A Brief Description of the SAIL Environment

Stephen Bannasch and Robert Tinker

July 31, 2008

Overview

SAIL (the Scalable Architecture for Interactive Learning) is both a framework and a collection of applications and databases that permit non-programmers to create, modify, discover, and deploy from the Web dynamically created learning activities with embedded highly interactive simulation, modeling, probeware, graphing, and analysis components. All the work a learner does while interacting with a SAIL client-based activity—visiting pages, answering questions, creating a drawing, investigating a model, collecting data from probes—is saved over the network and is available for the learner when the activity is run again. In addition, all the learner's work is available for a teacher or researcher to review.

The current SAIL framework, which is an outgrowth of the NSF-funded Web-based Inquiry in Science Environment, has been developed by the NSF center for Technology Enhanced Learning in Science (TELS) as an optimal environment for the development of inquiry learning investigations. SAIL represents the product of over a decade of prior research into technology-enhanced student learning. Technologists at the Concord Consortium and U.C. Berkeley have been leading SAIL development for the past several years.

SAIL Architecture

There are two key ideas in SAIL. One involves and architecture for assembling reusable, pedagogically-aware Java components into curricular activities. These rich components already include:

Computational models with rich visual representations. These include, among others, molecular dynamics and biological models.

Graphs for displaying both real-time and saved data.

Sensor collection components for collecting and graphing real-time data from sensors as well as analyzing data collected previously.

Drawing tools that support a range of formats from a simple bitmapped painting, to object drawing, to concept mapping.

Models written in general purpose-modeling languages such as NetLogo.

Assessments ranging from multiple-choice to open-response text input.

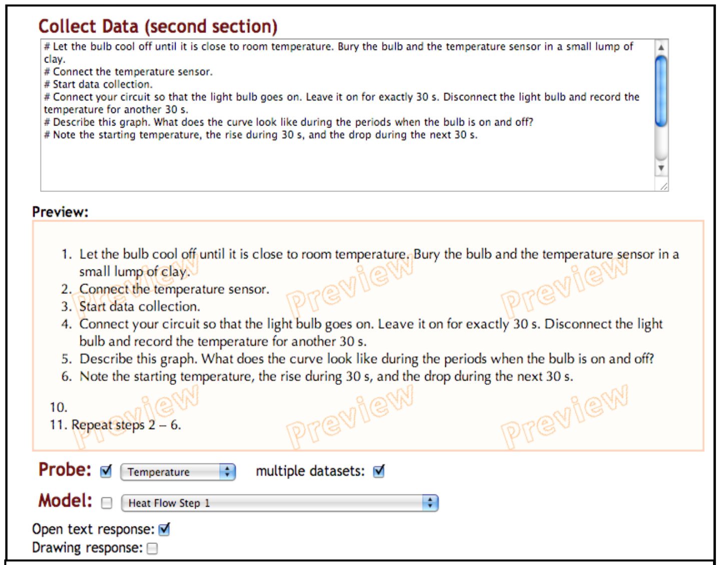
Components that can render web content ranging from html, css, to flash and QuickTime. While browsers are capable of this, there are many times in which web content may need to be delivered in a more constrained environment which does not necessarily allow browsing to other sites.

The integration of the many forms of web content and interaction with the more powerful modeling and analysis tools that are available in Java to deeper learner exploration and inquiry and the creation of both richer explicit and implicit learner artifacts.

The second key idea is that SAIL delivers these components a network-enabled pedagogically-aware persistence service that lets the components load and save learner data. The underlying SAIL architecture takes care of storing a complete revision history of what has been saved and also makes sure that the data are associated with the correct student, workgroup, class, and teacher. This persistence is supported by the core SAIL framework that is included with the client application and the SAIL Data Service (SDS) web service.

The SDS is designed to integrate with existing web portals to allow them to easily deliver SAIL-based activities to learners, persist the learner data, and report back to he main portal. At this time, the SDS is supporting the TELS WISE portal as well as Concord Consortium's TEEMSS2 Do-It-Yourself portal. These are two completely different portals with different underlying architectures integrated with SAIL and the SDS to support authoring and deploying SAIL-based curricula.

Jim Slotta developed the SAIL concept and leads further development of it at Berkeley and Toronto. Concord Consortium, under Stephen Bannasch's leadership has been the technical lead on the SAIL and SDS persistence integration as well as the author of many of the modeling components. Concord has also developed a scripting environment and framework called Pedagogica that supports dynamic adaptation of component presentation and interaction to learners based on learner actions and data. For simplicity, we are using the term "SAIL Environment" to refer to all the supporting software, the SAIL framework, SDS, the scripting environment, and Pedagogica.



Collect Data (second section)

Let the bulb cool off until it is close to room temperature. Bury the bulb and the temperature sensor in a small lump of clay.
Connect the temperature sensor.
Start data collection.
Connect your circuit so that the light bulb goes on. Leave it on for exactly 30 s. Disconnect the light bulb and record the temperature for another 30 s.
Describe this graph. What does the curve look like during the periods when the bulb is on and off?
Note the starting temperature, the rise during 30 s, and the drop during the next 30 s.

Preview:

1. Let the bulb cool off until it is close to room temperature. Bury the bulb and the temperature sensor in a small lump of clay.
2. Connect the temperature sensor.
3. Start data collection.
4. Connect your circuit so that the light bulb goes on. Leave it on for exactly 30 s. Disconnect the light bulb and record the temperature for another 30 s.
5. Describe this graph. What does the curve look like during the periods when the bulb is on and off?
6. Note the starting temperature, the rise during 30 s, and the drop during the next 30 s.

10.
11. Repeat steps 2 – 6.

Probe: ✓ Temperature multiple datasets: ✓
Model: ☐ Heat Flow Step 1
Open text response: ✓
Drawing response: ☐

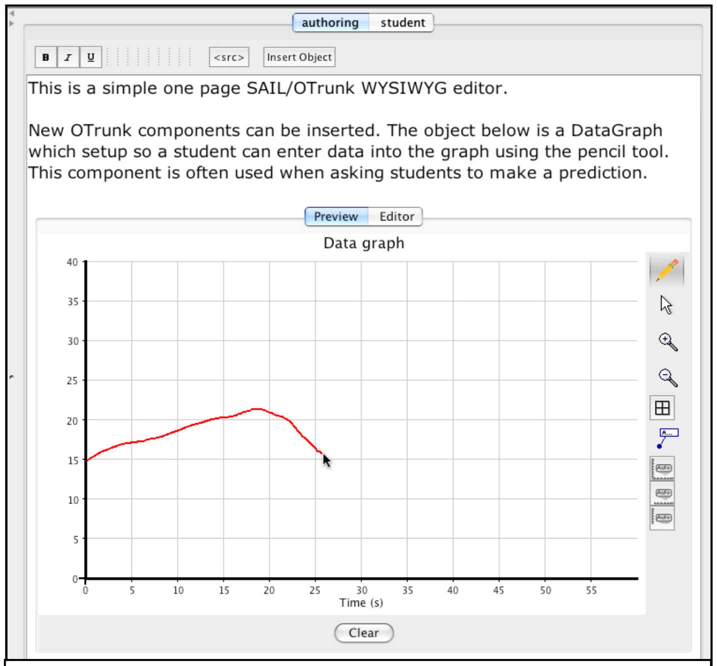
Figure 1: The DIY ITSI Editor. This shows one section of a page with the user input in Textile and the resulting rendering. It also shows a section where the author can select a model or probe.

Creating Activities

We have created several SAIL activity editors that can allow materials developers to combine components into complete learning experiences, which we call SAIL learning activities. The activities that are produced can start life as blanks or recycled activities.

The editors reflect the specific needs of different projects at CC and the growing capacity of the software.

The following editors have been created at CC. Other editors can be easily developed that give different appearance and provide different affordances.



This is a simple one page SAIL/OTrunk WYSIWYG editor.

New OTrunk components can be inserted. The object below is a DataGraph which setup so a student can enter data into the graph using the pencil tool. This component is often used when asking students to make a prediction.

Figure 2: A WYSIWYG editor. In this example, the author has inserted some text and a graph object.

The DIY ITSI Editor

This editor creates single pages in a format consisting of a fixed sequence of sections. For each section, the user simply fills content (or changes) into forms and selects options. The user can preview each section in the authoring environment. An activity typically requires a fraction of a class period. Figure 1 is an example of editing the Collect Data section of an activity of an ITSI activity in a web browser.

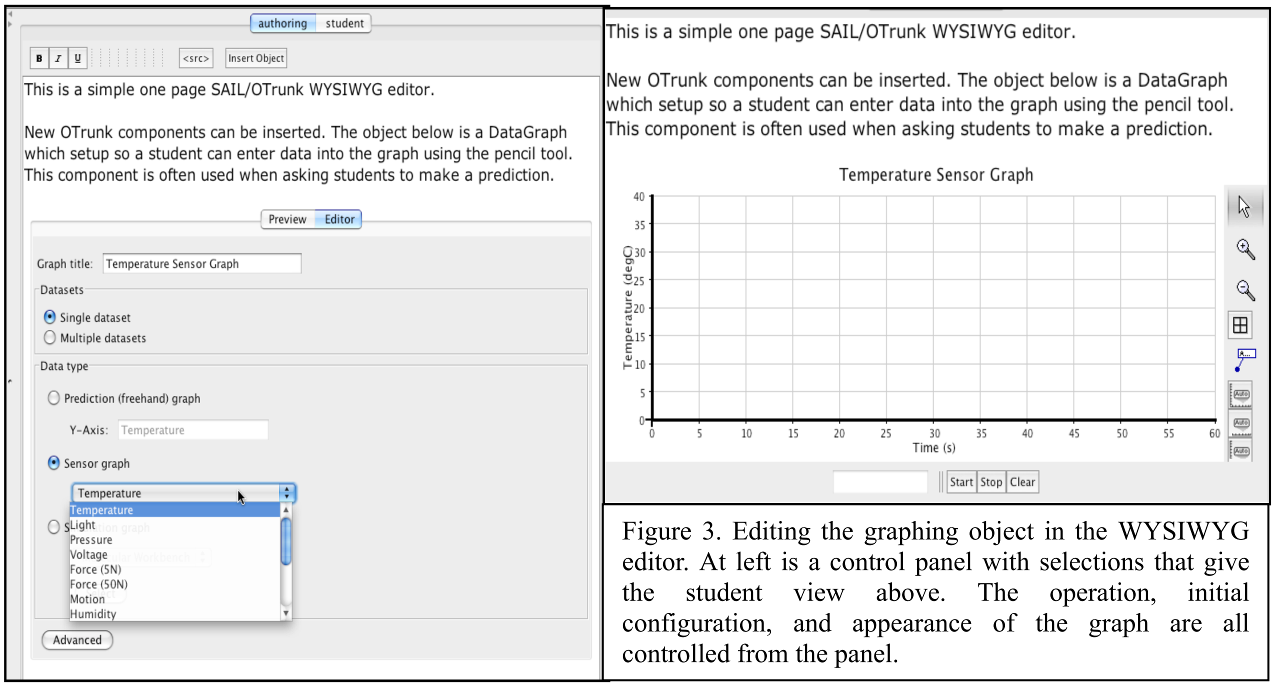
This editor is the first we created in the SAIL Environment and its relative simplicity has unleashed enormous creativity. Our staff recreated materials from several different collections using this editor, creating over 100 short activities. Teachers in our workshops used these as starting points and created over 100 additional meaningful customizations. Other projects have begun using this editor as a way of testing ideas quickly.

<u>The WYSIWYG Editor</u>

We have created a WYSIWYG editor that uses a "flow" metaphor, permitting objects and text to be intermixed on the page. In this editor, large objects such as models and graphs, are treated as large letters that are placed on the page as text would be. Figure 3 illustrates this editor.

An important feature of this editor is that outputs generated by one component can be attached to inputs to another. This creates interesting possibilities of combining different components. Thus, for instance, it is possible to connect the output of a data collector to a graph. It is also possible to connect the data collector output to a dynamic model that can represent the state of the atoms in the system being measured. A graph sketch generated by a student can generate an output that controls the motion of an object such as a car or walking person.

Figure 3 illustrates how an author can change the data collector in Figure 2 to collect data from a sensor. Note that no programming is required. Similar panels allow the user to define data flows from any component to another.



This is a simple one page SAIL/OTrunk WYSIWYG editor.

New OTrunk components can be inserted. The object below is a DataGraph which setup so a student can enter data into the graph using the pencil tool. This component is often used when asking students to make a prediction.

Graph title: Temperature Sensor Graph

Datasets
(● Single dataset) ○ Multiple datasets

Data type
○ Prediction (freehand) graph
Y-Axis: Temperature
(● Sensor graph)
Temperature

This is a simple one page SAIL/OTrunk WYSIWYG editor.

New OTrunk components can be inserted. The object below is a DataGraph which setup so a student can enter data into the graph using the pencil tool. This component is often used when asking students to make a prediction.

Figure 3. Editing the graphing object in the WYSIWYG editor. At left is a control panel with selections that give the student view above. The operation, initial configuration, and appearance of the graph are all controlled from the panel.

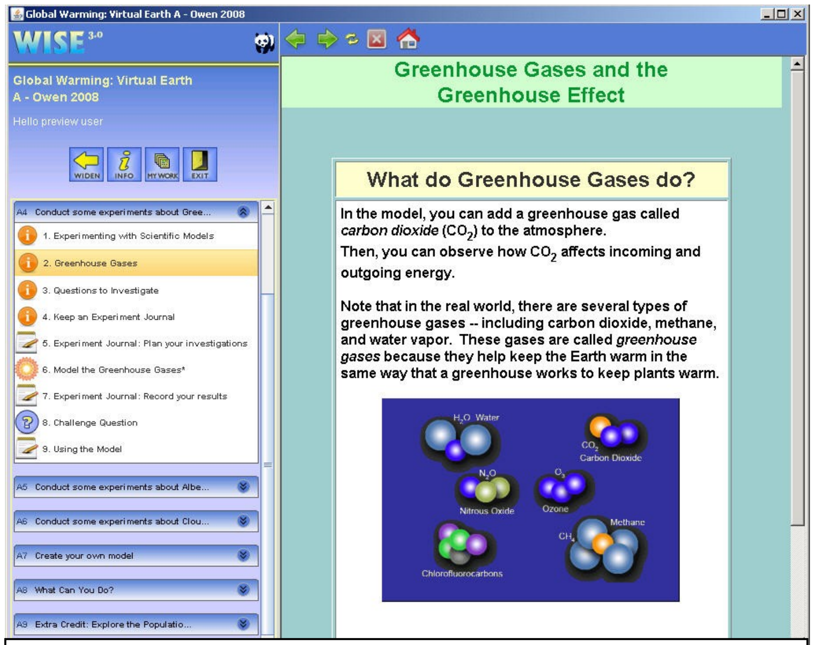
<u>PAS—Project Activity Steps.</u>

The PAS editor is designed to generate compound activities that function like older WISE "projects," creating multi-day projects from activities. A WISE project consists of multiple activities and each activity consists of several steps. Because the new PAS editor uses the SAIL Environment, PAS creates WISE projects that can persist data. This means that a student can quit mid-project and resume later at the same point, even using a different computer. This was not possible with earlier versions of WISE. Additional enhancements have been added, justifying calling the products of PAS "WISE 3" activities. Figure 4 illustrates a WISE 3 activity generated from the PAS editor.

Currently, the PAS editor cannot place multiple components in a step, although components can consist of Molecular Workbench, NetLogo, or ITSI activities which themselves can contain multiple components created using their own editors. It is also the case that the PAS editor cannot now be used to define real-time data flows from one component to another.

The Tools and Components

The editors described in the previous section assemble learning activities from tools and other components. One of the strengths of SAIL is that the growing number of compatible components can be used by any of the editors. In addition, in some cases the components can be combined and share data.



Global Warming: Virtual Earth A - Owen 2008

Hello preview user

A4 Conduct some experiments about Gree...
1. Experimenting with Scientific Models
2. Greenhouse Gases
3. Questions to Investigate
4. Keep an Experiment Journal
5. Experiment Journal: Plan your investigations
6. Model the Greenhouse Gases*
7. Experiment Journal: Record your results
8. Challenge Question
9. Using the Model
A5 Conduct some experiments about Albe...
A6 Conduct some experiments about Clou...
A7 Create your own model
A8 What Can You Do?
A9 Extra Credit: Explore the Populatio...

Greenhouse Gases and the Greenhouse Effect

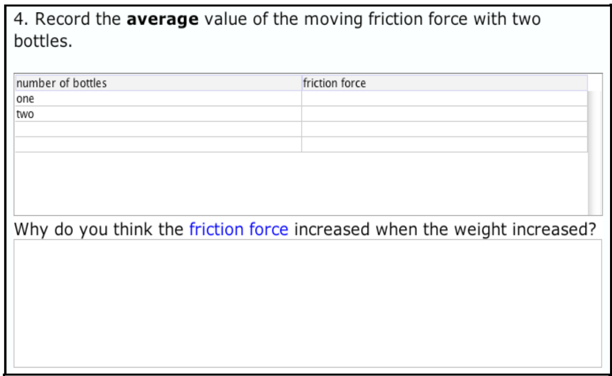
What do Greenhouse Gases do?

In the model, you can add a greenhouse gas called *carbon dioxide* (CO_2) to the atmosphere.
Then, you can observe how CO_2 affects incoming and outgoing energy.

Note that in the real world, there are several types of greenhouse gases -- including carbon dioxide, methane, and water vapor. These gases are called *greenhouse gases* because they help keep the Earth warm in the same way that a greenhouse works to keep plants warm.

Figure 4. WISE 3 authored using the Project Activity Steps editor. A step is shown on the right and located within the activity structure on the left.

For compatibility, SAIL components have to use a consistent API (application program interface) called OTrunk. OTrunk has been developed at CC to connect arbitrary objects that can be imagined as being pulled out of a trunk. This section briefly describes the major SAIL components currently supported with OTrunk interfaces. Any one or combination of these objects can be used by the editors to create learning activities.

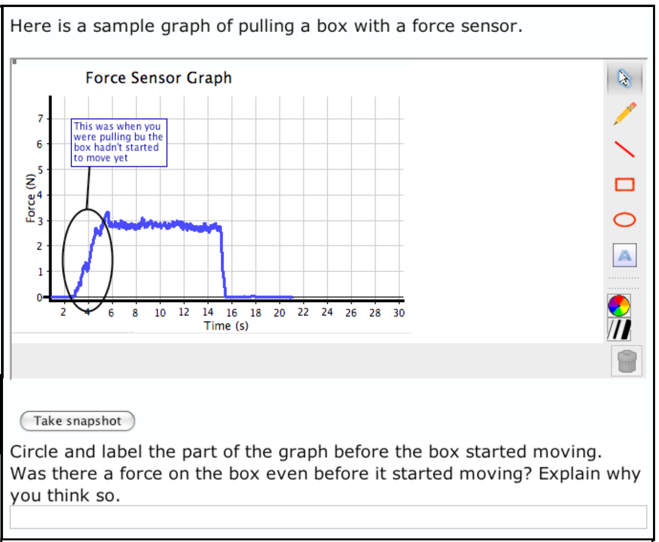


4. Record the **average** value of the moving friction force with two bottles.

number of bottles	friction force
one	
two	

Why do you think the friction force increased when the weight increased?

Figure 6. CCTable supports both displaying and entering data.



Here is a sample graph of pulling a box with a force sensor.

Circle and label the part of the graph before the box started moving. Was there a force on the box even before it started moving? Explain why you think so.

Figure 5. CCGraph showing data collected from a force sensor. The student has used the annotation function to highlight a part of the graph.

CCGraph

CCGraph is a powerful and flexible grapher that can display line, scatter, or bar graph data collected from sensors, generated with equations or models, imported from a spreadsheet, entered into a data table or sketched by hand. The graph scales can be easily changed, annotations affixed to the graph, and drawings added. Exciting new functions are under development that will add functionality to all software that uses the grapher. See Fiture 5.

CCDraw

Based on CCGraph this is drawing program that can be used by itself or used to annotate other objects.

CCTable

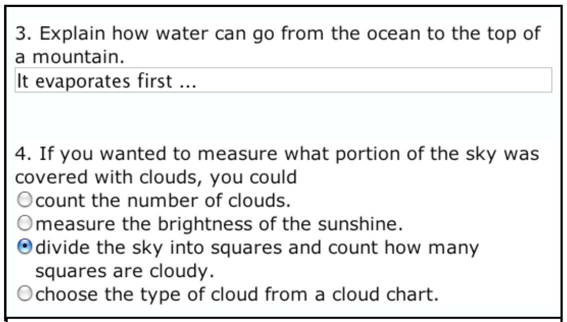
This is a simple table that can be used to enter or display data. See Figure 6.

Standard Queries

A set of components support open responses, multiple-choice items, and check lists.

Custom Queries

Any standard query can be customized so that the student response uses any other object that saves state. Variations of these include open response where the response is a drawing, or a prediction graph. See Figure 7.



3. Explain how water can go from the ocean to the top of a mountain.

It evaporates first ...

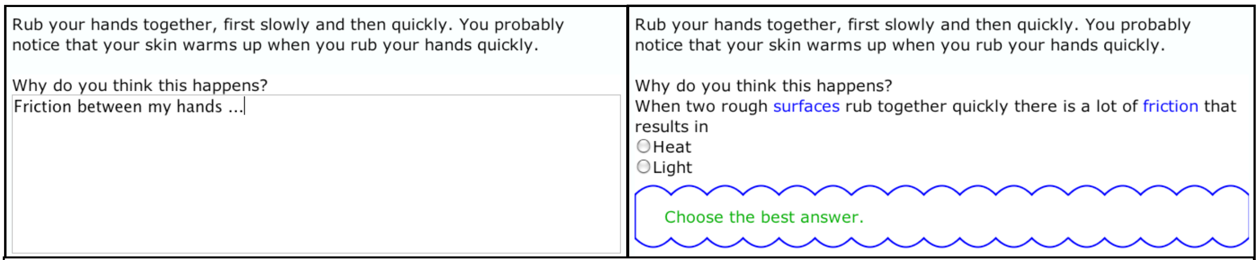
4. If you wanted to measure what portion of the sky was covered with clouds, you could

- ○ count the number of clouds.
- ○ measure the brightness of the sunshine.
- [✓ divide the sky into squares and count how many squares are cloudy.]
- ○ choose the type of cloud from a cloud chart.

Figure 7. Some of the queries that can be generated by the custom query object.

Scaffolded Queries

This is a query form that supports variable scaffolding. A single scaffolded query contains of one question and multiple hints each identified with a different "level." The user can be allowed to sequence through the levels, or the teacher can decide what level is appropriate for a given student. This can be used to provide differentiated instruction. Figure 8 illustrates two levels for the same question.



Rub your hands together, first slowly and then quickly. You probably notice that your skin warms up when you rub your hands quickly.

Why do you think this happens?

Friction between my hands ...

Rub your hands together, first slowly and then quickly. You probably notice that your skin warms up when you rub your hands quickly.

Why do you think this happens?

When two rough surfaces rub together quickly there is a lot of friction that results in

- ○ Heat
- ○ Light

Choose the best answer.

Figure 8. Two different levels for the same question. The meaning of the levels can be set by the author.

SnapShot Album

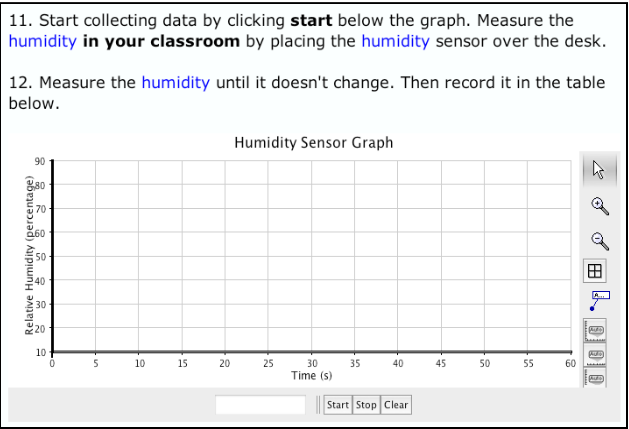
These support taking and annotating snapshots of the state of any embeddable object. The annotated snapshots are dropped into an album object which supports further annotation, ordering, and deletion.

LabBook

This is similar to the SnapShot Album however the content selected by the learner to save is stored in a document form as though it was saved to disk. This allows the learner to use the application that generated the data to edit the content.

Probeware

The probeware component supports collecting

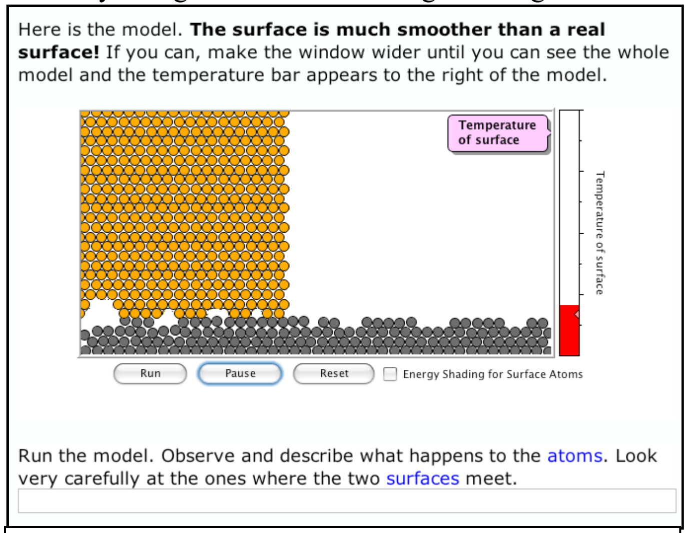


11. Start collecting data by clicking **start** below the graph. Measure the humidity **in your classroom** by placing the humidity sensor over the desk.

12. Measure the humidity until it doesn't change. Then record it in the table below.

Figure 9. Output from the probe data collector component can be sent to CCGraph as shown.

data in real time from a range of probes. Most common probe types connected through seven different interfaces manufactured by five different vendors are supported. It is straightforward to add more probes and interface types. Figure 9 shows part of an activity using the data collector generating data that is displayed in real time using CCGraph.


Here is the model. **The surface is much smoother than a real surface!** If you can, make the window wider until you can see the whole model and the temperature bar appears to the right of the model.

Run the model. Observe and describe what happens to the atoms. Look very carefully at the ones where the two surfaces meet.

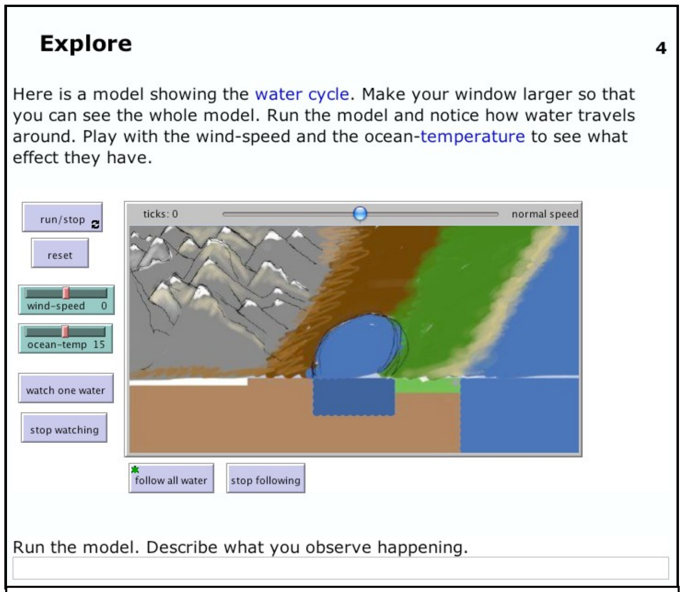
Figure 10. A Molecular Workbench page containing the MW engine and other components.

Molecular Workbench.

Almost any Molecular Workbench model can be easily embedded into SAIL. At this time over 200 MW models have been embedded. These cover topics as diverse as Brownian motion, latent heat, change of state, stoichiometry, elastic collisions, and pendulum motion. There are hundreds more high quality models available for use. It is straightforward to create more using the powerful and mature MW authoring environment.

NetLogo.

As with MW, almost any NetLogo model can be easily added. Currently 25 NetLogo models have been embedded covering topics as diverse as sheep population, heat diffusion, and global climate change. See Figure 11.


Explore 4

Here is a model showing the water cycle. Make your window larger so that you can see the whole model. Run the model and notice how water travels around. Play with the wind-speed and the ocean-temperature to see what effect they have.

Run the model. Describe what you observe happening.

Figure 11. A NetLogo model in an activity created by the DIY editor.

BioLogica.

This is a widely-used genetics and evolution model developed at CC that simulates classical genetics at the molecular, phenotype, individual, and population level.

PhET Simulations

The Physics Education Technology (PhET) group at the University of Colorado has developed a large number of open source science model written in Java. Currently, we have interfaced the PhET Circuit Construction Kit as well as the Wave Interference, Sound, Faraday, Discharge Lamps, and Energy Skate Park. Many others could easily be added.

Seeing Math Algebra Interactives

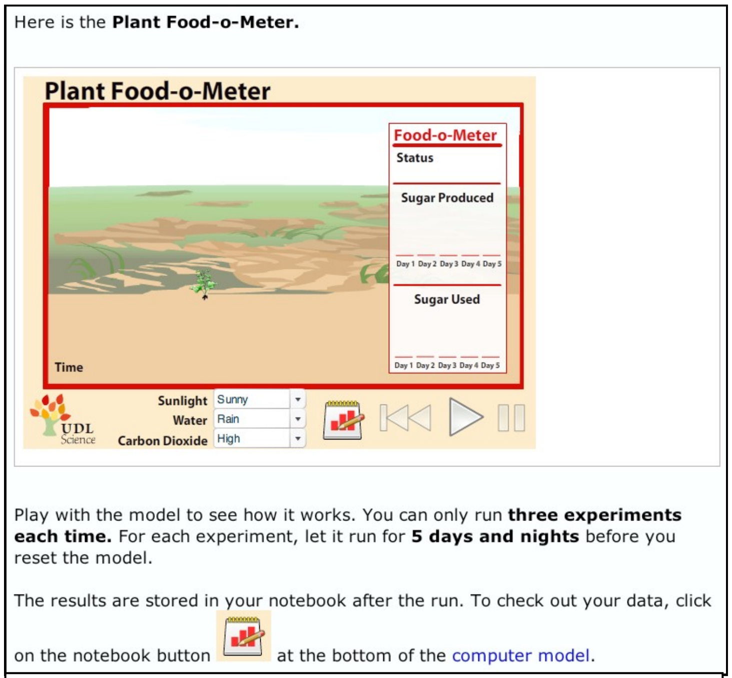
We have adapted five of the Algebra Interactives created at CC by a middle school math project. These include the Qualitative Grapher, Linear Transformer, Function Analyzer, and Quadratic Transformer.

Mozilla

We have integrated the Mozilla web framework (FireFox) into Java. Using it, authors can include web content into an activity. In addition the scripts that control the display of the pages are accessible to Java allowing dynamic content modification appropriate for the learner.

Flash

An author can include any component that can be embedded into FireFox, including Flash applications. In addition, as long as the embeddable object supports the functionality, we can both set and read values in the object. For example, we can use the state of an evolving MW model showing a microscopic simulation to drive a Flash animation showing a macroscopic representation. Figure 12 illustrates a simple model in SAIL.


Here is the **Plant Food-o-Meter.**

Sunlight Sunny
Water Rain
Carbon Dioxide High

Play with the model to see how it works. You can only run **three experiments each time.** For each experiment, let it run for **5 days and nights** before you reset the model.

The results are stored in your notebook after the run. To check out your data, click on the notebook button at the bottom of the computer model.

Figure 12. An embedded Flash application modeling plant biology running in a SAIL Java activity.

Executing Activities

The learning activities have properties that generated through student use. Some of these are explicit such as answers to questions, while many are implicit such as how many times the student used he model and what variables they changed while trying to solve the challenge.

The activity sends data to the SAIL Data Service (SDS) as soon as a student completes an action. In normal use, if a registered user aborts the activity, it can be restored later to its last-saved state for that user from any computer. Processes running in the SDS analyze the data and generate student performance data. Performance data is sent to a teacher portal where it is displayed. For UDL applications, the teacher portal can generate modifications in the activity to match individual student needs. Other processes can generate data for a research portal.

We associate metadata with each activity. Currently this includes use and publication (public/private) data only. We plan to add

Descriptive data: Grade, topic, standards, etc.

Teacher guide: Description of content, background, student misconceptions.

Assessments: Student assessment strategies and actual items.

Evaluation data: Review results, pre-post student gains.

Provenance: Authors, revision history, reasons for revisions.

We are working on two teacher portals that can display SAIL data. One is an outgrowth of the WISE work and one serves the needs of several current CC projects.

References

Software Models and Simulations:

PhET Interactive Physics Simulations

Physics Education Technology project at the University of Colorado:

http://phet.colorado.edu/web-pages/index.html.

Molecular Workbench

http://mw.concord.org/modeler/index.html

Seeing Math Algebra Interactives

http://seeingmath.concord.org/sms_interactives.html

WISE—Web-based Inquiry in Science Education

http://wise.berkeley.edu/

Ongoing CC projects using this these frameworks:

TELS, Center for Technology Enhanced Learning in Science:

http://www.telscenter.org/

ITSI, Information Technology in Science Instruction:

http://itsi.portal.concord.org/

LOOPS, Logging Opportunities in Online Programs for Science

http://confluence.concord.org/display/LOOPS/Home

UDL, Universal Design in Science Education

http://udl.concord.org/

CAPA, Computer-Assisted Performance Assessment

http://capa.concord.org/

Geniquest

Genomics Inquiry through Quantitative Trait Loci Exploration with SAIL Technology

http://confluence.concord.org/display/GEN/Home

Technology Frameworks:

SAIL, Scalable Architecture for Interactive Leaning

http://www.telscenter.org/confluence/display/SAIL/Home

OTrunk, Object Trunk:

https://confluence.concord.org/display/CSP/OTrunk

Citations