A Framework for Microworld-style Construction Kits

Adrienne H. Slaughter
Stanford University
c/o 310 W. Michigan Avenue
Clinton, MI 49236 USA
+1 517 456 4822
ahs@alum.mit.edu

Carol Strohecker
MERL - Mitsubishi Electric Research Laboratories
201 Broadway
Cambridge, MA 02139 USA
+1 617 621 7517
stro@merl.com

This paper appears in Proceedings of EdMedia 2001, World Conference on Educational Multimedia, Assoc. for the Advancement of Computing in Education. An earlier version appeared as MERL Technical Report 2000-19.

Abstract. We describe a genre of game-like, microworld-style construction kits and an extensible Java framework that models method and structures for generating them. We include descriptions of example kits' conceptual bases and design principles. We also explain how work with users of the "Kit for Kits" framework addresses needs of people with a broad range of expertise in programming, design, multimedia production, and learning theory. Continued development of the framework is being informed by museum-based contexts for kit use.

Introduction

We are developing a Java framework to facilitate implementation of kits based on principles of Constructionist learning (Papert 1980, Harel & Papert 1991, Kafai & Resnick 1996). The kits bear a family resemblance to Tinker Toys™, LEGOs™, and other building toys, but are dynamic and especially visual. Currently the kits are prototyped as highly interactive 2D graphical software, but we project augmentation with tangible input and output devices, and concurrent development of social contexts supporting the playful building activity.

The kits are inspired by the notion of "microworlds," perhaps best exemplified by Logo-based Turtle Geometry (Papert 1980, Abelson & diSessa 1980). This computational world elegantly models basics of differential geometry: the graphical "turtle" is characterized by just two properties, position and heading. Users (typically children) interact by typing commands to effect operations of movement. The turtle responds by translocating forward, turning, and so on, leaving a trace as it goes. Delightful pictures result, which quickly become complex. Thus children play with ideas in building-block fashion as the properties of a vector ground development of further understandings – of angles, the geometries of squares and spirals, etc.

Some of our kits are intended for children, and others are intended for people of all ages, but all involve similar modeling of a conceptual domain. Players build with basic elements and operations, and then activate the constructions. Dinosaur skeletons balance as they walk and run; maps reveal street-level views; geometric tiles form symmetric patterns; animistic creatures spawn, maintain, and disrupt social distances; and dancers' breathing patterns determine cyclic timing for a shared dance (Strohecker & Slaughter 2000).

Collaborators at MERL have developed versions of these kits during the past several years. Commonalities of design and implementation, as well as conceptual underpinnings, have led us to formulate a "Kit for Kits" (K4K), extensions to Java's Abstract Windowing Toolkit (AWT) that facilitate development of the growing genre. Because the kits are so strongly visual, much of the Kit4Kits supports creation of structure, function, and appearances of objects. K4K works with Java for capturing and dealing with actions and elaborates Java2D facilities for transforms, shapes, painting, and image-handling to provide customizable, manipulable objects.

Four existing construction kits and designs for a fifth informed development of a first version of K4K. We made it available to people who used it for developing their own kits. These experiences helped us to improve the framework. While implementing the refinements, we continued using K4K to develop our own new kit. Next steps include creating facilities for sound and for web-supported play.

K4K Beginnings

PatternMagix players build colorful tiles and spread them into mosaic-like patterns (Fig. 1, left). Panels change size as the player moves from the build area to the activation area, modeling the turn-taking of a dialog (Ackermann & Strohecker 1999, 2001). The implementation (in Java 1.1.6) involves rectangular building elements, filters effecting geometric transformations, and structures for animating image buttons.

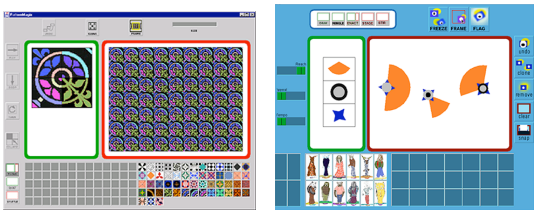

Figure 1: The *Magix* kits

AnimMagix players select and adjust attributes that simulate social behaviors of animistic creatures (Fig. 1, right). Varying degrees of awareness and attraction affect the creatures' movements as they interact with one another (Ackermann & Strohecker 1999). The AnimMagix interaction design is again based on the idea of a conversation between the player and the system. Its implementation (in Java 1.1.8) makes use of many of the same structures as PatternMagix. We added filters for color blending as creatures' perceptual fields overlap, and structures enabling transparent backgrounds for image files containing creatures' animistic costumes.

WayMaker players diagram a city, real or imagined, by forming a map from geographic primitives such as districts, paths, and landmarks (Fig.2, left) (Lynch 1960/1992). The software then shifts scale, view, and representation to illustrate a stroll through the mapped environment. The dynamic illustration takes the form of a frame-by-frame animation that preserves topological relationships among the primitive structures (Strohecker 1999). Building elements are Java 1.1.8 polygons, as in AnimMagix, but we added facilities for manipulations like scaling and stretching. We also elaborated the construction process by enabling specification of particular images, such as towers and mountains for landmarks, which appear in both the maps and path views. A miniature copy of the map echoes its development. This mini-map maintains its screen position as the display changes to reveal street-level views. Variable transparency filters allow visibility of views beneath the mini-map and effect edge blending between image segments. We also elaborated processes and structures for saving and retrieving constructions so they can be extended and/or re-viewed, and ultimately shared and traded.

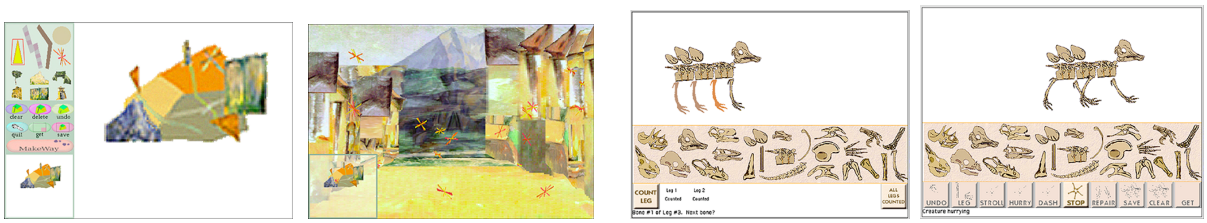

Figure 2: The *WayMaker* and *Bones* kits

Bones players put dinosaur parts together and then test the skeletons to see if they can balance (Fig. 2, right). Based on a constructed creature's number of legs, the location and mass of its center, and a user-selected speed of movement, the software analyzes whether the creature can balance as it moves. Animations, including a rich set of gait patterns, derive from literature on biomechanics and animal locomotion (Strohecker 1995). As in PatternMagix, the building elements are image files, but are arbitrarily shaped polygonal images rather than rectangular ones. This change has profound implications for image production, for mouse event structures and manipulability, and for filters that effect geometric transforms and size and color changes. We addressed these issues using Java 1.1.8. We also developed a two-phase construction process in which players first assemble a creature and then indicate which bones are meant to constitute the legs. Currently we are using Java 1.3 to improve the implementation and reformulate the leg identification process as part of the activation phase.

"Microworld" approach

The design of these kits relates to the approach exemplified by Turtle Geometry, in which basic elements and operations characterize a conceptual domain. The graphical "turtle" is defined by its position and heading, and players effect changes to these properties to create movements and emergent images. In Bones, the building elements consist of a program-assigned mass value and a player-determined position. These properties combine to locate a creature's overall center of mass, and the position of the center with respect to the base determines whether the creature will move successfully. The conceptual domain is physics; more particularly, it is motion study; still more particularly, the domain is balance; and most particularly, it is the role of center of mass in balancing. For WayMaker, the domain is topology, or basic relations of proximity. The building elements are representations of districts, edges, paths, nodes, and landmarks; operations consist of positioning the elements with respect to one another, thereby establishing spatial relations that remain constant when view and scale change. For PatternMagix, the domain is geometric symmetry. The building elements are square-shaped tiles, and operations are rotation and reflections. For AnimMagix, the domain is social dynamics. The elements are representations of sensori-motor/attract functions, and operations consist of adjustments to degrees of perceptivity, sociability, and motility.

Design highlights

Our design challenges include identifying salient features of a conceptual domain and developing representations that people can build with. We advocate a "screen-first" approach, including design heuristics of "object permanence" (c.f. Gruber & Vonèche 1977), "transparency" (c.f. Resnick et al. 1999), and "multiple simultaneous views" (Tufte 1990). Devices such as miniatures maintain accessibility of screen areas, buttons, and other mechanisms even when they are not in use. Visualizations of algorithms, calculations, and processes present difficult design and implementation challenges but are crucial to this highly visual genre (Tufte 1983, 1997). We represent constituent properties of objects, often in ways that facilitate users' modifications of them, and we provide various forms of visual and aural feedback so that results of actions are apparent. Comparisons help people to perceive the shifts of scale, perspective, time, and representation that can be needed to understand dynamic phenomena. Through such treatments we address the principle that you understand something best when you understand it in more than one way (Minsky 1986). Kits employing such treatments may also be more accessible to a range of users with diverse thinking styles (Turkle & Papert 1990).

Adjustments of panel sizes in Magix exemplify one kind of visualization. The changes represent the shared control between the player and the system (Fig. 3, left). WayMaker's side-by-side comparisons of a miniature map and path-level views facilitate understandings of spatial relationships among the elements (Fig. 3, right).

Figure 3: Dynamic screen panels in *Magix* and the anchor of the miniature map in *WayMaker*

Bones includes a diagrammatic visualization of the center-of-mass calculation, and leg movements act as visualizations of gaits (Fig. 4). We are now adding footprints as complementary visualizations of gait patterns.

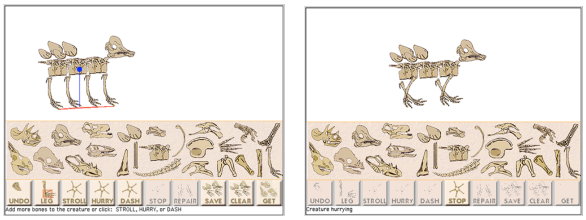

Figure 4: Visualizations of the center-of-mass calculation and gait-inspired leg movements in *Bones*

K4K Development

The Kit4Kits framework includes the "elements and operations" conceptual basis (Gruber & Vonèche 1977, Papert 1980) as well as Java structures acknowledging the importance of visual presentation to this genre of kits. We chose Java for object orientation in implementing the highly interactive systems, and in anticipation of cross-platform, web-friendly requirements as communities of players evolve.

The first version of K4K includes classes for screen areas, building elements, widgets, and structures for managing them (Slaughter & Strohecker 2000). To test the usefulness and robustness of these facilities, we invited people interested in producing "computer games more meaningful than shoot 'em ups" to try out the K4K. An interesting mix of people responded, with various degrees of programming skill supplementing experience in visual design, interaction design, and computer game use and development, as well as ecology and linguistics and cognition. This diversity was fortuitous: because of our application-orientation we had been focusing on players of the kits as "users," but from the perspective of K4K they are end-users. We needed now to address ranges and extents of expertise for people who would want to use a framework like K4K.

We organized the workshop as a design studio. During the course of a few days we hoped to explain the microworlds approach and design principles, and to work with participants in using K4K to develop cursory examples of construction kits. Demos and supporting documentation described the framework, our existing kits, their conceptual grounding and design, relevant digital image production techniques, and coding examples showing how K4K could be used to create our existing kits. There was insufficient time to do a thorough "elements and operations" analysis for the kits that participants proposed, but we addressed these concerns and emphasized the "screen-first" approach for rapid development. This approach brought designers and developers together by centering implementation around devices on-screen. It also helped us to clarify the "build – activate – save – trade" premises, emphasizing focuses on end-users' constructions of graphical objects and potential behaviors, the software's activation of constructions, and ultimately, social contexts in which saving and trading the constructions can enhance learning.

The participants created kits for exploring timing relationships through music-making, for building words from phonemes, and for simulating feeding behaviors in an ant farm. As a result of these trials, we added functionality for new kinds of interactivity and developed methods for necessary but potentially cumbersome considerations like double buffering and loading images and sounds. We then used the improved K4K in implementing our fifth kit, Zyklodeon (Fig.5). This kit (implemented in Java 1.3 beta) helped initially in identifying functionality for K4K and subsequently in testing the code for bugs, completeness, and extraneous constructs. Zyklodeon is the most complex of the kits we have implemented, mainly because of the shape and arrangement of items on the screen. As in Bones, the building elements are arbitrarily shaped polygonal image files, which in Zyklodeon represent a dancer's arms, legs, torso, and head. Players build by composing dance figures and setting properties that affect their movements and cyclic timing for a shared dance. We added arbitrarily shaped image buttons, a two-tiered structure for saving and retrieving constructions, and a greatly elaborated system of event-triggered visualizations. In this world of time/space relationships, dancers' movements visualize breathing cycles, progress through a shared dance cycle, and characteristics affecting choreographic concerns such as leap moments and heights (Strohecker et al. 2000).

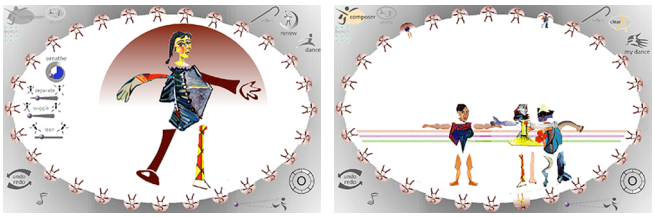

Figure 5: The *Zyklodeon* kit

Related Work

During the course of this work we compared our concerns and output to other software construction kits, authoring tools for simulations and multimedia, environments for learning about programming basics, and Java frameworks. While the distinguishing characteristic of our effort remains the basis in microworld theory, we would hope to develop or maintain features that several notable projects have also achieved or pointed toward, should our kits and the Kit4Kits move beyond prototype to more widely disseminatable forms.

Other software construction kits, such as SimLife, SimCity, Incredible Machine (and its sequel, The Even More Incredible Machine), Lemmings, Tom Snyder Productions, ToonTalk, and HyperGami, resemble K4K kits in various ways. Several of these could be described by some aspect/s of the "build – activate – save – trade" mantra. Some include a range of media as we would ultimately like for our kits. Tom Snyder Productions, for example, distributes some kits with printed booklets and activities, and HyperGami includes user-designed printed output in the construction process (Eisenberg & Nishioka 1997).

It is interesting that authoring tools for simulations – such as SimLife, SimCity, Microworlds Pro, AgentSheets / Visual AgenTalk (Repenning 2000), Cocoa, and Stagecast – tend to have more in common with environments for learning about programming basics than do multimedia authoring tools such as 3D MovieMaker, Macromedia Director, and NACDRAW (though of course a tool like Microworlds Pro can be used for both). Although it is still not clear to what extent we hope K4K will become an environment for learning about programming basics, we have learned from environments such as Microworlds Pro, ToonTalk, Cocoa, AgentSheets/Visual AgenTalk, Squeak, and SmallTalk (Goldberg & Robson 1989).[1]

The two Java frameworks we have found most relevant are the Interactive Illustrations produced at Brown University (Simpson et al. 1999) and the growing movement toward developing HCI patterns. We believe that despite the Constructionist claim of the Interactive Illustrations, our genre of kits maintains a stronger focus on users' and end-users' constructions and deliberations. We considered presenting K4K as a kind of pattern language for microworld-style construction kits, but feared engendering a "cookie-cutter" approach that would work against flexibility (Gabriel 1994, Erikson & Thomas 1997, Gamma et al. 1997, Tidwell 1999). We feel that including various forms of support for novice programmers will prove a more fruitful direction.

K4K Futures

We aim to develop better facilities for handling sound and enabling kit players' web-based trading of constructions. We are considering recent versions of Swing and Java Beans to see whether they may now be more robust and compatible with our effort. Up to now Beans have not sufficiently addressed reliance on imagery to the extent that the K4K genre demands, but we anticipate evolution of Beans and their potential compatibility with our notion of "seeds" (Slaughter & Strohecker 2000). We also plan improvements to the existing kit prototypes, which will contribute to further K4K facilities. Toward this end we have begun collaboration with exhibit and program developers at Boston's Museum of Science. Several of the kits are currently installed there as test exhibits that visitors can try and critique. In addition to improving the kits and K4K, these trials will help in developing museum-based social contexts for long-term kit use. Ideally we would develop separate versions of each kit, to support episodic use within exhibit areas and extended use within

[1] See http://www.lcsi.com, http://www.toontalk.com, http://members.aol.com/schmucker1/cocoa_faq.html, http://www.agentsheets.com, http://www.squeak.org

studio-like environments in museums, homes, and other situations networked to one another. Including tangible input and output for the kits could help to effect craft-based activities with learning communities (e.g., Wrensch & Eisenberg 1998).

References

Abelson, H., & DiSessa, A. (1980). *Turtle Geometry*. Cambridge, MA: MIT Press.

Ackermann, E., & Strohecker, C. (1999). Build, launch, convene: Sketches for constructive-dialogic play kits. MERL TR99-30, Mitsubishi Electric Research Laboratories, Cambridge, MA.

Ackermann, E., & Strohecker, C. (2001). PatternMagix Construction Kit Software. *Design Expo, CHI'2001.*

Eisenberg, M. & Nishioka, A. (1997). Creating polyhedral models by computer. *Journal of Computers of Mathematics and Science Teaching.*

Erikson, T., & Thomas, J. (1997). Putting it all together: Pattern languages for interaction design. *Proceedings of CHI'97*.

Gabriel, R. P. (1994). The failure of pattern languages. *Journal of Object-Oriented Programming*.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1977). *Design Patterns.* Reading, MA: Addison-Wesley.

Goldberg, A., & Robson, D. (1989). *Smalltalk-80.* Reading, MA: Addison-Wesley.

Gruber, H. E., & Vonèche, J. J., eds. (1977). *The Essential Piaget.* New York: Basic Books.

Harel, I., & Papert, S., eds. (1991). *Constructionism.* Norwood, NJ: Ablex.

Kafai , Y., & Resnick, M., eds. (1996) *Constructionism in Practice.* Mahwah, NJ: Lawrence Erlbaum.

Lynch, K. (1960/1992). *The Image of the City*. Cambridge, MA: MIT Press.

Minsky, M. (1986). *The Society of Mind.* New York: Simon and Schuster.

Papert, S. (1980). *Mindstorms.* New York: Basic Books.

Repenning, A. (2000). AgentSheets®: An interactive simulation environment with end-user programmable agents, *Proceedings of Interaction 2000*.

Resnick, M., Berg, R., & Eisenberg, M. (1999). Beyond black boxes. *Journal of the Learning Sciences*.

Simpson, R. M., Spalter, A. M., & van Dam, A. (1999). Exploratories: An educational strategy for the 21st century. Brown University online ID number: schoolhouse_1449.

Slaughter, A., & Strohecker, C. (2000). A Framework for Microworld-style Construction Kits. MERL TR2000-19, Mitsubishi Electric Research Laboratories, Cambridge, MA.

Strohecker, C. (1995). A model for museum outreach based on shared interactive spaces. *Multimedia Computing and Museums,* 57-66.

Strohecker, C. (1999). Toward a developmental image of the city. *Visual and Spatial Reasoning in Design*, 33-50.

Strohecker, C., & Slaughter, A. (2000). Kits for learning and a kit for kitmaking. *Extended Abstracts, CHI'2000*.

Strohecker, C., Slaughter, A. H., Horvath, M. A., & Appleton, N. J. (2000). Zyklodeon: A software construction kit modeling cyclic timing patterns. *Proceedings of ACM Multimedia.*

Tidwell, J. (1999). Common ground. http://www.mit.edu/~jtidwell/ui_patterns_essay.html.

Tufte, E. R. (1983). *The Visual Display of Quantitative Information.* Cheshire, CT: Graphics Press.

Tufte, E. R. (1990). *Envisioning Information.* Cheshire, CT: Graphics Press.

Tufte, E. R. (1997). *Visual Explanations.* Cheshire, CT: Graphics Press.

Turkle, S., & Papert, S. (1990). Epistemological pluralism. *Signs* 16:1, 128-33.

Wrensch, T., & Eisenberg, M. (1998). The programmable hinge: Toward computationally enhanced crafts. *Symposium on User Interface Software and Technology (UIST) Proceedings,* 89-96.

Acknowledgments

Bones was developed between 1993 and 2000 by Carol Strohecker, William Abernathy, Rachel Greenstadt, Daniel Gonzalez, Dan Gilman, Beatrice Witzgall, John Shiple, and AARCO medical illustrators. WayMaker was developed between 1996 and 2000 by Carol Strohecker, Barbara Barros, Adrienne Slaughter, Dan Gilman, Rachel Greenstadt, and Maribeth Back. PatternMagix was developed between 1996 and 1997 by Edith Ackermann, Carol Strohecker, and Aseem Agarwala. AnimMagix was developed between 1997 and 1998 by Edith Ackermann, Carol Strohecker, Aseem Agarwala, Adrienne Slaughter, and Dan Gilman. Zyklodeon was developed between 1999 and 2000 by Carol Strohecker, Adrienne Slaughter, Mike Horvath, Jack Kwok, Noah Appleton, Jeffrey Henrikson, Beatrice Witzgall, and Nadir Ait-Laoussine. The Kit4Kits was developed between 1999 and 2000 by Adrienne Slaughter and Carol Strohecker. The efforts are supported by Mitsubishi Electric Research Laboratories.