

Rewriting Interaction

Roland Hübscher

Department of Computer Science & Institute of Cognitive Science
University of Colorado at Boulder, Boulder, CO 80309-0430
(303) 492-4932
rolandh@cs.colorado.edu

ABSTRACT

Interactive visual computer animation is becoming an important tool for science education in grade school. Unfortunately, students and teachers cannot easily create their own animations, because programming these systems tends to be too hard for non-professional programmers. I present an approach that simplifies the description of complex interactions of objects by describing interactions with declarative, temporal constraints. A system that describes animation in terms of the *actions* of the objects and the *interactions* between the objects is being built on top of a grid-based, graphical programming environment.

KEYWORDS: visual animation, science education, visual programming, rewrite rules, temporal constraints.

INTRODUCTION

Visual animation can be used to display dynamic processes and are an important tool in grade school science education. Animation is even more useful if it can be programmed by the end users, that is, by the students and their teachers. Unfortunately, the complexity of programming animations can keep the end users from creating their own animation and thus, the interactional value of the tool is diminished.

The need for languages that allow grade-school students to create interesting, non-trivial animations is therefore of great importance. The domain of visual animation suggests to use visual programming languages, which would allow the end user to directly manipulate the animation instead of some text which

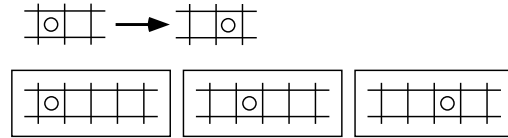


Figure 1: Above is a rewrite rule moving a ball to the right if the square to the right of the ball is empty. Below the rule are three snapshots of a ball animated by the rule.

is separate from the animation. Having a textual description of a visual animation has the problem of mapping the visual and textual concepts to each other.

VISUAL PROGRAMMING LANGUAGES

I will restrict myself to rule-based visual languages since they have proven to be useful expressing a wide variety of animations [1, 3]. A system based on rewrite rules consists of three components, namely the picture of the animation as seen by the viewer (the state of the computation), a collection of rewrite rules (the program), and a rule interpreter. A rewrite rule consists of a before- and an after-picture. The rule interpreter tests whether there is a rule whose before-picture describes a part of the animation picture. If the test is successful, the described part of the animation is replaced by the after-picture as is demonstrated in Figure 1.

ACTION AND INTERACTION

Rules describe what actions an object can execute. As long as it is clear what the actions of the objects are and which action is chosen in what situation, no problem arises. But this is not always so simple. An interaction involves several objects, and it is not always clear which of the objects acted on which other objects. For instance, when on the pool table the (white) cue-ball collides with the resting eight-ball, the cue-ball stops and the eight-ball accelerates as Figure 2 shows. Did the cue-ball accelerate the eight-

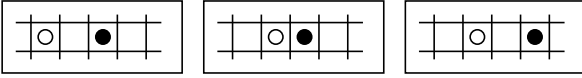


Figure 2: The white cue-ball collides with the black eight-ball.

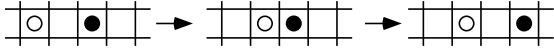


Figure 3: The temporal constraint describes the collision of the cue- and the eight-ball.

ball or did the eight-ball stop the cue-ball, or did both happen—and do we really want to make this decision anyway? A physicist would probably prefer the physically correct description of the collision process. The chances are, that he or she needs to figure out exactly how the balls react to each other which may require additional parameters about mass, impetus, spheres and so on. On the other hand, the user might only be interested in a description of the interaction of the two balls since, in an animation, all the balls have to do is behave correctly when bumping into each other.

A *temporal constraint* describes an interaction by constraining each state in a sequence of states. Constraining a state means that the constraint, a condition on the state, allows only certain states to be at this place in a sequence. There is no need to figure out the actions and conditions that make the interaction emerge from the actions of the individual objects. As the name *interaction* suggests, the interaction can be viewed as an action *between* the objects. Of course, the user is allowed to use just actions to describe an interaction, since this can sometimes be easier than describing a complex interaction resulting from simple actions. A good example for the latter case is the Braitenberg vehicles [2]. A temporal constraint (see Figure 3) describes a sequence of states (for instance, the ones in Figure 2) and looks similar to a rule. There are a few important differences, though. Whereas a rule *manipulates* the picture of the animation, a constraint *chooses* one of possibly many different picture sequences generated by the rules. Furthermore, a temporal constraint restricts an arbitrary long sequence of states, although they are rarely more than three states long, whereas a rule refers only to two states. An additional advantage is the declarative of constraints which allows the user to ignore implementation details.

The following analogy may help to make the differences of between temporal constraints and rewrite rules clearer. The possible states the human body can be in and how the body can go from one state

into another depends on the laws of physics and the body’s structure. This is captured by the rules that define the space of states and state transitions. If the person, that is, the owner of the body decides to use the body, for instance to reach for the coffee cup, then all the actions of many body parts must be coordinated. This is exactly what a temporal constraint does. By constraining the possible action sequences, only action sequences are allowed that result in the hand grasping the cup.

FUTURE WORK AND CONCLUSIONS

An implementation of a prototype is being implemented on top of Agentsheets [4], a grid-based graphical programming environment suitable for the construction of visual programming languages. Using Agentsheets has the advantage of reaching a quite large community of users simplifying the user testing which has not been done yet.

Simplifying the creation of visual animation is of great importance since it increases their value to science education. Not only can the students create their own animation of the real world, they can interact with it, trying out different assumptions and ideas.

ACKNOWLEDGMENTS

The frequent discussions with Clayton Lewis, Alex Repenning and the members of the NAP group have been very helpful.

References

- [1] BELL, B. Using programming walkthroughs to design a visual language. Tech. Rep. CU-CS-581-92, Department of Computer Science, University of Colorado at Boulder, February 1992.
- [2] BRAITENBERG, V. *Vehicles, experiments in synthetic psychology*. MIT Press, 1984.
- [3] FURNAS, G. Formal models for imaginal deduction. In *Proceedings of the Twelve Annual Conference of the Cognitive Science Society*. (Hillsdale, NJ, July 1990), Lawrence Erlbaum, pp. 662–669.
- [4] REPENNING, A. Agentsheets: A tool for building domain-oriented dynamic, visual environments. Tech. Rep. CU-CS-693-93, Department of Computer Science, University of Colorado at Boulder, December 1993.