MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Approaches to Processes of Constructing in Software Construction Kits

Carol Strohecker, Adrienne H. Slaughter

TR2000-28 December 2000

Abstract

A shorter version of this paper appears in Proceedings of the International Workshop on Applied Learning Technologies, IEEE Computer Society Press, 2000 (MERL WP2000-07). We have developed a genre of software construction kits and a framework for implementing them. The framework is both conceptual and structural. Its conceptual aspect derives from constructivist learning theory, and its structural aspect extends the Java Abstract Windowing Toolkit. This framework, called the K̈it4Kits,s̈upports generation of software kits that are highly graphical and highly interactive. They are characterized by two main processes: playerśbuilding of objects from graphical elements, and the softwareś activation of the constructions. Five existing kits demonstrate a range of techniques for constructing objects. Additional techniques have become apparent as users of the framework created their own kits. We review these results and discuss various techniques for constructing graphical, dynamic, two-dimensional objects in software tools for learning.

Proceedings of the International Workshop on Applied Learning Technologies

MITSUBISHI ELECTRIC RESEARCH LABORATORY

Approaches to Processes of Construction in Software Kits

Carol Strohecker and Adrienne H. Slaughter

TR2000-28 June 2000

A shorter version of this paper appears in
Proceedings of the International Workshop on Applied Learning Technologies,
IEEE Computer Society Press, 2000 *(MERL WP2000-07).*

Abstract

We have developed a genre of software construction kits and a framework for implementing them. The framework is both conceptual and structural. Its conceptual aspect derives from constructivist learning theory, and its structural aspect extends the Java Abstract Windowing Toolkit. This framework, called the "Kit4Kits," supports generation of software kits that are highly graphical and highly interactive. They are characterized by two main processes: players' building of objects from graphical elements, and the software's activation of the constructions. Five existing kits demonstrate a range of techniques for constructing objects. Additional techniques have become apparent as users of the framework created their own kits. We review these results and discuss various techniques for constructing graphical, dynamic, two-dimensional objects in software tools for learning.

Approaches to Processes of Construction in Software Kits

Carol Strohecker
MERL - Mitsubishi Electric Research Lab
stro@merl.com

Adrienne H. Slaughter
Stanford University
ahs@alum.mit.edu

Abstract

We have developed a genre of software construction kits and a framework for implementing them. The framework is both conceptual and structural. Its conceptual aspect derives from constructivist learning theory, and its structural aspect extends the Java Abstract Windowing Toolkit. This framework, called the "Kit4Kits," supports generation of software kits that are highly graphical and highly interactive. They are characterized by two main processes: players' building of objects from graphical elements, and the software's activation of the constructions. Five existing kits demonstrate a range of techniques for constructing objects. Additional techniques have become apparent as users of the framework created their own kits. We review these results and discuss various techniques for constructing graphical, dynamic, two-dimensional objects in software tools for learning.

1. Introduction

We are developing a series of software kits based on the notion of "microworlds" [7] and the theory of "constructionism" [4, 5]. In this view, people construct rather than acquire knowledge, actively inventing ideas for themselves. Idea invention (or knowledge construction, or *learning)* is based on internalization of actions and experiences in the world [3]. Therefore the nature of particular activities becomes interesting, and activity design has become a specialization in learning research. Many of the designs find broader application in real-world domains such as toys, puzzles, and software [e.g., 8].

Considerations in activity design and interaction design are guiding development of our software construction kits. They form a genre in which end-users build and activate 2D graphical objects [11, 15]. Dinosaur skeletons balance as they walk and run [10]; maps transform into street-level views [12, 14]; colorful tiles spread into geometric patterns [2]; animistic creatures simulate the push-pulls of social dynamics [1, 2]; and dancers' breathing rates form a cycle for a shared dance [17].

Figure 1. Five kit prototypes demonstrate varying approaches to building 2D graphical, dynamic objects.

These kits focus on subject domains as varied as geometry, symmetry, physical forces, mechanical structures, time/space relationships, and system dynamics; yet they incorporate common strategies in activity design and interaction design. We are currently formulating generalizations of the strategies and programming constructs to support production of further instances of the genre. The resulting Java framework, called the "Kit4Kits," is both conceptual and structural [15, 16].

The Kit4Kits is comparable to systems like Microworlds, Cocoa, Agentsheets, ToonTalk, Squeak, and other tools for developing simulations and game-like learning environments.* However, we pay particular attention to notions of conceptual elements and operations as characterizers of an epistemological domain [7, 13]. We also tend specifically to computational supports for image treatments such as transparencies, filters, and gradients.

In our kit designs, and in kits that others have developed using the Kit4Kits, we have noted interesting variations and outstanding problems pertaining to a key facet of interaction design for the genre. Primarily, the kits support players' constructions of graphical objects, which upon activation become animate in some way. Players effect the constructions through direct manipulation of graphical elements, but the manner of access and assembly of the elements varies from kit to kit.

Here we review varying construction techniques for our existing prototypes, report on additional construction techniques developed by trial users of the Kit4Kits, and identify considerations for further construction techniques.

2. Existing Prototypes

2.1 Bones

In the *Bones* kit, the player creates skeletons by dragging individual bones into the work area and arranging them into the form of a dinosaur. The player can then animate the construction.

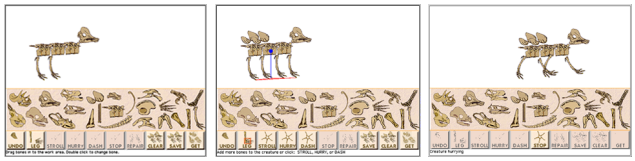

Figure 2. *Bones* players assemble parts into skeletons and then animate the creature. The software locates the construction's center of mass in order to determine whether the creature can balance as it moves.

In the first version of the prototype [10], clicking any of the movement buttons ("stroll," "hurry," or "dash") triggered several calculations. The program compared upper and lower portions of the composition and made guesses about which bones constituted the skeleton's legs. Then the program compared the combined mass values of the bones in the upper portion of the construction to those in the lower portion, and if the upper portion was too heavy the skeleton collapsed. Finally the program calculated the location of the skeleton's overall center of mass and illustrated it with a line projecting downward. If the line fell within a polygon connecting the points of contact with the "ground", the creature was deemed balanced and it proceeded to move, its legs swinging according to a gait pattern appropriate to the speed and the number of legs. If the line fell outside of base polygon, the skeleton collapsed.

Unfortunately the *Bones* algorithm could not always decide correctly which pieces constituted the legs, so some peculiar animations resulted. We made a revision in the current version of the prototype, such that designating the legs is part of the construction process. This ensures that the algorithm has the proper number and locations of legs, but shifts a burden to the player, whose freeform construction process is now encumbered by the specification process prior to seeing the animation.[1]

Figure 3. Prior to animating, *Bones* players identify which parts constitute the creature's legs. Legs can have several parts but their movement is articulated only at the hip. Any of the bones in the kit can be used as leg parts.

[1] We also extended the center-of-mass calculation and the set of gait patterns, so that creatures' legs now swing according to a pattern appropriate to the speed, the number of legs, and frontward or backward location of the center of mass. In a future version we hope to include articulated legs and perhaps spines, necks, etc., which would improve the animations but might necessitate further changes to the construction process.

The trade-off benefit is that any of the bones can be used anywhere in the skeleton. For example, the fanciful creature at the right, above, is composed of just three kinds of bones: skull, pelvis, and digit. (Skulls form the "thighs," digits form the "ribs," and so on.) Players can invent whimsical creatures or match creations to textbook illustrations of dinosaurs: the set of bones is based on parts found in reference books on paleontology. This flexibility would be lost if we pre-designated a part strictly as a head bone, a pelvis, vertebra, or etc., though such designations could simplify the construction process.

2.2 WayMaker

In the *WayMaker* kit we provide specific parts for building city layouts, but also allow for their further specification. The player arranges representations of districts, edges, paths, landmarks, and nodes into the form of a map, and the software generates street-level views along pathways through the mapped domain while maintaining the relative placements of the elements [12, 14]. However the elements are represented abstractly: landmarks are triangles, paths are dotted lines, and so on. The player can substitute more detailed representations: triangles can become towers, bridges, houses, etc.; lines can take on the look of textured terrain, etc.

Figure 4. *WayMaker* players assemble elements of the city image into a map and then trigger frame-by-frame displays of views along the pathways.

This construction approach poses benefits for both the player and the algorithm: the player enjoys freeform placement of the elements in shaping a map, and the pre-designation of elements into structural types simplifies the algorithm's handling of the elements as it transforms the construction. The extra step of specifying representations does not seem to be a burden for players: most prefer seeing a picture of a tower to an abstract symbol like a triangle, and they seem to enjoy making the selections.

Other kits constrain the construction process within a grid-like structure, to guide the player's building process and facilitate the software's handling of elements and constructions.

2.3 PatternMagix

In *PatternMagix* a four-part grid supports exploration of geometric symmetries as players reflect tiles around the x- and y-axes and rotate tiles within quadrants [2].

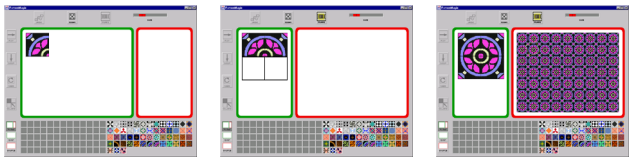

Figure 5. *PatternMagix* players experiment with reflections and rotations. The software replicates a resulting tile, and surprising patterns emerge.

2.4 AnimMagix

In *AnimMagix* a tripartite column guides assembly of animistic creatures' perceptual, social, and mobile behaviors [1, 2]. Sliders enable further adjustments, such as to the degree of a behavior. Perceptual fields can be deep as well as broad, attraction can be strongly or mildly positive or negative, and sweeping movements can be slow or fast.

Figure 6. *AnimMagix* players work within a tripartite column to specify ways in which creatures will interact with one another.

This manner of construction is familiar from toys, books, and other media. It constrains the construction process but has the advantages of providing pre-established designations for the algorithm and helping to clarify how the player should go about making a construction.

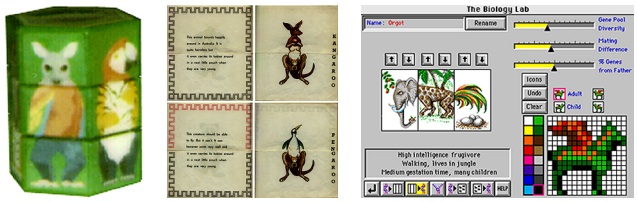

Figure 7. Other playthings make use of a similar manner of construction. [2]

2.5 Zyklodeon

We are employing a similar technique for a prototype now in progress, *Zyklodeon,* in which players create humanistic figures and endow them with properties that effect timing for a shared dance [17]. Dancers comprise six parts: head, torso, arms, and legs. Changing from a default part to a more colorful representation is similar to element specification in *WayMaker,* though the overall construction process is simpler because the defaults are already in place. In *Zyklodeon* we add a third tier to the construction process: within the torso are slider-controlled settings, like those in *AnimMagix,* with which the player can adjust a dancer's breathing rate and other choreographic parameters.

Figure 8. *Zyklodeon* players replace default elements and specify parameters characterizing dancers' movements.

Thus our existing prototypes exemplify a range of construction strategies: freeform construction, freeform construction with a specification phase, structured construction, and structured construction with varying levels and manners of further specification. What remains constant from one prototype to the next is the importance of the relationship between the build and activate processes, which typically plays out as an alternating pattern, usually with greater player control in the building and greater algorithm control in the activating.

Acknowledgment of this pattern led us to create separate structures for the two functions within the Kit4Kits. The Composer and Arena structures identified the nature of the activity within a specific screen area. Composers typically handled building elements; Arenas handled constructions and the associated algorithms that activated them. We explored the usefulness of these structures with several kit creators.

3. Kits by Initial Users of the Kit4Kits

3.1 Abacaudio

Alex wanted to make a kit with which players could create timing relationships in the context of music-making. Ball and soundpad elements would be paired such that a ball falling on a soundpad would make a sound, which could be specified as a particular tone. Building would consist of adding paired ball/soundpad elements to the Composer. Upon activation, each ball would strike its soundpad, and the Arena would display the strike patterns in a graph-like notation resembling a musical score. The patterns could be saved for replay.

Most notable about Alex's design is that, as in *Bones,* the build and activate processes share a screen area but constitute quite distinct activities. Thus our Composer and Arena constructs were suitable for his design.

---

[2] Fig. 7, left: *Animal Twister,* Club Earth, Cumberland, RI. Middle: J. Riddell, *Hit or Myth: More Animal Lore and Disorder,* Harper and Rowe, NY, 1949. Right: K. Karakotsios et al., *SimLife: The Genetic Playground,* Maxis, Orinda, CA, 1992.

3.2 WordBuilder

Max and Jan began a kit with which players can build letter combinations into phonemes, and phonemes into words. The player progresses downward through a process of word building: letters combine to form phonemes, which become syllables that form words. Letters must match according to particular sonority rules in order to form a phoneme [6]. Matches are saved into pockets ordered according to proper position of the phoneme within a word: an onset phoneme combines with a vowel to begin a word, which ends with a coda syllable. Saved words may or may not yet appear in an English dictionary, but must follow the onset-vowel-coda pattern.

Figure 9. The *WordBuilder* design evolved through several arrangements of screen areas and corresponding work flow.[3]

Max and Jan carefully separated the screen areas according to each of these functions, yet the main areas support both building and a kind of activating, which takes the form of checking for proper letter matches and syllable patterns. Nevertheless Jan implemented both areas by extending our Composer structure, rather than using the Composer for one and the Arena for the other.

3.3 Bugs

Chris wanted to make a simulation kit that would deal with notions of ecology. He wanted players to be able to control aspects of the environment, which resembles an ant farm, and creatures that inhabit it, which he called "bugs." He separated the two modes into screen displays that differed somewhat but also contained constant features.

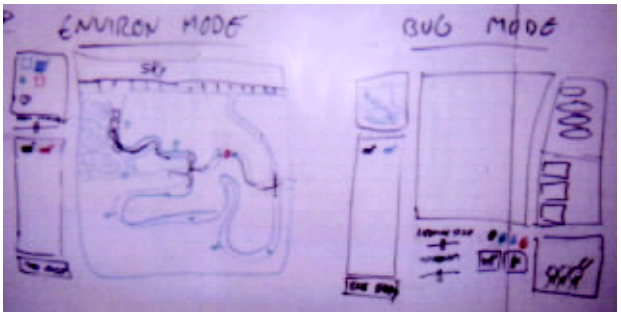

Figure 10. Each of the two modes in *Bugs* includes both build and activate processes.

In environment mode, the player can add bugs and food while the simulation is running. This is a kind of constructing, since the environment becomes more elaborate, but it is also a kind of activating, since behaviors play out over time.

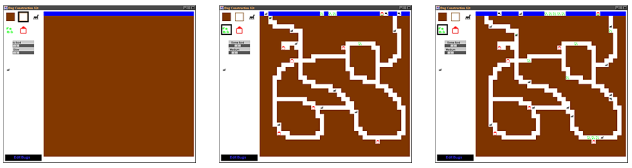

Figure 11. The environment mode in *Bugs*

In bug mode, the player can specify rules governing bugs' properties and behaviors, such as being hungry, seeking or avoiding food, seeking or avoiding other bugs, seeking food stores, dying when hungry and not finding food, and so on.

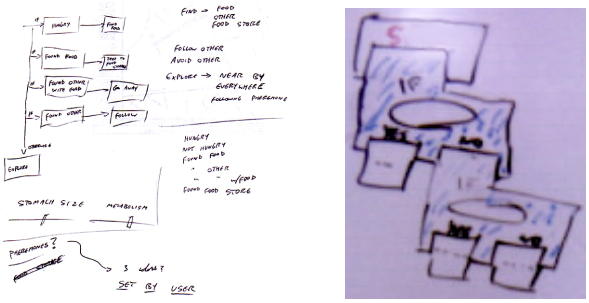

Figure 12. Sketches for guiding bug-building in *Bugs*

At first Chris represented the rule structure as a kind of logical chart, but through discussion he moved to more graphical representations of the settings.

[3] Graphical letterforms are from [9].

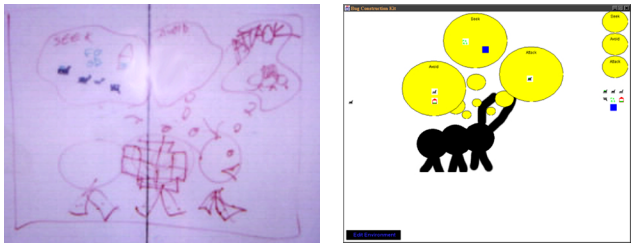

Figure 13. The bug mode reinterpreted

This notion of building is similar to the specification phase of building in our prototypes. Chris implemented this functionality by extending the Composer structure.

4. A Typology of Construction Techniques

By working with these kit creators, we realized that the Composer and Arena structures initially in the Kit4Kits overspecified notions of building and activating. Now a more general Zone structure allows for varying notions of building and activating, which may be interrelated in some designs.

Experiences with our kits and those devised by Kit4Kits users has called attention to different kinds of building that learning environment designers may want to support. Including different kinds of building may be a good way to address different learning and thinking styles.

Structures that guide building can be useful when constructions within a kit take a consistent form. Phased construction may include sub-processes such as specification of behavioral properties and image details.

While providing a set of building elements inevitably constrains what players can make with a given kit, interaction designs may range from encouraging recipe-style production of particular constructions to freeform building that relies on the player's creativity. We prefer open-ended building in which players produce novel constructions but note that younger builders and builders of complex constructions may benefit from guided processes such as construction grids and phases. Other interesting possibilities include reversing a construction process for an existing object and adjusting or completing a partially started construction process.

5. Acknowledgments

We owe particular thanks to Edith Ackermann and Aseem Agarwala, who made key contributions in originating the Composers and for the Magix kits. They also contributed significantly in other ways to the designs of these kits. Several other people have also contributed to design, development, and use of Magix and the other kits informing our framework: AARCO medical illustrators, William Abernathy, Noah Appleton, Maribeth Back, Barbara Barros, Dan Gilman, Mike Horvath, John Shiple, Doug Smith, students at Harvard University's Graduate School of Design, colleagues at MERL, and anonymous friends. We thank John Evans, Aradhana Goel, Tim Gorton, and Milena Vegnaduzzo for participating in trials of the Kit4Kits. MERL supports the research.

6. References

[1] E. Ackermann and C. Strohecker, "Interaction design for AnimMagix prototype", TR98-13, Mitsubishi Electric Research Lab, Cambridge, MA, 1998.

[2] E. Ackermann and C. Strohecker, "Build, launch, convene: Sketches for constructive-dialogic play kits", TR99-30, Mitsubishi Electric Research Lab, Cambridge, MA, 1999.

[3] H. E. Gruber and J. J. Vonèche (eds.), *The Essential Piaget,* Basic Books, New York, 1977.

[4] I. Harel and S. Papert (eds.), *Constructionism,* Ablex, Norwood, NJ, 1991.

[5] Y. Kafai and M. Resnick (eds.), *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World,* Lawrence Erlbaum, Mahwah, NJ, 1996.

[6] M. Kenstowicz, *Phonology in Generative Grammar,* Blackwell, Cambridge, MA, 1994.

[7] S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books, New York, 1980.

[8] Papert et al., http://el.www.media.mit.edu/.

[9] H. A. Rey, *Curious George Learns the Alphabet,* Houghton Mifflin, Boston, 1963, 1991.

[10] C. Strohecker, "Embedded microworlds for a multiuser environment", TR95-07, Mitsubishi Electric Research Lab, Cambridge, MA, 1995.

[11] C. Strohecker, "Construction kits as learning environments", *Proceedings of IEEE International Conference on Multimedia Computing and Systems* 2, 1999, pp. 1030-1031.

[12] C. Strohecker, "Toward a developmental image of the city: Design through visual, spatial, and mathematical reasoning", *Proceedings of Visual and Spatial Reasoning in Design,* University of Sydney and Massachusetts Institute of Technology, 1999, pp. 33-50.

[13] C. Strohecker, "Why knot?", Ph.D. diss., Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group, 1991.

[14] C. Strohecker and B. Barros, "Make way for WayMaker", *Presence: Teleoperators and Virtual Environments 9:*1, 2000, pp. 97-107.

[15] C. Strohecker and A. Slaughter, "Kits for learning and a kit for kitmaking", *CHI'00 Extended Abstracts,* 2000.

[16] C. Strohecker and A. Slaughter, "A Framework for Microworld-style Construction Kits", TR2000-19, Mitsubishi Electric Research Lab, Cambridge, MA, 2000.

[17] C. Strohecker, A. Slaughter, and M. Horvath, "Zyklodeon Interaction Design", WP2000-03, Mitsubishi Electric Research Lab, Cambridge, MA, 2000.

* See www.lcsi.com, www.crim.ca/~hayne/Cocoa/, www.agentsheets.com, www.toontalk.com, www.squeak.org.