Balloons and Static Electricity

HTML5 Version

Team
Noah Podolefsky (lead designer)
Ariel Paul
Emily Moore
Katherine Perkins
Trish Loeblein
Sam Reid (MLL liaison)
Mobile Learner Labs (developers)

Original Sim

Balloons and Static Electricity was an early PhET sim. It’s main learning goal is for students to “Describe and draw models for common static electricity concepts (transfer of charge, induction, attraction, repulsion, and grounding)”.

The present Java-based version can be found here:

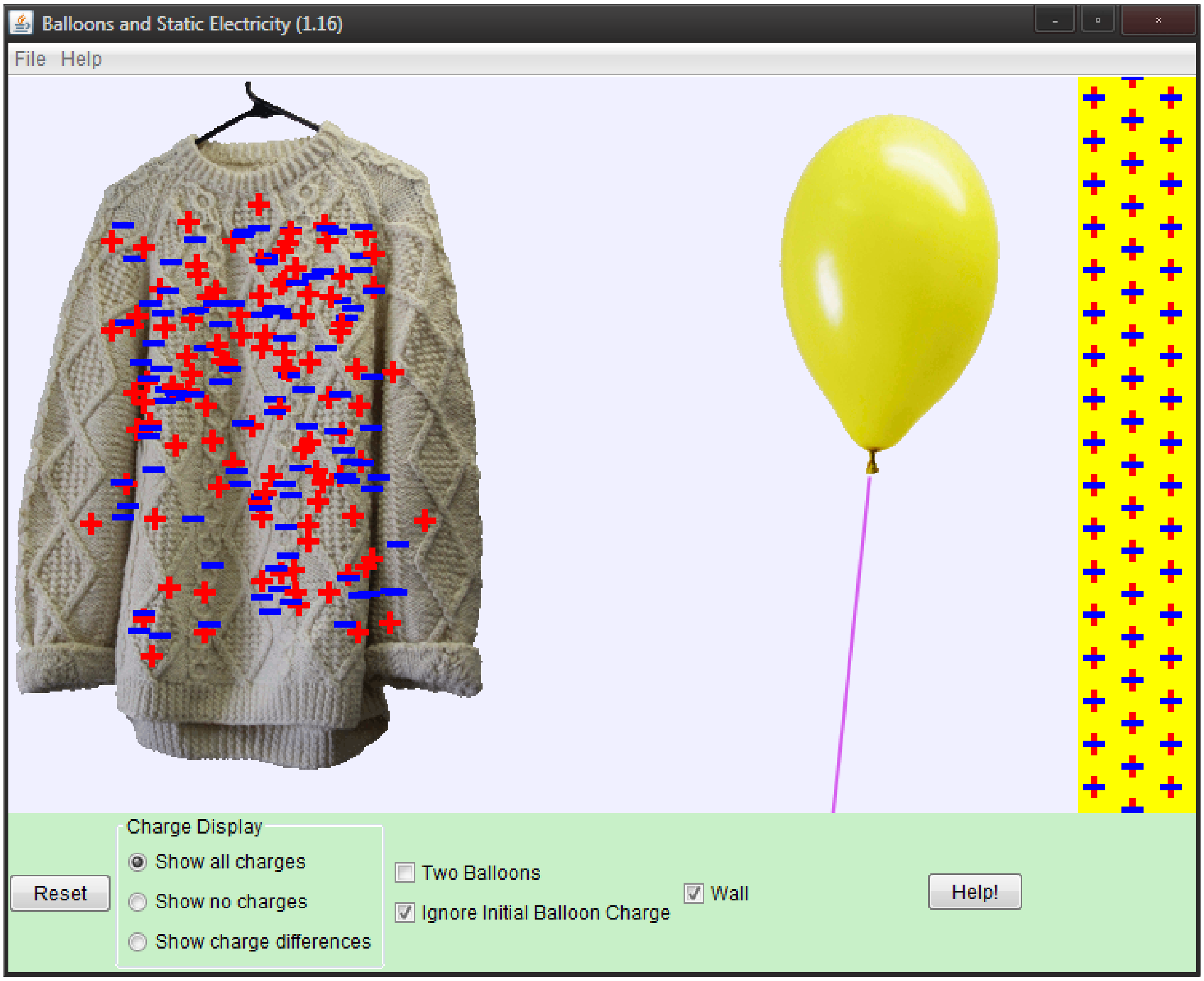
http://phet.colorado.edu/en/simulation/balloons

The new HTML5 version will be very similar in functionality. It will have a number of cosmetic design updates to match the look and feel of PhET sims that has evolved over the last several years.

The sim begins with a sweater, a balloon, and a wall (yellow box on right). Rubbing the balloon on the sweater causes the balloon to pick up negative charges. The balloon will then be attracted to the (now positively charged) sweater, or can stick to the wall due to an induced surface charge.

Users can:

- Move the balloon
- Activate “Two Balloons” which will interact with each other.
- Show all charges, no charges, or charge differences (i.e., net charge)
- Uncheck “Ignore Initial Balloon Charge”, which shows + and - charges on the balloon (initially equally balanced)
- Show or hide the wall
- Reset the sim to initial state
- Show “Help”, which is just two lines of text that say “Rub the balloon on the sweater” and “Bring the balloon near the wall”



Charge Display

● Show all charges

○ Show no charges

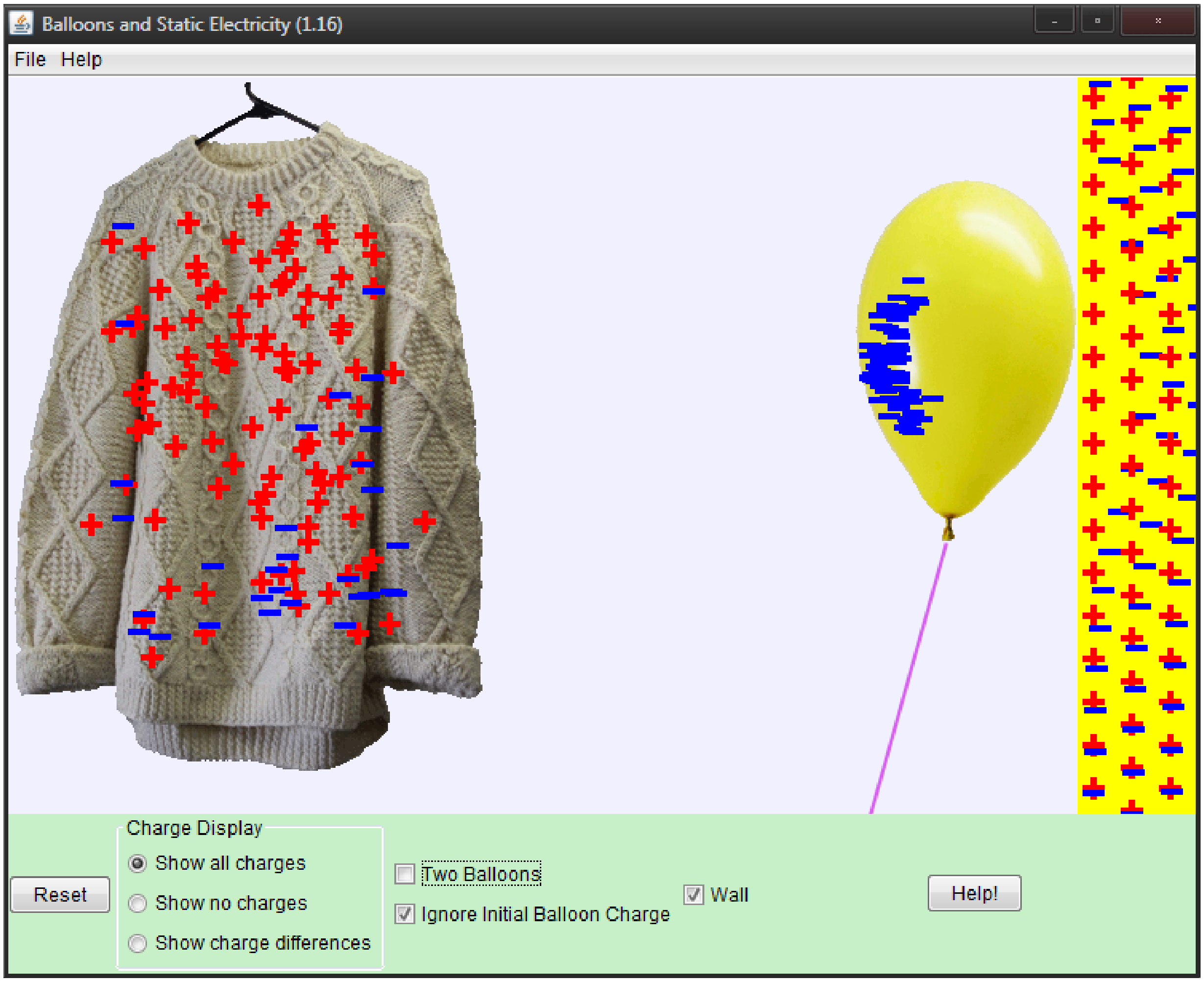
○ Show charge differences

□ Two Balloons

☑ Ignore Initial Balloon Charge

☑ Wall

The balloon picks up negative (-) charges from the sweater. When placed near the wall, the (-) charges of the balloon repel (-) in the wall, creating an induced surface charge. The balloon can then stick to the wall.



Charge Display

● Show all charges
○ Show no charges
○ Show charge differences

□ Two Balloons
☑ Ignore Initial Balloon Charge
☑ Wall

HTML5 Version

Fonts: Tahoma (all text)
Multitouch Support: None

Changes for HTML5 version:

- Updated for 1024x768 screen size
- Floating control panels and rearrange controls
  - Controls at bottom for touch
  - One/two balloon buttons
  - “Refresh” button replaces “Reset”
  - Added “Reset Balloon” button
  - Wall removed with “Remove Wall” button instead of checkbox
  - Removed “Ignore Initial Balloon Charge” button
- New wall appearance

- New sweater image
- New transparent balloon image
- Charges shown as red / blue spheres with + / - on them
- Charges distributed in grid (not randomized)
- (-) charges displaced to lower right of (+) charges
- Balloon string tied to point above balloon buttons
- Removed "Help" button
- Add “PhET” menu / title bar
- All controls should grow and shrink isometrically as the window size changes, but (if necessary) use a lower threshold beyond which the buttons no longer shrink so the buttons don’t become too small on the Nexus 7.

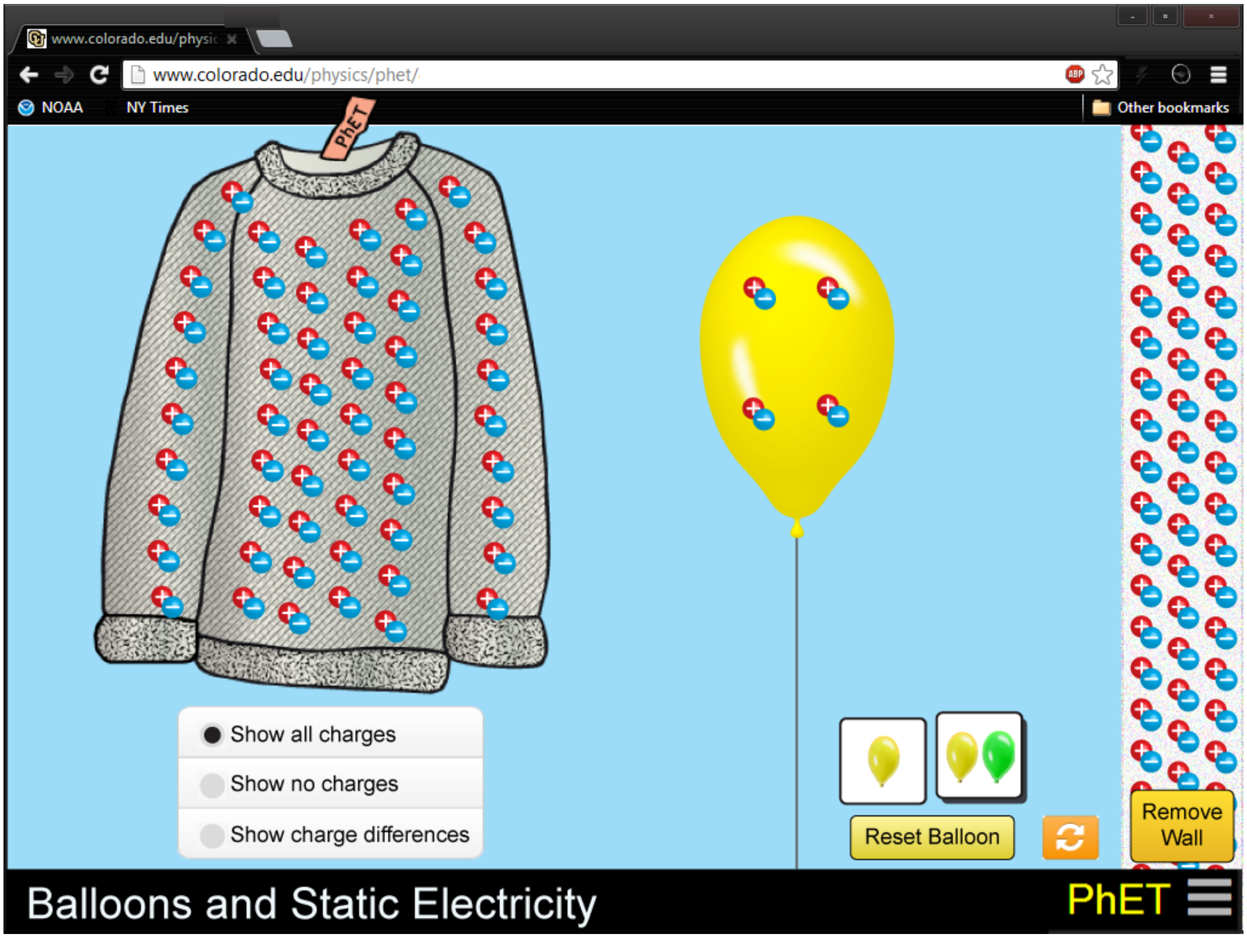
Text translation handling:

- Show charges radio button labels: expand button to max 200% of original width, then reduce text size to fit
- Reset Balloon button: expand button to max 150% of original width, then reduce text size to fit
- Remove Wall button: expand button to max 110% of original width, then reduce text size to fit. Can also expand button vertically to accommodate more than 2 lines of text.

Startup

- Balloon can be dragged and picks up (-) charges from the sweater as it sweeps past (see below)
- Sweater has fixed position (cannot be moved)
- Can choose one balloon or two with buttons below balloon
    - When the number of balloons is two the balloon reset button should have the text “Reset Balloons”, when there is only one balloon the button should say “Reset Balloon”
- Balloon string is tethered at bottom point
    - When there are two balloons, they should be tethered to the same point.
- “Reset Balloon” puts all charges and balloon back to original state (but does not change any other selections).
- Refresh (reset) puts entire sim back to initial state
- Can
    - “Show all charges” (as shown below)
    - “Show no charges” (hide all charges)
    - “Show charge differences” (show net charge on each object)
- Charged balloon will stick to wall (see below)
- Can remove wall with “Remove Wall” button, add back with “Add Wall” (see below)
- All charged objects interact
    - After picking up charges, balloon and sweater will be attracted
    - Two negatively charged balloons will repel
- Sweater charges are initially distributed in a grid as shown below

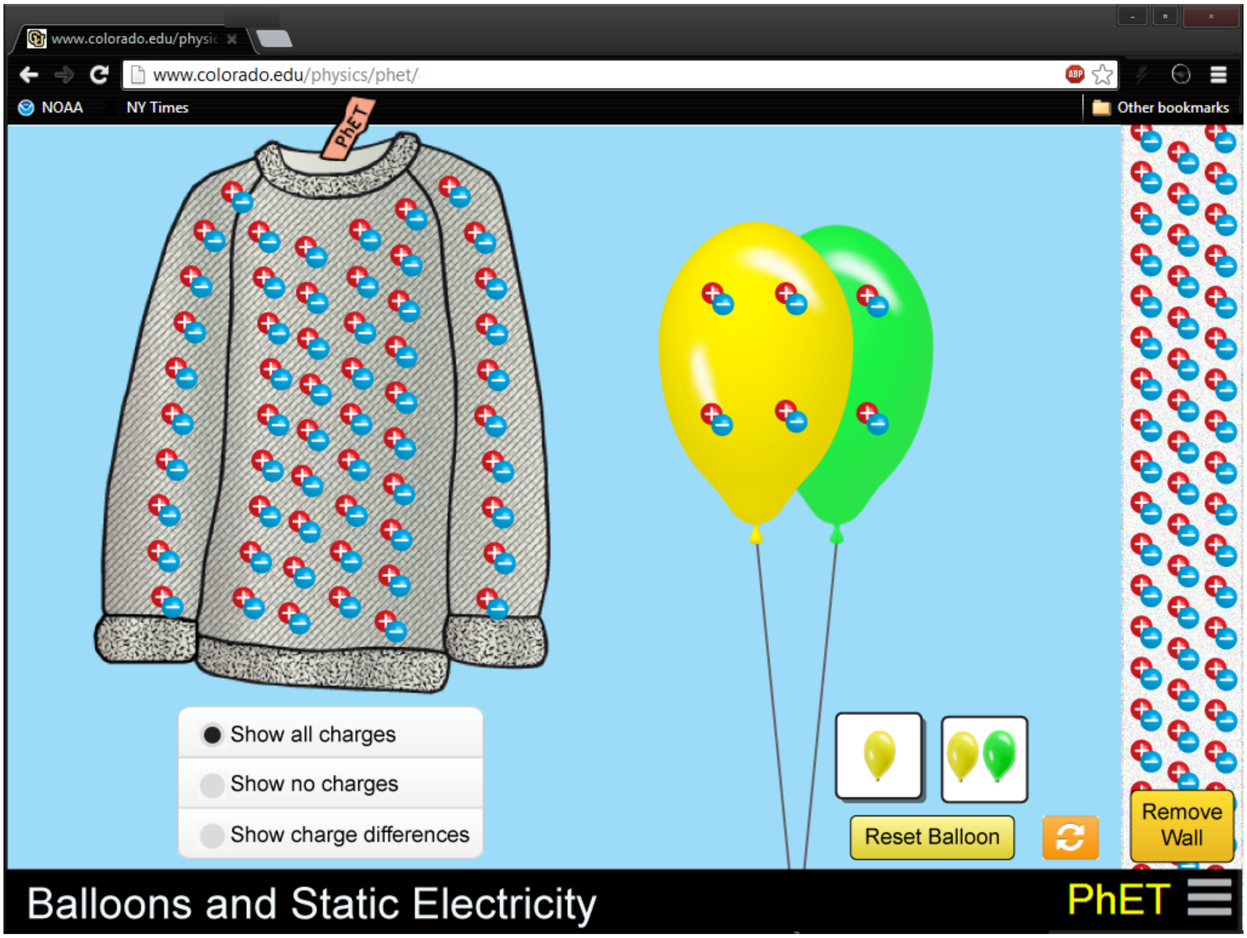
  - Grid lines curve to follow the contours of the sweater
  - +/- charge pairs have (-) on top, (-) displaced to lower right
  - + charges are fixed, never move
  - - charges get picked up by balloon
- The cursor should change into a hand when over any button, radio button or balloon.
- Unless otherwise specified in this document or further discussion, the behavior (motion) of the balloons and charges should roughly match that of the Java version.
- **Unlike in the drawings below**, the sweater graphic will be the one provided in the github repo. We will be providing a newer sweater, with a different texture but the same geometry.



● Show all charges
○ Show no charges
○ Show charge differences

Two balloons

- Sim appears as below when two balloons initially selected



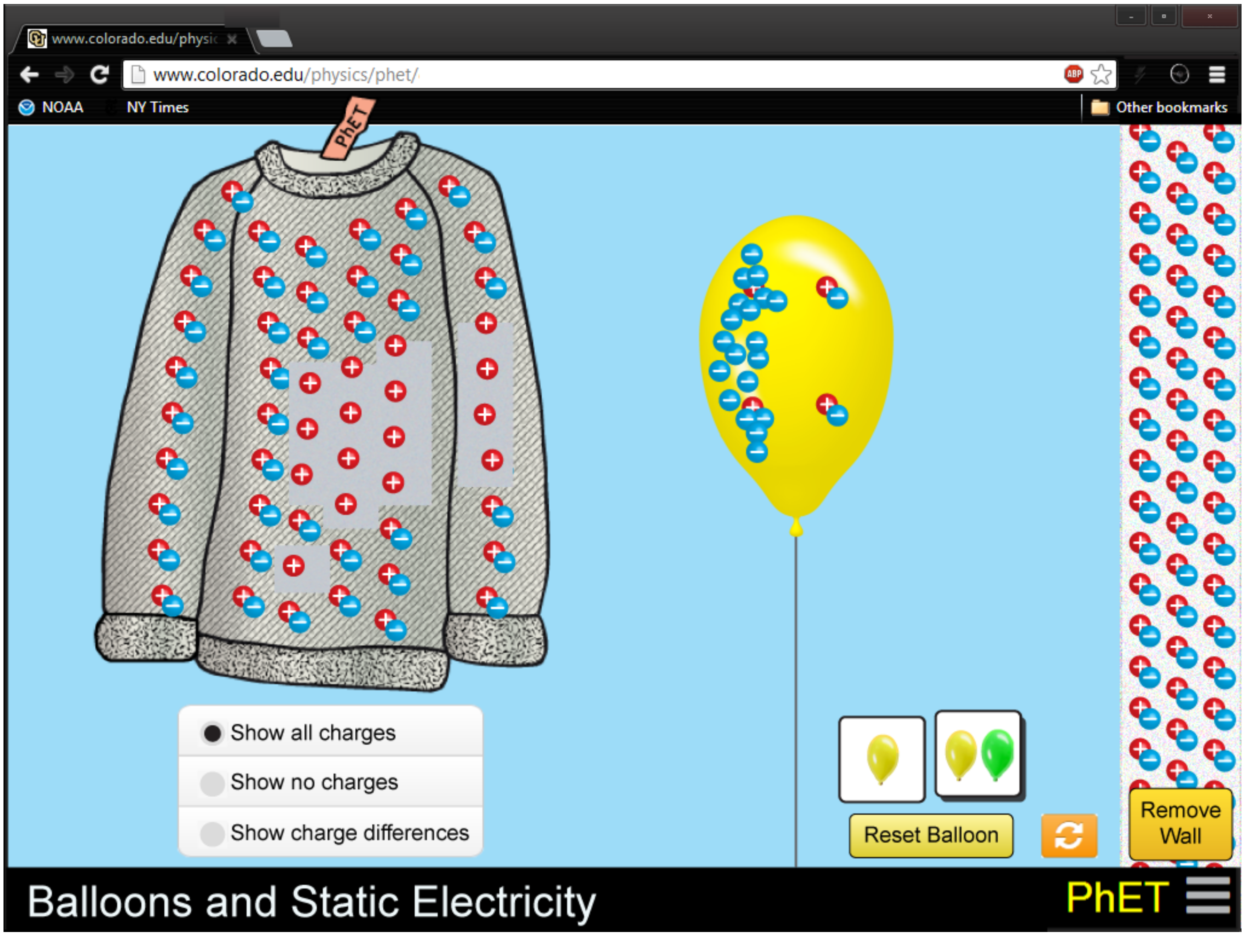
● Show all charges

○ Show no charges

○ Show charge differences

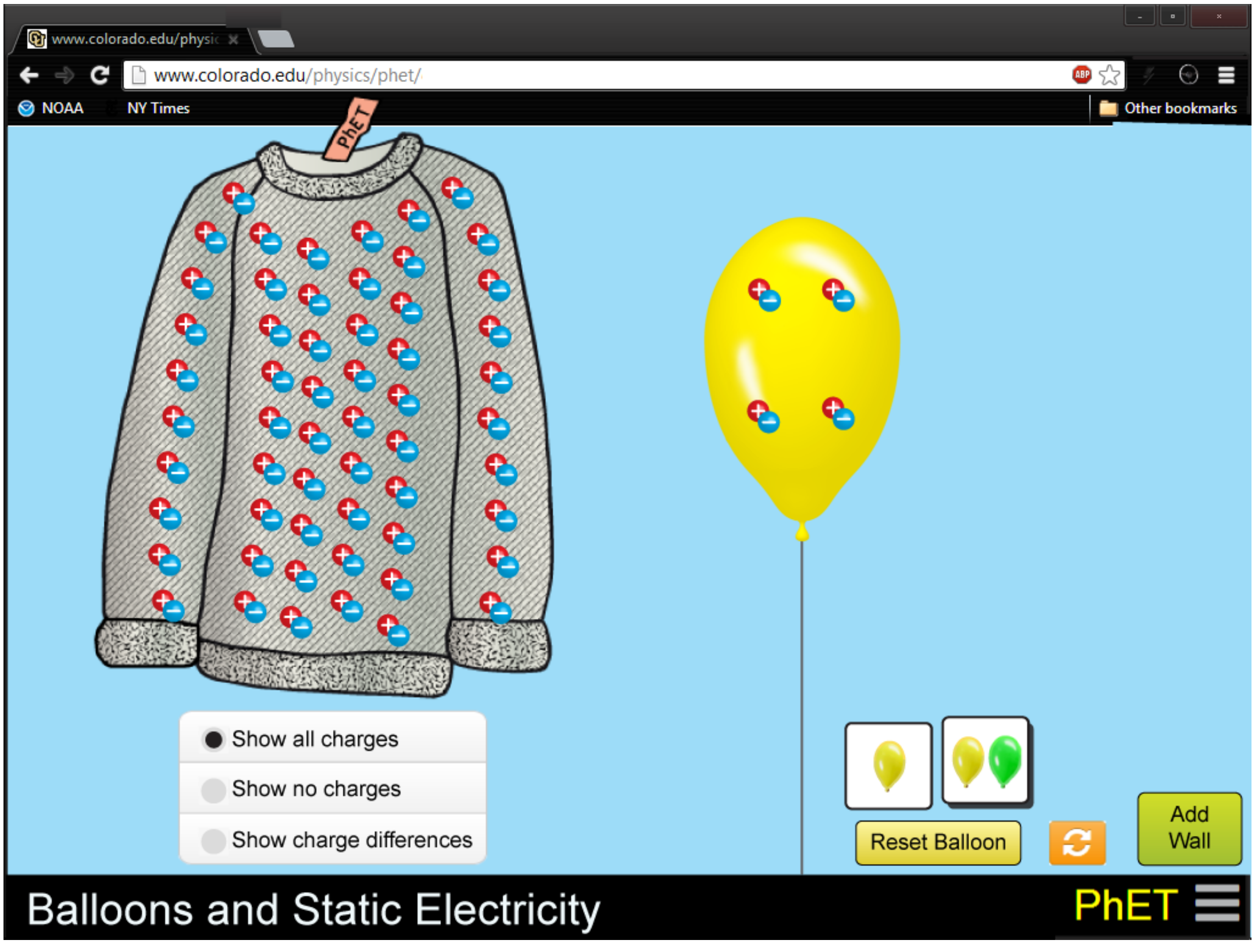
Charged balloon

- (-) charges are picked up as balloon sweeps across
- (+) charges are always fixed in place
- Charges should stick on the side of balloon charges hit
    - ○ If charges hit left side of balloon, they should tend to pile up on left side (as shown below)
    - ○ Vise versa for right (if this balloon had been swept from left to right across sweater, charges would be on the right)



Wall removed

- Use “Add Wall” button to put wall back



● Show all charges
○ Show no charges
○ Show charge differences

Implementation Details

We have developed our own scene graph called "Scenery" which should be used instead of Easel. The interface is fairly similar to Easel but it provides abstractions and addresses performance concerns that would have been difficult to address with Easel alone. Here is the github page for Scenery, which links to this homepage including documentation, a "tour" of different features and more specific examples. We have updated example-sim to use Scenery as an example, you can see it in this branch 'fort-scenery'. Scenery has a few dependencies that PhET wrote, such as 'kite' (geometry), dot (math), phet-core (utilities) and assert (an assertions library). Please let us know if you have immediate needs in the documentation, Scenery features, or if you see any bugs that should be solved there.

For adding bootstrap buttons and radio buttons, please create them in Javascript and add them using scenery.DOM so they can be positioned and layed out using Scenery code, see the fort-scenery branch of example-sim for an example of this. For standalone buttons and possibly button groups, separate html templates are not necessary.

For the artwork, I added as SVG file for the sweater that you can use. It is already checked into the git repository. The provided SVG graphic is different than the ones in the screenshots above. Scenery provides radial gradients which can be used for creating the +/- charges and spheres--here's an example of how it can be coded. The + and - signs should just be created using shapes in Scenery (not fonts and "+" "-" text strings). The “show” radio buttons and “reset balloon”, “reset balloons” “reset all” and “add wall” buttons should be created in Bootstrap and the 1 vs 2 balloon buttons should be created using Scenery. I also added the balloon images into the balloons-and-static-electricity git repository. Using scenery, you should choose whether canvas or SVG/CSS is better for performance in rendering the shapes.using this option in the Node constructor: renderer: 'svg', rendererOptions: {cssTransform: true} otherwise the default is canvas.

Use submodules for PhET library dependencies such as assert, dot, kite, phet-core, phetcommon and scenery. A stable version is already checked in to the repo into libs/phet/. If you see any bugs in those libraries, please let us know right away so they can be resolved and we can set up another stable checkout of dependencies. Other dependencies should go in libs/

Property.js vs Fort.js For this simulation, we have the option to use Property.js as in the previous sim, or Fort.js (built on the open source project Backbone.js) which is being developed as a replacement for Property.js. I would like to hear cost estimates for each approach. If you have experience with Backbone, please let us know. Also, both "fort" and "scenery" are being used in the "fort-scenery" branch of example-sim as further examples of how to use it.