Towards Construction Kits for Virtual World Artifacts

Kenneth D. Forbus
Institute for the Learning Sciences
1890 Maple Avenue
Evanston, IL, 60201, USA
Email: forbus@ils.nwu.edu
Fax: 847-491-5258

Abstract: Playing with physical construction kits (Legos, Erector sets, Fischer-Technik, etc.) is a good way to build physical intuitions and spark interest in technical fields. Construction kits for virtual artifacts could have similar benefits, with complementary advantages and disadvantages. Yet we currently do not have construction kits for virtual analogs of physical artifacts that provide the same engagement as physical kits. This essay examines why, and some of the technical barriers that must be overcome to create such systems.

Introduction

Construction kits, such as Erector sets, Legos, Fischer-Technik, and many others, provide children (and adults) with many hours of enjoyment. Such kits are also valuable tools for learning and teaching. Building, modifying, and interacting with their own creations enables students to experience, in a safe and constrained environment, how a broad variety of physical things work. Construction kits in virtual worlds could serve similar goals. Like physical construction kits, construction kits for virtual artifacts could enable students to build, operate, and experiment with systems that operate according to complex physical principles. Virtual artifacts can be cheaper, safer, and less time consuming to deal with than physical artifacts: No one would suggest that students learn thermodynamics by building and testing jet aircraft engines. On the other hand, while creating a "virtual blocks set" might be a good way to learn certain programming ideas, a set of physical blocks is likely to be better for learning how to build elementary structures. The educational power of construction kits for virtual artifacts is likely to be highest in situations where the real thing is too dangerous, too expensive, too intricate, unobtainable, or at the wrong time scale for the learner's situation. Many of today's scientists and engineers have fond memories of such construction kits from their childhood; perhaps by creating the right kinds of construction kits for virtual artifacts we can expand participation in science and engineering to a wider segment of the population.

Despite this potential, there are few successful construction kits for physically inspired[1] virtual artifacts available today. Moreover, none of them seem as successful as physical construction kits. This essay examines why, and examines some of the technical barriers that must be overcome to make such kits more compelling for learners.

Limitations of today's virtual artifact construction kits

Roughly, there are four kinds of virtual artifact construction kits available today:

1. *Game customization software*. These programs enable players to customize the underlying simulation engine of a computer game, changing the properties of the objects or characters, thresholds, or even adding new types of entities. Examples are legion, some commercial (e.g., the SimCity 2000 Urban Renewal Kit), but most being shareware created by enthusiasts (e.g., editors for Quake, Civilization, etc.).

[1] Arguably, any compiler or interpreter is a construction kit for a crucial kind of virtual artifact: software. Our focus in this essay is on simulation environments inspired by physical world artifacts.

2. *Simulation authoring environments*. These environments enable students to create new simulations, rather than just customizing a given simulation/game engine. Examples include Agentsheets [1] Stella [2], and Star-Logo [3].
3. *Special-purpose programming environments*. These environments enable students to create new instances of specific types of software, typically games. Examples include Klick 'n Play [4], and the (now unavailable) Pinball Construction Kit
4. *Articulate Virtual Laboratories.* These educational environments provide students with the ability to design and analyze complex systems, focusing on the ability to provide explanations coaching [5,6]. Examples include CyclePad [7] and the Feedback Articulate Virtual Laboratory (FAVL) [8].

Based on our experience with these programs, and discussions with others about them, one can summarize the limitations on the artifacts one can create with them as follows:

- *Can't play with it anywhere else.* Artifacts built in game customization software work only in the original gaming environment. Simulation authoring environments and special-purpose programming environments are aimed more at producing self-contained systems rather than virtual artifacts that can be integrated into larger worlds. Articulate virtual laboratories are intended to provide virtual artifacts for integration into larger-scale simulations, e.g., power plants for SimCity, or engines for aircraft/spacecraft simulations, but this capability has not been developed yet for reasons outlined below.

- *Can't play with it in several ways.* With Lincoln Logs one can build a garage for toy cars, and see what happens when the toy cars run into it, or a dog runs into it, and so on. The range of behaviors supported in today's virtual artifacts is far narrower than for physical artifacts. In many ways this can be an advantage, because it enables world designers to suppress unimportant or unenlightening details. However, it means that alternate routes for engagement must be found to make the world compelling for students.

- *Can't figure out what is going on.* Even though simulations simplify, their generativity means that students can easily create situations where they cannot interpret the results, given their limited expertise. Providing explanations and assistance with the novel artifacts that construction kits can create presents new challenges for the design of software coaches and tutors.

- *Can't figure out what to do.* A blank workspace on screen can be just as intimidating as a blank piece of paper, or a large pile of unfamiliar parts on the floor. Booklets and posters illustrating what kinds of things one might build successfully within a given system, while not specifying step-by-step how to build them, are a common trick used in physical construction kits. Organizing similar libraries of artifacts, that highlight interesting possibilities while preserving the joy of discovery, is an interesting challenge for designers.

The rest of this essay explores some technical issues involving virtual artifact construction kits. The first set of issues, concerning exporting and embedding virtual artifacts, address the first two problems (i.e., how to play with your artifact in different places and in different ways). The second set of issues, concerning explanations and coaching, address the second two problems (i.e., how virtual worlds can be designed to scaffold students in creating and exploring virtual artifacts). We end by discussing some work in progress towards addressing these issues.

Exporting and embedding virtual artifacts

Here is a homework assignment that has never been seen in an engineering course:

> Develop a propulsion system for a single-stage to orbit (SSTO) spacecraft, based on one of the design families discussed in class. Airframe designs and other subsystem components can be found at
> http://www.cs.nwu.edu/academics/courses/froshdesign/SSTO/warehouse.html

> Turn in flight recorder data that demonstrates that your design can reach near-earth orbit and safely return, the rationale for your design choices, and three suggestions about how your design might be improved.

It is hard to find an engineering student who would not love to tackle this assignment -- assuming that they had enough support software to manage the complexity, background materials to explore, and a virtual testing environment where they could survive their failures. Alas, it will be some time before we can give out such assignments on a regular basis. There are several reasons for this (some of which are deferred to the section below). Articulate virtual laboratories can help students produce complex designs, including rich tools for analyzing their behavior. But giving them virtual environments where they can try out artifacts based on their designs is difficult. To maximize engagement, such environments much be rich, ideally as rich as the best hard-wired, hand-coded computer gaming simulations. The test environment must also support the use of imported simulations as components, so that the student's artifact(s) can be accommodated. Such environments need to be off-the-shelf, if the courseware development burden is to be manageable.

There are simple cases that should yield pedagogical benefits while being tractable with today's technology. For example, suppose that there is a single specific type of component to be inserted. Also suppose that the test environment uses a modular method for specifying that component (e.g., a declarative specification of the component's model, or a dynamic link library for the component simulation). Under these assumptions, the articulate virtual laboratory only needs to produce a piece of software in the appropriate format to "manufacture" the student's design. Unfortunately, this solution only allows the student to create and use a single component at a time. Suppose for instance that a student using CyclePad wanted to design both a power plant for a city simulation as well as an engine for the car they will tour their city in. A more flexible architecture that can support distributed simulations will be needed to provide the same ability to populate an environment with multiple student artifacts that the physical world provides.

Maintaining rich, believable interactive behavior in a simulator composed of a federation of other simulators is of course a difficult problem. The protocols and datastructures used for communication between simulators impose the physics (and biology) of the virtual world they define. It is obvious that richness can trade off against interactivity – a virtual world where the air avatars moved in was simulated via computational fluid dynamics would run so slowly that few visitors would find it attractive. Another tradeoff that seems important for virtual worlds for education is richness versus focus. The abstraction of the virtual worlds enables us to select which aspects of the physical world we want to highlight, to maximize potential learning. By simply leaving out those aspects of the world that are irrelevant, we reduce potential distractions for the student (as well as simplify development and optimize runtime resource usage). Unfortunately, enshrining decisions about focus in the design of the virtual world can preclude exploring issues outside that focus (i.e., limit the ways to play with the artifact). For example, … The strategy analogous to what is done in the physical world, i.e., operate in a rich world but filter what the student attends to, would certainly be useful for detecting unanticipated interactions, but could be wasteful of computational resources. It may be worth exploring architectures for virtual worlds that provide a lattice of environments, organized by what aspects of the most complex world they include. Initial explorations could be carried out in highly simplified environments chosen to stress particular aspects of a student's design. Richer environments would then be used to explore integration issues.

Explanation and coaching in virtual artifacts

No matter how immersive the environment one's virtual artifact is embedded in, the ways one interacts with it will not provide the same degree of engagement provided by the analogous actions on the equivalent physical artifact. An engine burnout in a simulated SSTO craft will not crater the physical landscape nor actually kill the pilot, at least in any responsibly designed virtual world. This reduced driver for engagement is simply part of the price one pays for the cost and safety advantages of a virtual world. One of the payoffs of virtual worlds is that they can be manipulated more easily than, and sometimes in ways impossible in, the physical world. With the right technology, we could provide different drivers that lead both to deeper engagement and to ways of engagement that better foster learning.

Consider again the scenario of (virtually) flying one's own SSTO design into near-Earth orbit. If the craft begins to shake uncontrollably, one could pause the simulation and ask why this is happening. An *articulate simulation* will be able to help the student explain how the currently observed behavior results from the interaction of the environment with their actions and design decisions. Rather than continuing a doomed mission, the student can choose to go back to the (virtual) laboratory to improve their design.

Articulate simulations require two capabilities:

1. *The ability to track dependencies.* The simulation must be able to identify to some degree why particular outcomes arise from particular inputs.
2. *The ability to communicate with users in conceptually natural terms.* The simulator must be able to generate explanations of the behavior that combine numerical and quantitative information with the kind of qualitative, conceptual information that people use to organize their thinking about the physical world.

Simulations can be designed with varying degrees of articulateness, of course. At minimum, the simulation should track how the physical phenomena it is reproducing is expressed in the simulation behavior. Richer levels of articulation include deeper understandings of the physical phenomena, and how it can tie into student's goals.

Building complex simulations and tuning them for accuracy is already hard enough that the additional demand for articulateness may seem unreasonable. Fortunately, there is a simulation technology that is useful for a broad class of systems, which provides a high degree of articulation as a side effect of automatically compiling simulators from general domain knowledge. *Self-explanatory simulators* [9, 10, 11] provide the speed and accuracy of traditional numerical state-space simulators, but also provide qualitative explanations. Self-explanatory simulators are constructed automatically by a compiler that uses techniques from qualitative physics to (a) identify the relevant physical phenomena in the situation, using an off-the-shelf domain theory, (b) identify the relevant mathematical models, and (c) write the simulation code based on these models. The reasoning the compiler performs for steps (a) and (b) is also used to create an explanation system incorporated with the simulator. By keeping track of a small amount of additional information during simulation, the simulator can explain the complete causal and mathematical account that held at any time during a simulated behavior, without any run-time qualitative reasoning.

The explanations provided by self-explanatory simulators relate properties of the observable behavior to the student's assumptions and the laws of the phenomena being simulated. While important, there are additional aspects of explanation that an expert human tutor could provide that could lead to deeper learning. Creating software tutors and coaches that could provide these services might radically improve the conceptual engagement with a student's constructions. First, the ability to recognize higher-order patterns of behavior (e.g., resonance modes of vibration) would enable coaches to point out qualitatively interesting outcomes and changes in categories of behavior that occur in response to changes in a student's design. Second, the ability to understand the student's goals for the artifact would enable coaching software to help the student evaluate how well their design fit their goals, and suggest improvements. Finally, the ability to link the specific representations for phenomena occurring in the simulated artifact to rich information sources of background knowledge could enable students to delve more deeply when their curiosity is aroused.

Discussion and work in progress

The challenge of creating construction kits for virtual artifacts that are as compelling as physical construction kits in enchanting students, as well as helping them learn, is complex. Progress on a variety of issues will help, including richer virtual worlds, federated simulation environments, and representation formalisms for web-based information that make it machine-friendly. Our group is working to make articulate virtual laboratories and self-explanatory simulators more useful, including

- *Embeddable self-explanatory simulators.* We are developing APIs for our Common Lisp, C++, and Java-based self-explanatory simulators, so that others can embed them in their software systems.

- *Self-explanatory simulators as the "manufacturing facility" of articulate virtual laboratories.* We are extending the thermodynamics domain theory so that we can compile self-explanatory simulators based on the steady-state designs produced by students using CyclePad.
- *Behavior recognizers for coaches.* We are creating a library of routines for representing and recognizing patterns of behavior in controlled systems, to be used by the coach for our feedback articulate virtual laboratory (FAVL).
- *Case-based coaching for design.* Using analogical processing techniques [12], we are implementing an email-based design coach that can make suggestions about a student's design, based on interesting examples supplied by domain experts. This coach relies on the automatic recognition of the intended function of a student's design [13]

We hope that these improvements will lead to a suite of software that will make learning science and engineering radically more fun.

Acknowledgments

This research is supported by the Defense Advanced Research Projects Agency, under the Computer Aided Education and Training Initiative (CAETI), the National Science Foundation under the Applications of Advanced Technologies Program, and the Office of Naval Research. We thank Will Wright, Julie Baher, and Joyce Ma for productive discussions.

References

1 http://www.cs.colorado.edu/homes/ambach/public_html/agentsheets_html/agentsheets_page.html
2 http://www.hps-inc.com/
3 http://starlogo.www.media.mit.edu/people/starlogo/
4 http://www.maxis.com/games/klik_n_play/
5 Forbus, K. and Whalley, P. (1994) Using qualitative physics to build articulate software for thermodynamics education. *Proceedings of AAAI-94*, Seattle.
6 Forbus, K. Using qualitative physics to create articulate educational software. *IEEE Expert*, **12**(3), May/June 1997.
7 http://www.qrg.ils.nwu.edu/projects/NSF/Cyclepad/cyclepad.htm
8 http://www.qrg.ils.nwu.edu/projects/NSF/avl.htm
9 Forbus, K. and Falkenhainer, B. "Self-explanatory simulations: An integration of qualitative and quantitative knowledge", AAAI-90, August, 1990.
10 Forbus, K. and Falkenhainer, B. 1995. Scaling up Self-Explanatory Simulators: Polynomial-time Compilation. Proceedings of IJCAI-95, Montreal, Canada.
11 http://www.qrg.ils.nwu.edu/software/software.htm
12 Forbus, K., Gentner, D., Everett, J. and Wu, M. 1997. Towards a computational model of evaluating and using analogical inferences. *Proceedings of CogSci97.*
13 Everett, J. A Theory of Mapping from Structure to Function Applied to Engineering Thermodynamics. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. Montreal, Quebec, Canada, 1995.