PEARSON NEW INTERNATIONAL EDITION

Meaningful Learning with Technology

Jane L. Howland David H. Jonassen
Rose M. Marra

Fourth Edition

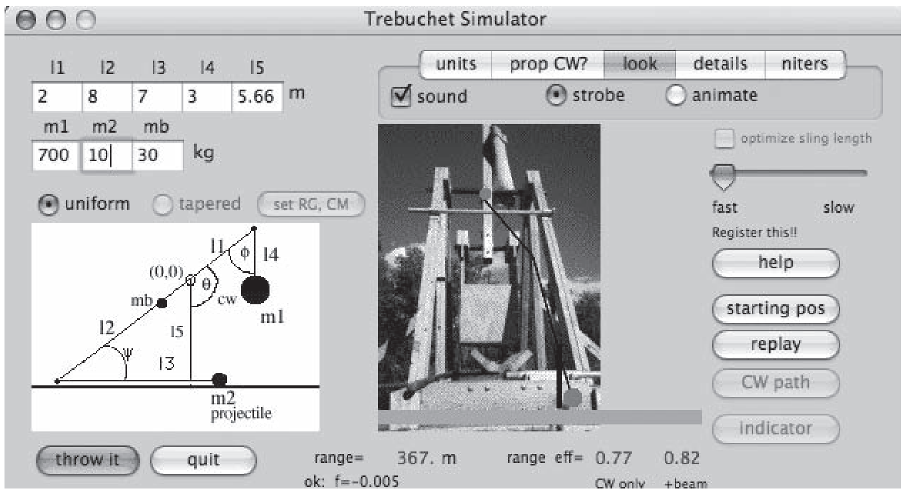
Pearson New International Edition

Meaningful Learning with Technology
Jane L. Howland David H. Jonassen
Rose M. Marra
Fourth Edition

PEARSON®

Figure 4.5

Results from Tests on Different Designs Appear at the Bottom of the Trebuchet Simulator Program



l1	l2	l3	l4	l5	
2	8	7	3	5.66	m

m1	m2	mb	
700	10	30	kg

[✓ uniform] tapered

[✓ sound] [✓ strobe] animate

range= 367. m
ok: f=-0.005
range eff= 0.77 0.82
CW only +beam

Problem Solving through the Design of Games and Animations

Software design is a content area where researchers conducted some of the early studies on the use of learning technologies to create meaningful learning environments. In fact, Seymour Papert's (1980) Logo programming language was one of the first contexts in which constructionist pedagogy was ever enacted (Kafai, 2005). Students using Logo could type computer commands and control an "evocative object" (Turkle, 1995): Logo used an on-screen turtle that made drawings on screen. By watching the turtle move and draw lines through these commands, children got the immediate feedback they needed to make debugging their programs more concrete and doable. This contrasted to kids programming with languages like BASIC, which allowed manipulation mainly of text or numbers. Studies in the 1980s showed that young children not only learned to program in Logo but also thought more reflectively and creatively (Clement & Gullo, 1984), although later research showed these gains were bound to the contexts in which students did their original work (Pea, Kurland, & Hawkins, 1985).

A number of Logo-inspired products have been developed since then, including "microworlds" that gave students creative control over multiple characters on screen, "construction kits" that enabled students to manipulate Lego bricks in the real world, and environments where students could design software for their peers (Kafai, 2005).

The process of designing games and animations draws on complex problem solving and decision making in the creation of original products, or the redesign of existing and even "classic" programs. Students must hypothesize and make predictions, create rules and test them, and learn from their initial attempts when they do not work. Students use many of the competencies promoted by 21st Century Skills and NETS as they engage in active, intentional, authentic, constructive, and cooperative learning processes (refer back to Figure 1.1). This section explores software that enables students to build their own games and simulations.

Scratch

Scratch is software that empowers users to create object-based programs that manipulate digitized video and audio. The Scratch program, which runs on most computer platforms, gives students the capability to build programs that manipulate digital images using tools similar to Photoshop's, add and replay audio tracks and music, and work with video, all while learning fundamentals in computer programming. Scratch's interface uses a Logo-styled building-block metaphor to represent programming moves. It replaces code writing with virtual programming blocks that users can insert, move and combine, and set values to variables through drop-down menus.

Scratch's creators targeted it to appeal to preteens and teenagers from different cultures, support them in creating projects they could be proud of, and, when used over time, could lead to the development of more advanced programming skills.

The main Scratch screen (Figure 4.6) has five main work areas. The "Blocks" palette to the left provides a collection of programming tools that manipulate imported digital images, sprites, and music through programming procedures ("control" commands are visible in Figure 4.6, left). In the middle-left "Scripts" area, users can drag blocks to construct sets of procedures, much as puzzle pieces can be fitted together. When procedures are not compatible, students encounter the graphical equivalent of a mismatch of puzzle parts. Scratch forces students to alter their program until they are in a form that can be implemented. Scripts manipulate "sprites," which are visible in the lower-right area. The full program is run by clicking on the green "go" flag at the right top, the results of which can be viewed in the "Stage" area.

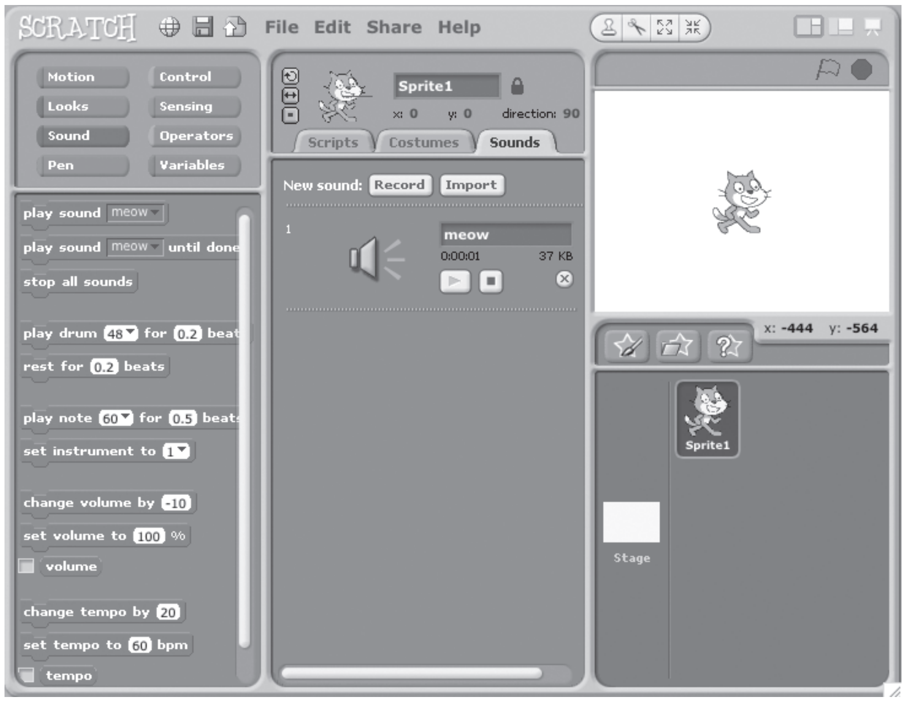
The program's capacity to give timely feedback helps students "tinker" in productive ways and supports meaningful exposure and learning of modular programming. Simultaneously, these features allow students to produce interactive stories, animations, games, music videos, and other creative projects in the media arts. Scratch is developed by the Lifelong Kindergarten Group at the MIT Media Lab (see http://scratch.mit.edu). The program is available for free download. You may also be interested in Scratch-Ed (www.scratch-ed.org), an online space where educators can share stories, exchange resources, and ask questions.

iStopMotion and Stagecast Creator

The iStopMotion software, produced by Boinx (www.istopmotion.com), enables designers as young as elementary students to create their own animated movies and time-lapse recordings. Using a Mac computer, digital camera, or webcam, students can capture and

Figure 4.6

Scratch has Different Sections with Programming, Graphics, and Sound Tools; a Place to Compose Scripts; and a Stage (right) Where Sprites Get Animated

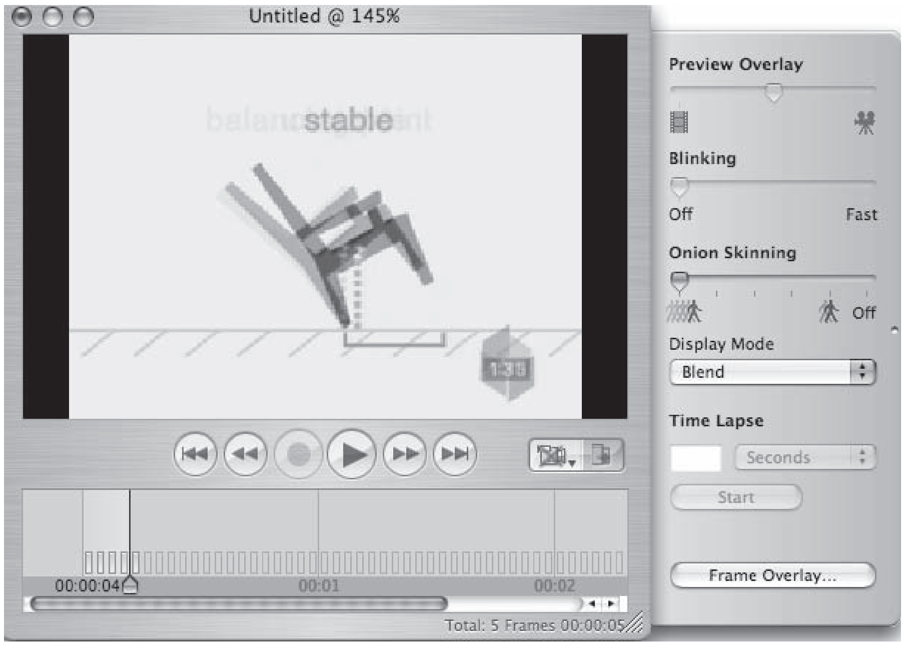


then compose animations frame by frame with this easy-to-learn application. When its "onion skinning" feature is turned on (Figure 4.7), the program allows for the easy juxtaposition of contiguous frames when making a movie file. This helps users produce a coherent video sequence without jerky movements from its characters, helping to limit the need for reshooting or reediting. Claymation movies can be easily made with iStopMotion. Children are capable of composing single-frame shots of the easy-to-manipulate clay figures whose evolving positions and changing shapes collectively tell a desired story.

George Rota, media specialist at Glen Lake Elementary School in Minnetonka, Minnesota, works with elementary teachers in his school to create animation and claymation movies. He introduces iStopMotion to all sixth graders at Glen Lake, who then use it to complete an extended animation project that is part of their social studies and language arts classes. Students work in groups of three or four, do research on their topics, and use index cards to create storyboards of major scenes when planning their movies. Their videos typically last from 20 to 60 seconds, not including titles and credits, which often double the total length of students' final movies.

Figure 4.7

iStopMotion's "Onion Skinning" Helps Ensure That Contiguous Frames Have Just the Right Amount of "Jump" between Each Other to Create Seamless Animations or Pans



Stagecast Creator, originally developed by Apple and available for Mac, Windows, and Unix systems, is another application that enables people to create stories, games, and interactive simulations using a visual interface rather than a programming language.

AgentSheets

Developed through research at the University of Colorado-Boulder, AgentSheets aims to foster K–12 student participation in STEM fields through design of applications using its technology. AgentSheets 3 employs the use of a "Conversational Programming" model by allowing the user to create an agent (i.e., "programming buddy") who moves about in the environment that is being designed to analyze and provide feedback on not only syntax, but also semantics. The agent can help students consider the consequences of rules and conditions they are creating by providing feedback on one's logic and pointing out conceptual mistakes. This preemptive troubleshooting prompts students to experience the meaning of the commands they've written in real time and results in a highly interactive iterative design process.

AgentSheets offers a Scalable Game Design wiki (http://scalablegamedesign.cs.colorado.edu/wiki/Scalable_Game_Design_wiki) with an extensive library of teacher resources. Sample tutorial simulations (e.g., Avalanche, Contagion/Virus, Ecosystems, Electricity, Fish Farm) may be used to support concept development in math or science curriculum (e.g., probability, ratios, and exponential vs. linear growth) and data analysis by exporting simulation data for graphing, determining line of best fit, or making statistical predictions. These simulations have links to related lesson plans, learning outcomes and national standards (e.g., NETS•S, NSES, NCTM Math), sample grading strategies, detailed tutorials, and suggestions for exploring questions with the simulation. For example, the Forest Fire Design simulation allows users to study the effects of different variables on forest fires by manipulating tree density and wind direction, and how those parameters affect the probability of a fire spreading or dying out. "Making Frogger," which is based on a classic Sega-developed arcade game from 1981, asks students to redesign the original arcade game, in which a frog must overcome obstacles and predators to get across a river and reach the opposite shore. In addition to lesson plans, learning outcomes and standards, and sample grading strategies, several detailed tutorials lead one through the steps of creating very basic through more advanced levels of Frogger games. The rich resources and support that AgentSheets offers for both students and teachers increases the likelihood that students will be successful in their design work without experiencing undue frustration. In particular, the "programming buddy" may scaffold students as they problem-solve design dilemmas. These are highly instructive examples of learning not *from*, but *with*, technology.

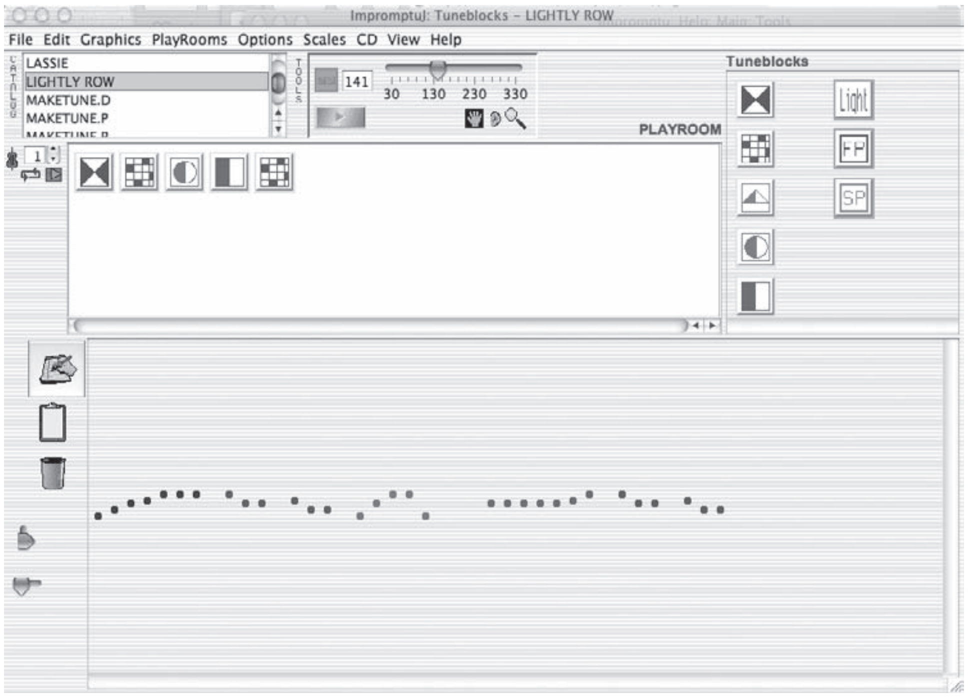
Designing Music with Composition Software

Applications for developing songs and scores on the computer have been available since the days when microcomputing first began sporting plug-in sound cards and could produce the synthesized "voices" of various musical instruments. These programs—and their subsequent offspring, including Apple's GarageBand—display a number of music tracks on which users can enter, edit, and replay their compositions. Depending on the program, notes can be inputted via direct recordings, playing a guitar controller or MIDI keyboard attached to the computer, or doing "old-fashioned" note-by-note inputting via a mouse onto a five-line staff. Applications such as these do for music composition what word processors have done for writing—they help users who start with a blank page to enter and format material easily, and to review, revise, and save groups of notes while working toward a finished composition.

A pedagogically powerful, elegant, and free piece of software for designing music, called Impromptu, was created by MIT professor emerita of music and urban education, Jeanne Bamberger. Her program gains inspiration from the children's programming language, Logo. Impromptu's workspace (Figure 4.8) is composed of five areas and a menu bar that supports students in developing their own musical intuitions (Bamberger, 2001, 2003) while constructing tunes from groups of musical notes called "Tuneblocks." The "Tuneblocks" area is the place where music chunks or motifs can be found and stored and where collections of prefabricated blocks, based on previously transcribed musical pieces, can be loaded from the program's "catalog." Users can assign Tuneblocks a color and an

Figure 4.8

The Impromptu Program Has Five Work Areas. Three Cursor Types Found in the Tools Area Allow Tuneblocks to Be Moved, Heard, and Modified



abstract pattern to help them identify one from another. They then arrange them as they wish in the "Playroom" area of the program's main window.

The key to this program's approach, seen earlier in Logo and also in Scratch, lies in how it has users—primary school aged or older—work with musical "chunks," or groups of musical sounds, rather than individual notes on a five-line score when constructing a musical piece. This enables users to grasp the composition more as a conceptual whole, as it contains far fewer parts, and each part has its own familiar sense and meaning. The immediate feedback that Impromptu gives to its users—an approach shared by many programs described in this chapter—helps users conduct their own experiments in musical meaning making. Impromptu also provides multiple representations of the collected notes contained in a row of Tuneblocks in its "Graphics" area (bottom of main window), where the relative relationship of individual notes according to pitch, rhythm, or both can be seen. The use of multiple representations has helped students build more flexible and transferable understandings since the 1980s.