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

Designing Building Processes in Software Construction Kits

Carol Strohecker, Adrienne H. Slaughter

TR2000-03 December 2000

Abstract

We have developed a genre of software construction kits and a framework for developing them, which is both conceptual and structural. The kits are highly graphical and highly interactive. They are characterized by two main processes: playerś building of objects from graphical elements, and the softwareś activation of the constructions. The existing kits demonstrate a range of interaction designs for creating constructions, and trial users of our framework have introduced further approaches. We review these results and identify considerations for articulation of optimal construction techniques.

MITSUBISHI ELECTRIC RESEARCH LABORATORY

Designing Building Processes in Software Construction Kits

Carol Strohecker
Adrienne H. Slaughter

TR2000-03 April 2000

Abstract

We have developed a genre of software construction kits and a framework for developing them, which is both conceptual and structural. The kits 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. The existing kits demonstrate a range of interaction designs for creating constructions, and trial users of our framework have introduced further approaches. We review these results and identify considerations for articulation of optimal construction techniques.

Designing Building Processes in Software Construction Kits

Carol Strohecker
Adrienne H. Slaughter
Mitsubishi Electric Research Laboratory (MERL)
201 Broadway, Cambridge, MA 02139 USA
+1 617 621 7517, +1 617 621 7594
stro@merl.com, slaughter@merl.com

ABSTRACT
We have developed a genre of software construction kits and a framework for developing them, which is both conceptual and structural. The kits 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. The existing kits demonstrate a range of interaction designs for creating constructions, and trial users of our framework have introduced further approaches. We review these results and identify considerations for articulation of optimal construction techniques.

Keywords
construction kits, Java framework, learning, interaction design, activity design

INTRODUCTION
We are developing a series of software kits based on the notion of "microworlds" [7] and the theory of "constructionism," which holds that people construct rather than acquire knowledge, inventing ideas for themselves based on actions in the world [4, 5]. Because actions are so important in knowledge construction, the nature of particular activities becomes especially interesting, and activity design has become a specialization in learning research [3]. Many of these designs find broader application in real-world domains such as toys, puzzles, and software [e.g., 8].

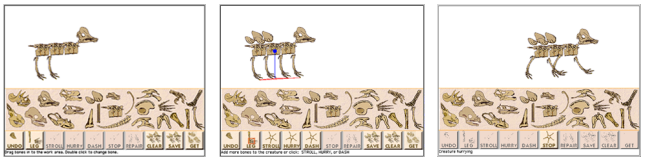
Considerations in activity design and interaction design are guiding development of our software construction kits. They form a genre in which users (variously called "learners" or "players") build and activate graphical objects [17]. Dinosaur skeletons balance as they walk and run [10, 11, 12]; maps transform into street-level views [12, 13, 14, 15, 16]; 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].

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. Our Java framework, called the "Kit4Kits," is both conceptual and structural [17].

In our kit designs, and through work with people employing the Kit4Kits to implement their own kit designs, we have discovered 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 processes for our existing prototypes, report on additional construction techniques developed by trial users of the Kit4Kits, and identify considerations for articulation of optimal construction techniques.

EXISTING PROTOTYPES
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 click a button to animate the construction.

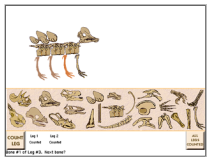


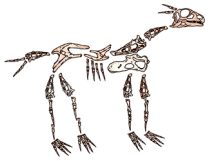
In the first version of the prototype [10, 11], clicking any of the movement buttons ("stroll," "hurry," or "dash") triggered several calculations:

- the program compared upper and lower portions of the composition and made guesses about which bones constituted the skeleton's legs;

- 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;
- 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 [12], 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]





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.

WayMaker

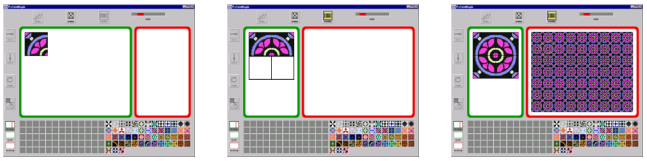
In the *WayMaker* kit we use similar designations but also allow further specification of elements. The player arranges representations of districts, edges, paths, landmarks, and nodes into the form of a map, and the software generates street-level views along pathways through the mapped domain while maintaining the relative placements of the elements [12, 13, 14, 15, 16]. 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.

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. Furthermore, 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 choosing the specification is part of the fun of using the software.

Other kits constrain the construction process within a grid-like structure.

PatternMagix

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



AnimMagix

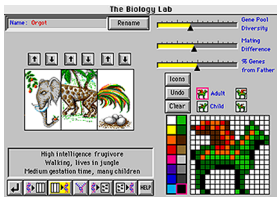
In *AnimMagix* a tripartite column guides assembly of animistic creatures' perceptual, social, and mobile behaviors [1, 2]. Sliders enable further adjustments, such as to the degree of a behavior.

The manner of construction is familiar from toys, books, and other media.[2] 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.

[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. There is also a cursory representation of dynamic balance for the faster speeds. In a future version we hope to include articulated legs and perhaps spines, necks, etc., which would improve the animations but may further complicate the construction process.

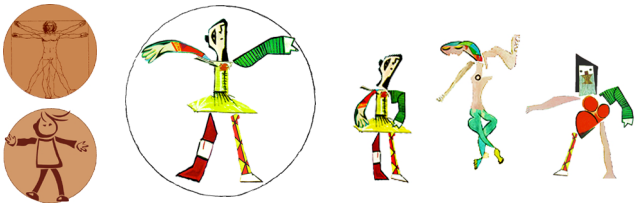
[2] Left: *Animal Twister,* Club Earth, Cumberland, RI. Middle: J. Riddell, *Hit or Myth: More Animal Lore and Disorder,* Harper and Rowe, NY, 1949. Right: K. Karakotsios et al., *SimLife: The Genetic Playground,* Maxis, Orinda, CA, 1992.





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 [18]. Dancers comprise six parts: head, torso, arms, and legs. Changing from a default part to a more colorful representation is similar to the element specification process in *WayMaker*. In *Zyklodeon* we add a third tier to the construction process: within the torso are pop-up, slider-controlled settings with which the player can adjust a dancer's breathing rate and other choreographic parameters.



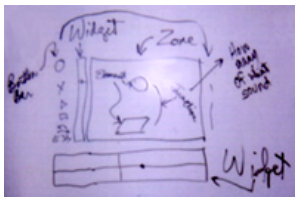
Thus our existing prototypes exemplify a range of construction strategies: freeform construction, freeform construction with a specification phase, and structured construction with varying levels and manners of further specification. What remains constant from one prototype to the next is the importance 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 identify the nature of the activity within a specific screen area. Composers typically handle building elements; Arenas handle constructions and the associated algorithms that activate them.

DESIGNS BY TRIAL USERS OF THE KIT4KITS

Abacaudio

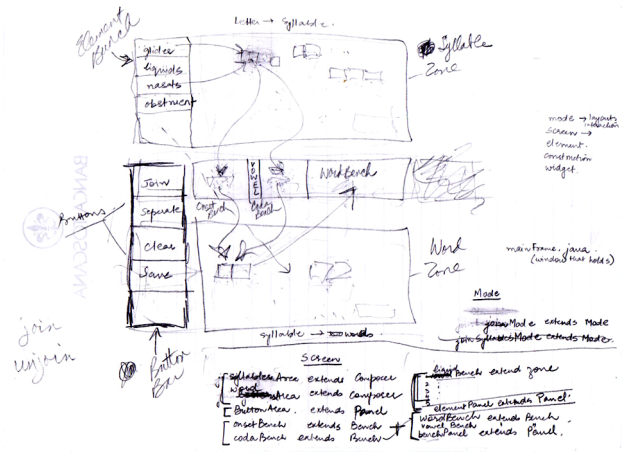
Alex wanted to make a kit with which players could explore timing relationships in the context of music-making. Ball and soundpad elements would be paired such that a ball falling on a soundpad would make a sound, which could be specified as a particular tone. Building consists of adding ball/soundpad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

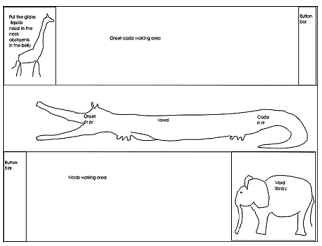
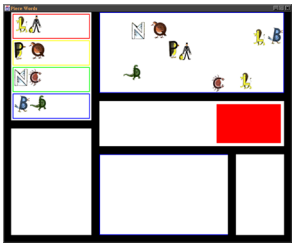



Most striking about Alex's design is that, as in *Bones,* the build and activate processes share a screen area but constitute quite distinct activities. Thus our Composer and Arena construct would be well suited to his design.

WordBuilder

Max and Jan began a kit with which players can build letter combinations into phonemes, and phonemes into words. Their design evolved through several arrangements of screen areas and corresponding work flow:[3]



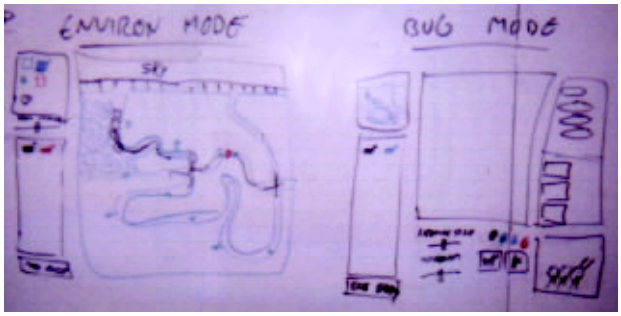
Eventually they settled on an arrangement based on downward movement as the player progresses through a process of word building: letters combine to form phonemes, which become syllables that form words. Letters must match according to particular sonority rules in order to form a phoneme [6]. Matches 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.

[3] Graphical letterforms are from [9].

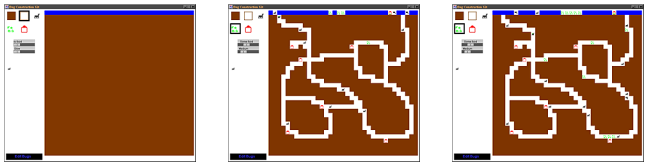
Max and Jan carefully separated the screen areas according to each of these functions, yet the main areas support both building and a kind of activating, which takes the form of checking for proper letter matches and syllable patterns. Nevertheless Jan implemented both areas by extending our Composer structure, rather than using the Composer for one and the Arena for the other. Composers typically handle operations on elements like the Jan's validity checking, so it is curious that this process comes closest to a notion of activation in WordBuilder.

Bugs

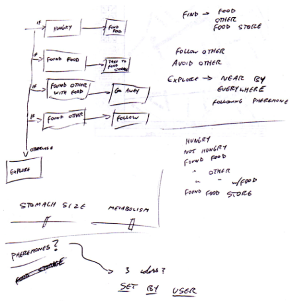
Chris wanted to make a simulation kit that would deal with notions of ecology. He wanted players to be able to control aspects of the environment, which resembles an ant farm, and of creatures that inhabit it, which he called "bugs." He separated the two into screen displays that differed somewhat but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

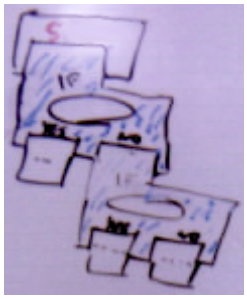


Interestingly, both modes include both build and activate processes. In environment mode, the player can add bugs and food for the bugs while the simulation is running. This capability differs noticeably from the usual separation of building and activating within our kit genre.

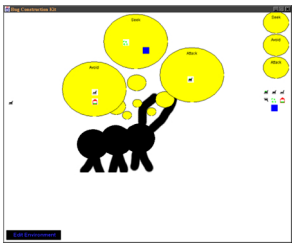


He wanted the player to be able to specify rules governing bugs' properties and behaviors, such as being hungry, seeking or avoiding food, seeking or avoiding other bugs, seeking food stores, dying when hungry and not finding food, and so on.





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



Interestingly, Chris's notion of build mode is more like the specification phase of building in our prototypes. He implemented this specification functionality by extending our Composer structure.

OPTIMAL CONSTRUCTION TECHNIQUES

We need to clarify the notions of building and activation with respect to their pertinence to elements and/or constructions. In particular, we need to make more explicit the notion of element specification as a sub-process within building. These clarifications should help users of the Kit4Kits to deal more easily with the existing Composer and Arena structures. Meanwhile, distinctions between varying notions of building and activating are informing our ongoing development of the Kit4Kits.

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.

REFERENCES

1. Ackermann, E., and Strohecker, C. Interaction design for AnimMagix prototype. MERL TR98-13, Mitsubishi Electric Research Laboratory, Cambridge, MA, 1998.
2. Ackermann, E., and Strohecker, C. Build, launch, convene: Sketches for constructive-dialogic play kits.

MERL TR99-30, Mitsubishi Electric Research Laboratory, Cambridge, MA, 1999.

3. Gruber, H. E., & Vonèche, J. J. (eds.). *The Essential Piaget.* Basic Books, New York, 1977.
4. Harel, I., & Papert, S. (eds.). *Constructionism.* Ablex, Norwood, NJ, 1991.
5. Kafai , Y., and Resnick, M. (eds.) *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World.* Lawrence Erlbaum, Mahwah, NJ, 1996.
6. Kenstowicz, M. Phonology in Generative Grammar. Blackwell, Cambridge, MA, 1994.
7. Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York, 1980.
8. Papert et al., http://el.www.media.mit.edu/.
9. Rey, H. A. Curious George Learns the Alphabet. Houghton Mifflin, Boston, 1963, 1991.
10. Strohecker, C. Embedded microworlds for a multiuser environment. MERL TR95-07, Mitsubishi Electric Research Laboratory, Cambridge, MA, 1995.
11. Strohecker, C. A model for museum outreach based on shared interactive spaces. *Multimedia Computing and Museums: Selected Papers from the Third International Conference on Hypermedia and Interactivity in Museums,* Archives & Museum Informatics, Pittsburgh, 57-66*,* 1995.
12. Strohecker, C. Construction kits as learning environments. *Proceedings of IEEE International Conference on Multimedia Computing and Systems* 2, 1030-1031, 1999.
13. Strohecker, C. 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, 33-50, 1999.
14. Strohecker, C., and Barros, B. A prototype design tool for participants in graphical multiuser environments. *CHI'97 Extended Abstracts,* 246-247, 1997.
15. Strohecker, C., and Barros, B. Make way for WayMaker. *Presence: Teleoperators and Virtual Environments 9:1,* 97-107, 2000.
16. Strohecker, C., Barros, B., and Slaughter, A. Mapping psychological and virtual spaces, *International Journal of Design Computing,* University of Sydney, 1998.
17. Strohecker, C., and Slaughter, A. Kits for learning and a kit for kitmaking. Submitted to *CHI'00*. Also available as MERL TR2000-02, Mitsubishi Electric Research Laboratory, Cambridge, MA, 2000.
18. Strohecker, C., Slaughter, A., and Horvath, M. Mitsubishi Electric Research Laboratory, Cambridge, MA, forthcoming.