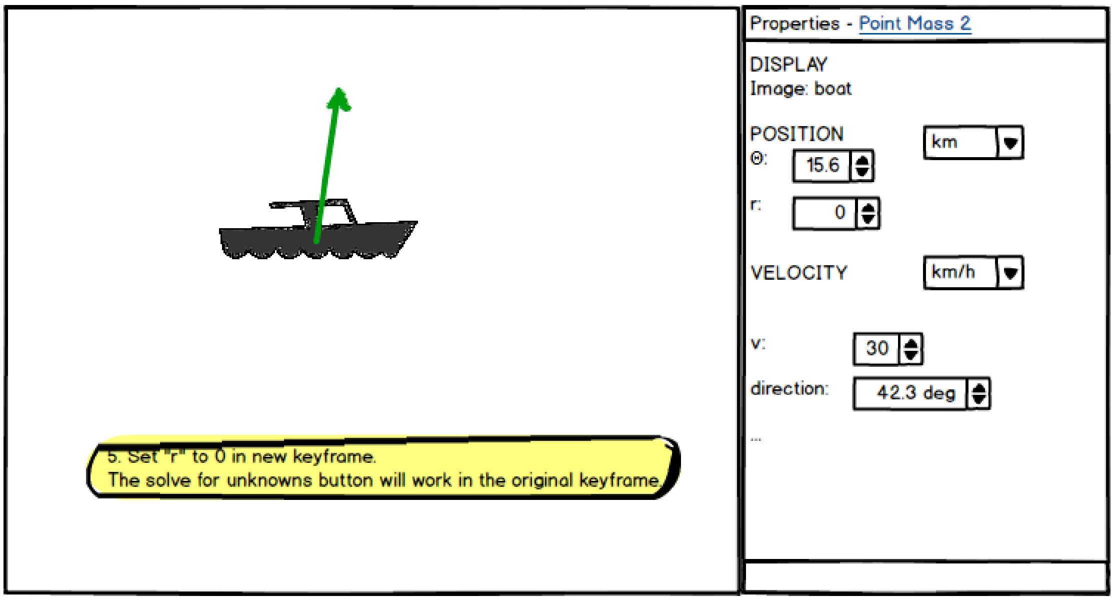
Properties - Point Mass 2

DISPLAY
Image: boat

POSITION km
Θ: 15.6
r: 19.4

VELOCITY km/h
v: 30
direction: 42.3 deg
...

Figure 20: B.4: 2D Kinematics


5. Set "r" to 0 in new keyframe.
The solve for unknowns button will work in the original keyframe.

Properties - Point Mass 2

DISPLAY
Image: boat

POSITION km
Θ: 15.6
r: 0

VELOCITY km/h
v: 30
direction: 42.3 deg
...

Figure 21: B.4: 2D Kinematics

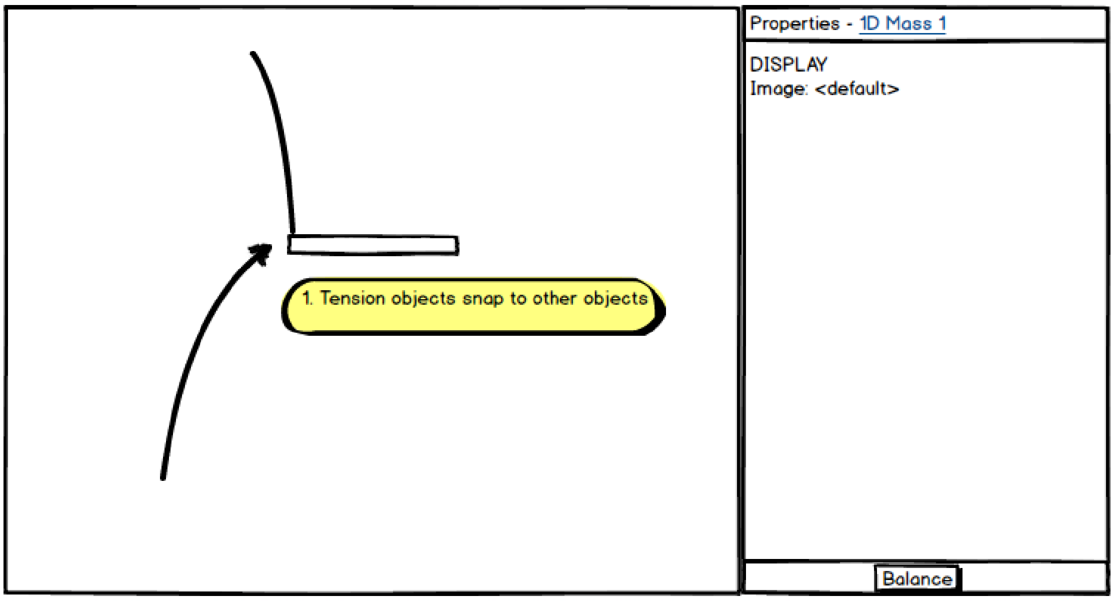


Figure 22: B.5: Statics

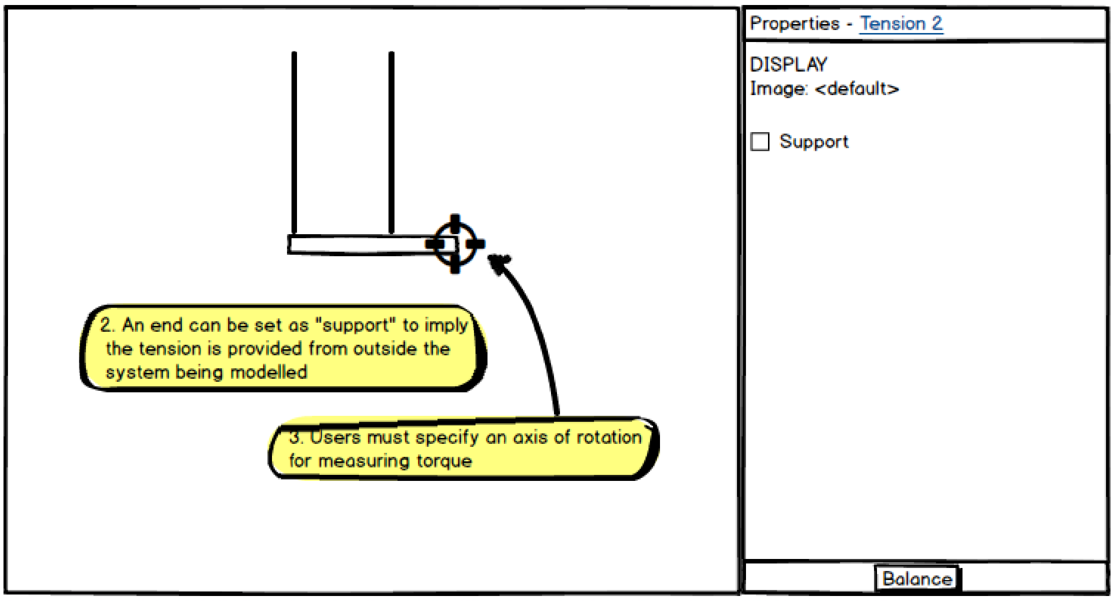

Properties - Tension 2
DISPLAY
Image: <default>
Support
2. An end can be set as "support" to imply the tension is provided from outside the system being modelled
3. Users must specify an axis of rotation for measuring torque
Balance

Figure 23: B.5: Statics

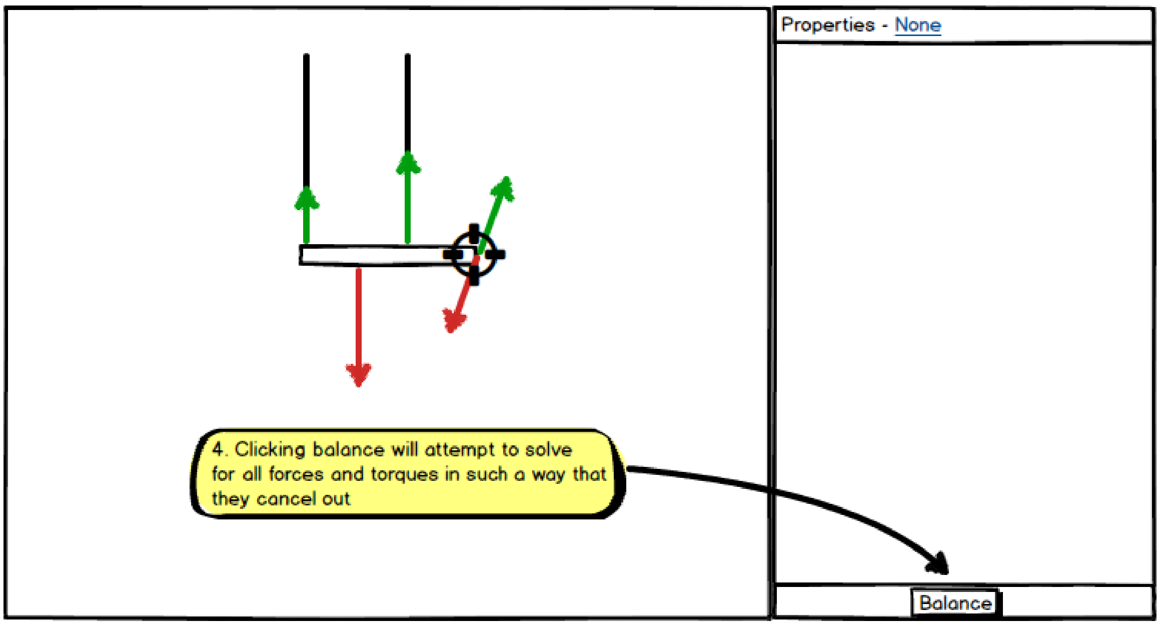
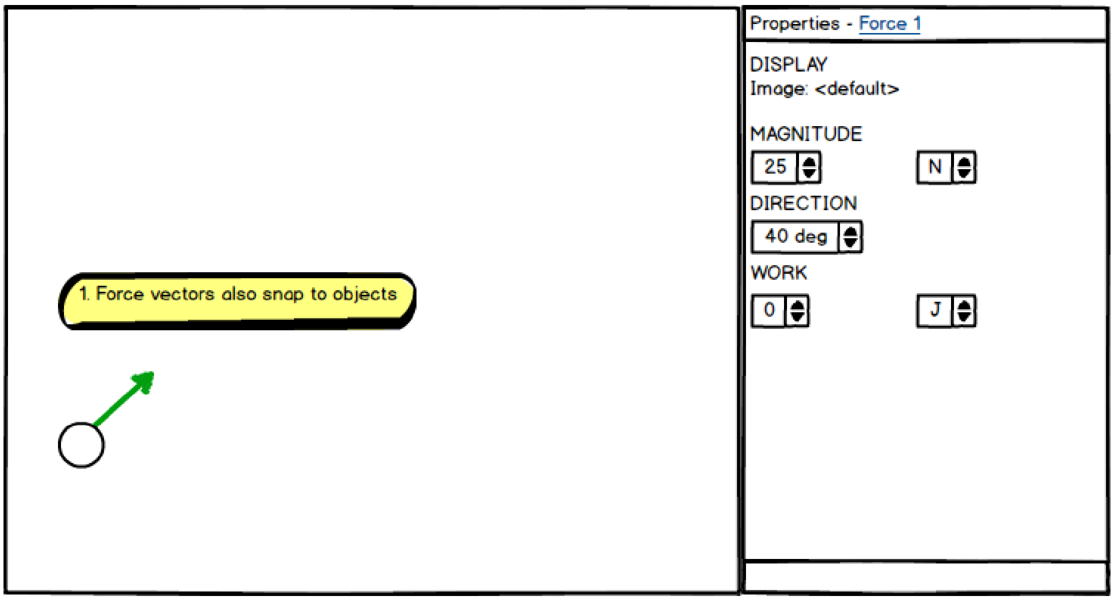
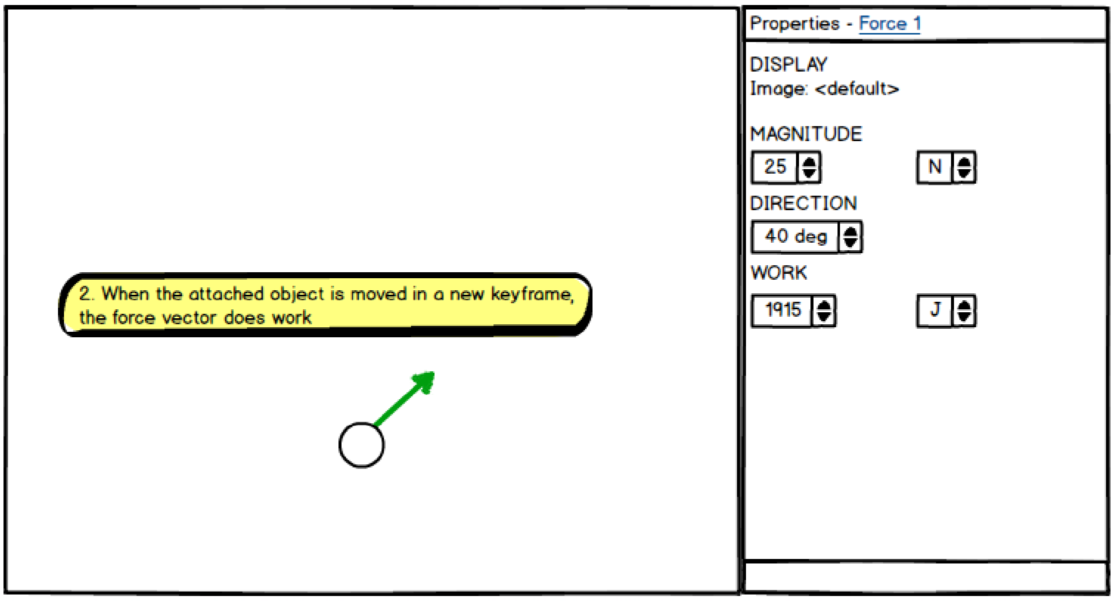

Properties - None
4. Clicking balance will attempt to solve for all forces and torques in such a way that they cancel out
Balance

Figure 24: B.5: Statics



DISPLAY
Image: <default>
MAGNITUDE
25 N
DIRECTION
40 deg
WORK
0 J

Figure 25: B.6: Work and Energy



2. When the attached object is moved in a new keyframe, the force vector does work

Properties - Force 1

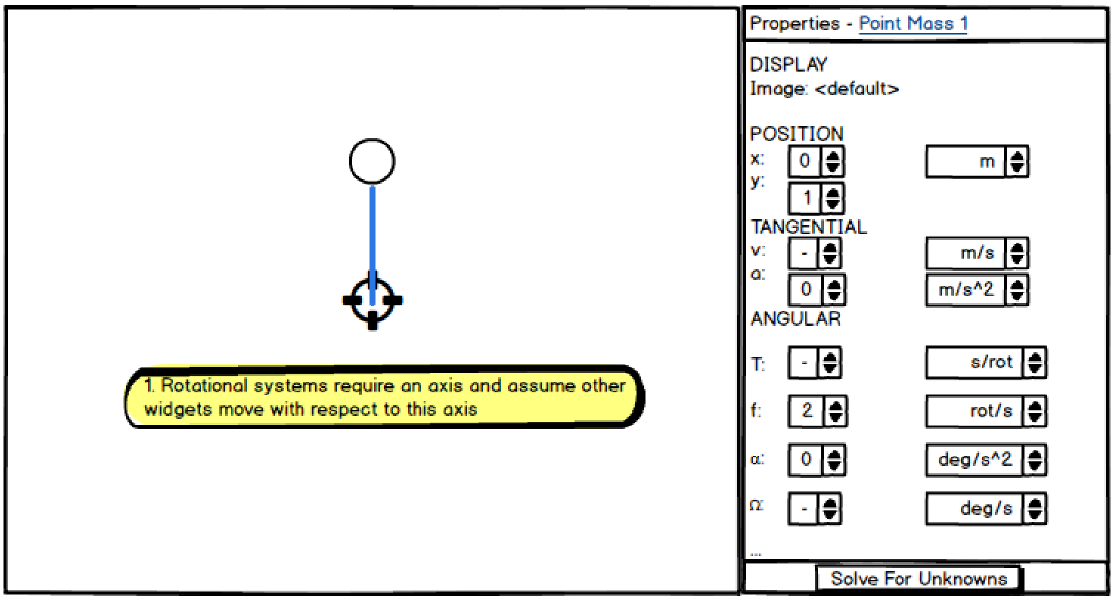
DISPLAY
Image: <default>

MAGNITUDE
25 N

DIRECTION
40 deg

WORK
1915 J

Figure 26: B.6: Work and Energy



1. Rotational systems require an axis and assume other widgets move with respect to this axis

Properties - Point Mass 1

DISPLAY
Image: <default>

POSITION
x: 0 m
y: 1

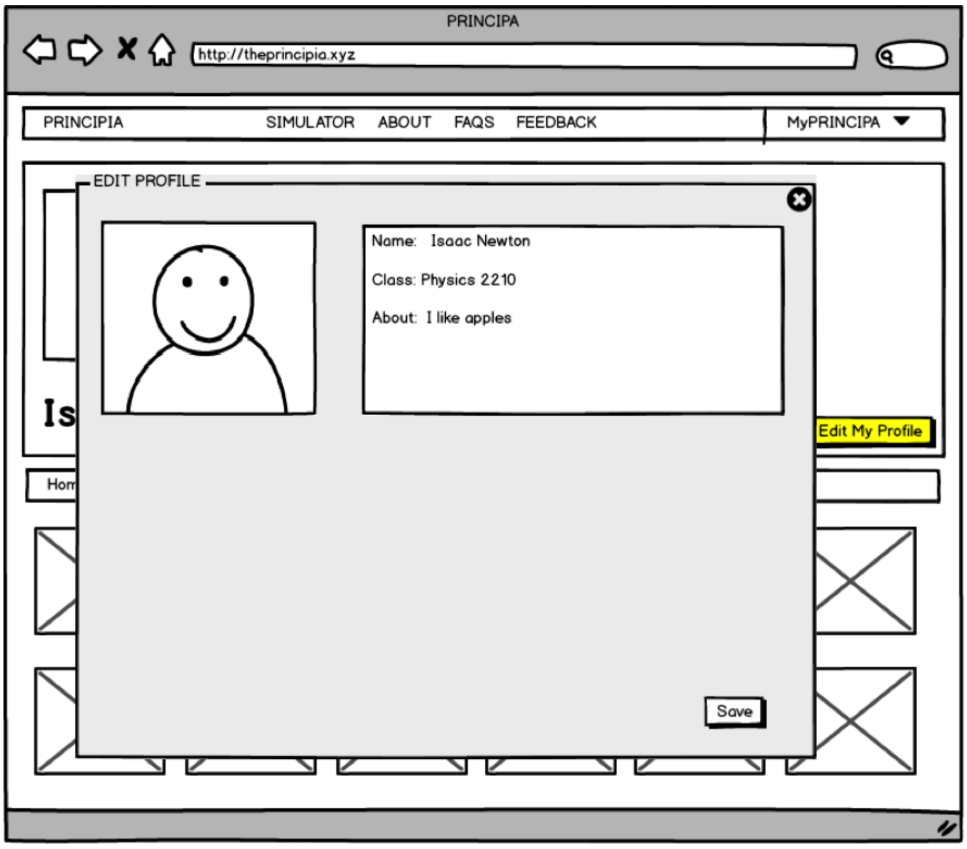
TANGENTIAL
v: - m/s
a: 0 m/s^2

ANGULAR
T: - s/rot
f: 2 rot/s
α: 0 deg/s^2
Ω: - deg/s
...

Solve For Unknowns

Figure 27: B.7: Angular Motion

11 Registered Users - UI Sketches

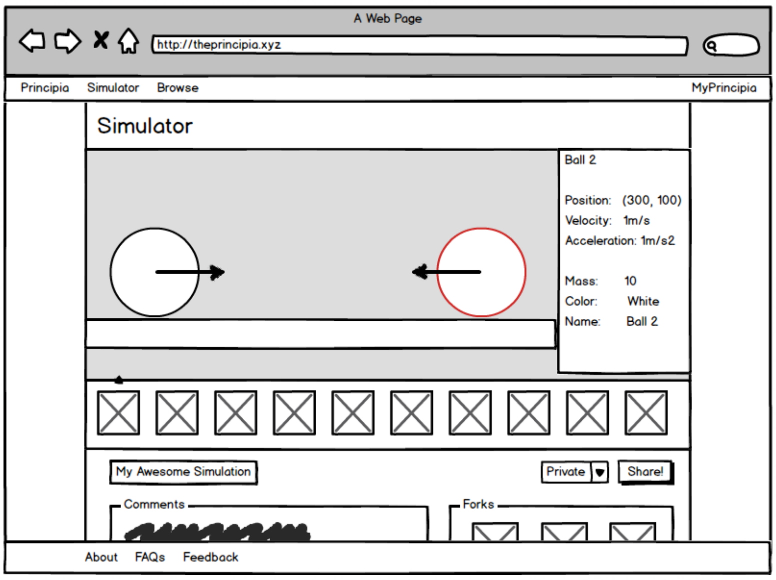


Name: Isaac Newton

Class: Physics 2210

About: I like apples

Figure 28: C.1 Create Profile Page

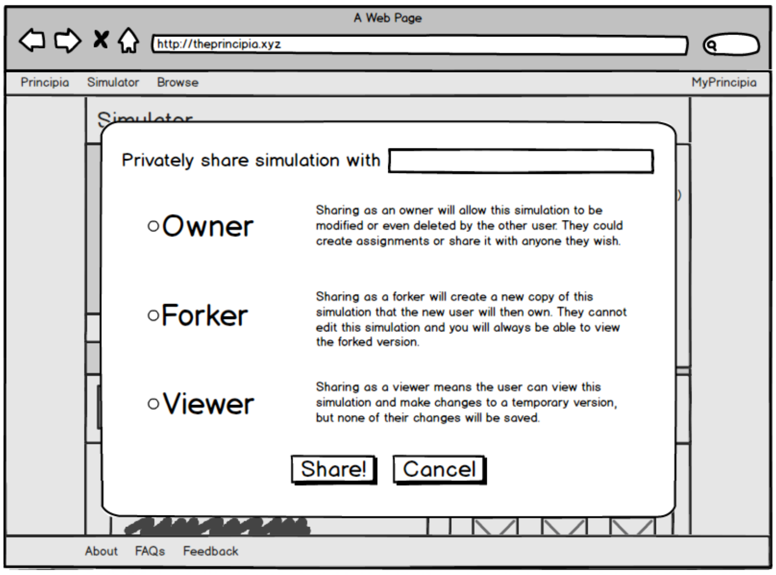


Ball 2

Position: (300, 100)
Velocity: 1m/s
Acceleration: 1m/s2

Mass: 10
Color: White
Name: Ball 2

Figure 29: C.3 & C.6: Share an existing fork



Privately share simulation with

○Owner

Sharing as an owner will allow this simulation to be modified or even deleted by the other user. They could create assignments or share it with anyone they wish.

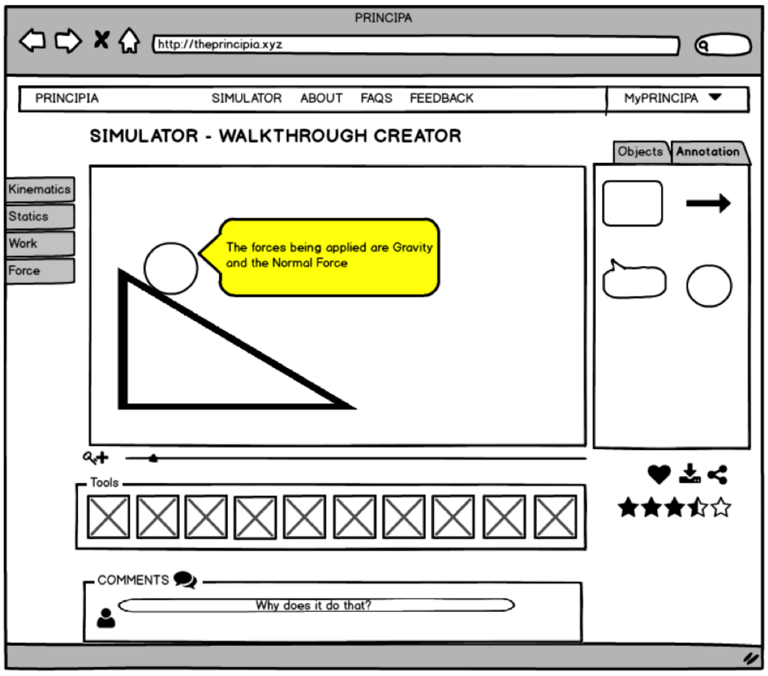
○Forker

Sharing as a forker will create a new copy of this simulation that the new user will then own. They cannot edit this simulation and you will always be able to view the forked version.

○Viewer

Sharing as a viewer means the user can view this simulation and make changes to a temporary version, but none of their changes will be saved.

Figure 30: C.4 & C.7 & C.8: Sharing permissions



The forces being applied are Gravity and the Normal Force

Why does it do that?

Figure 31: C.9: Create Walkthrough

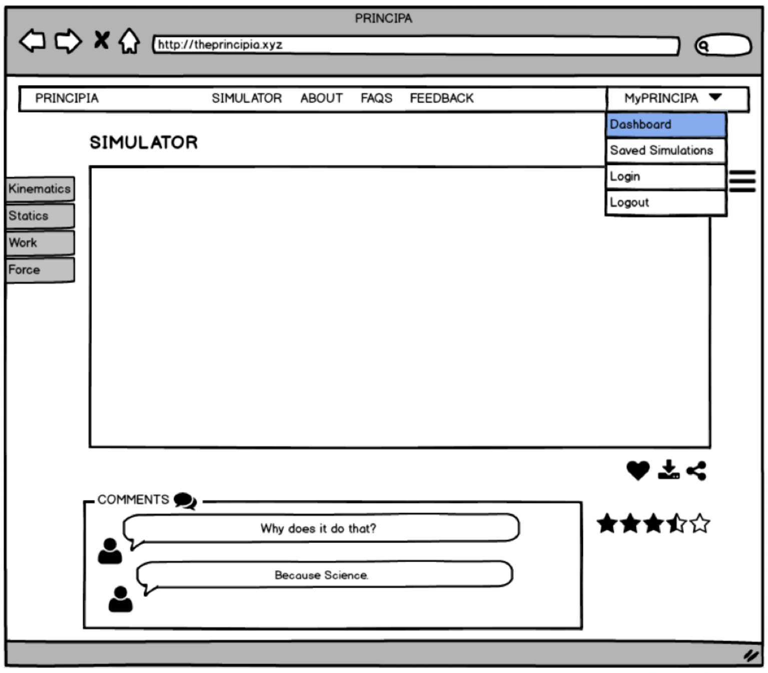


Figure 32: C.10: Rate a Simulation

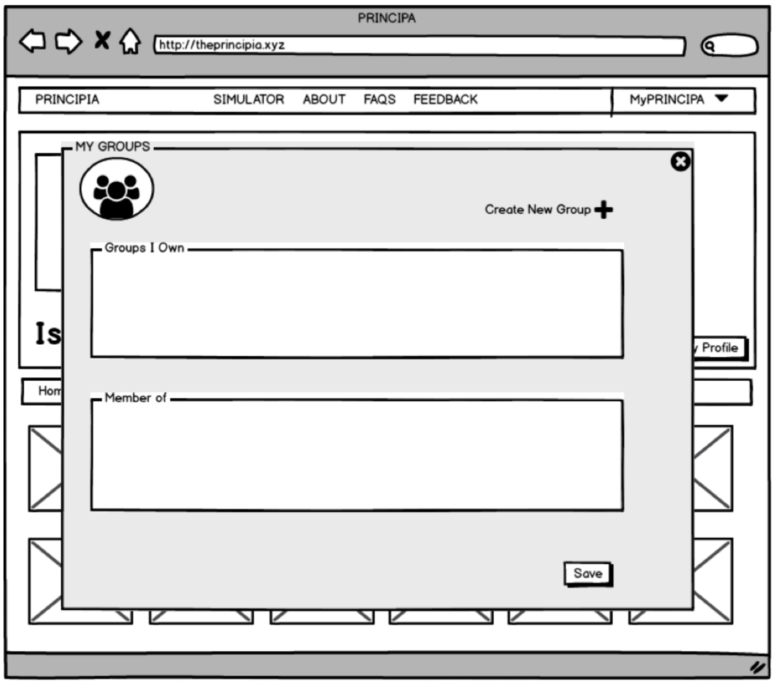


Figure 33: C.11: Create User Group

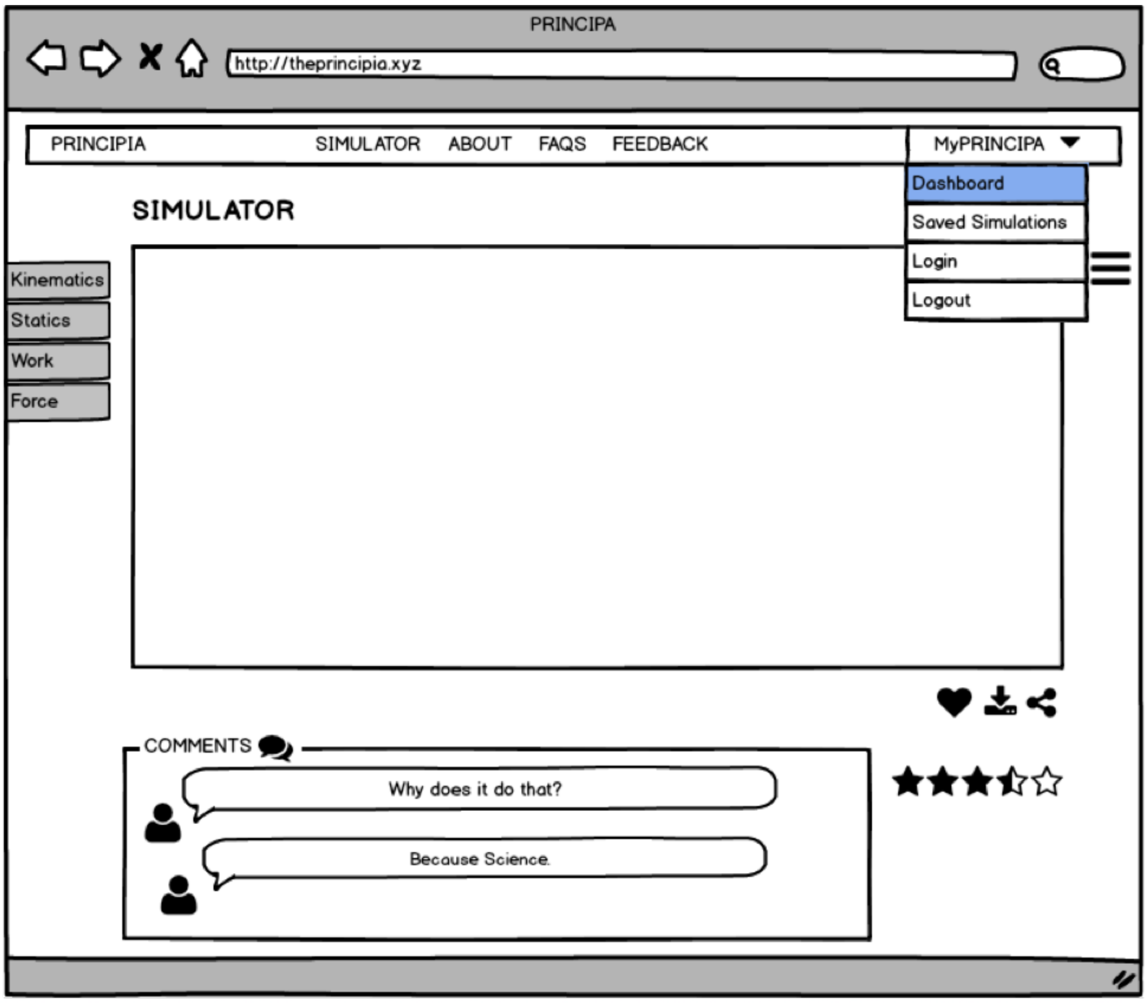


Figure 34: C.12 & C.13: Favorite/Comment on a Simulation

References

[1] Sam Bowen. American Physical Society. TIMSS - An Analysis of the International High School Physics Test. http://www.aps.org, August 1998. Accessed 2015-10-12.

[2] John W. Brelsford. Rice University. Physics Education in a Virtual Environment. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 37:1286–1290, October 1993.

[3] Stewart Software. Purchase Creative Physics ®. https://www.creativephysics.net. Accessed 2015-10-12.

[4] Gary Gladding, Mats Selen, and Tim Stelzer. smartPhysics: Checkpoint sample. https://www.smartphysics.com, January 2015. Accessed 2015-10-12.

[5] Kathy Perkins, Carl Wiemann, et al. UC Boulder. PhET Projectile Motion. https://phet.colorado.edu, May 2011. Accessed 2015-10-12.

[6] Cengage Learning. Discover: Enhanced WebAssign. http://sites.cengage.com, August 2014. Accessed 2015-10-12.

[7] Principia Repository: https://github.com/samolds/principia.

[8] Mastering Issues: https://guides.github.com/features/issues/.

[9] Michael Mathews. JSDoc 3: https://github.com/jsdoc3/jsdoc.