Journal of Visual Languages and Computing 20 (2009) 16–29

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

End-user strategy programming

Christoph Neumann *, Ronald A. Metoyer, Margaret Burnett

School of Electrical Engineering and Compurer Science, Oregon State University, Corvallis, OR, USA

ARTICLE INFO

Article history:
Received 20 September 2006
Received in revised form
20 March 2008
Accepted 28 April 2008

Keywords:
End-user programming
Natural programming
Visual programming
Computer animation
American football

ABSTRACT

Rule-based programming systems can be fragile because they force the user to account for all logical alternatives. If an unconsidered case does arise during execution, program behavior falls through the cracks into unspecified behavior. We investigate rule-based, end-user strategy programming by introducing our Interactive Football Playbook—a domain specific, end-user programming environment to allow American football coaches to create animated football scenarios by associating strategy information with virtual football players. We address the problem of rule explosion through "rule bending" to support a minimalist, scaffolding-driven programming environment. Additionally, we introduce visual language representations for logical and sequential "and" to mitigate end-user confusion with the semantic meaning of these "and" constructs.

1. Introduction

Computer generated content is richer than ever before, taking advantage of the greater capabilities of modern hardware. Scientific visualization experts can program complex software which generates visual, interactive content for users from data. Professional animators can use general purpose animation tools to create a vast array of content from instructional visualizations to life-like scenes for motion pictures. As the complexity of content increases, so do the challenges for content authors; the divide between the content creator and the content consumer grows.

Content consumers who want to bridge the gap and create content of their own are faced with a significant learning curve to get up to speed with readily available, general purpose content authoring tools. Lacking programming skills, an individual is limited to the domain-agnostic environment of the chosen content authoring tool (spreadsheet, animation tool, etc.), so the individual must learn a tool more abstract than the needed domain without supportive features relevant to the domain. This problem is compounded by the user's need to create content quickly and update it often. For example, an animation created using an animation tool is expressed at a low level—a set of concrete property changes over time (e.g. x–y location, orientation, scale, etc.) for a particular object being animated. Each of these points in time is termed a "keyframe" and properties are interpolated between keyframes in the "inbetween" frames. So a change to one object in part of the animation may result in large number of cascading changes to keyframes for other objects to keep the animation looking physically correct.

A simulation approach to animation allows a user to create animated content at a more abstract level than specifying concrete property changes over time. Rather than specifying desired properties for specific objects at points in time, the values of particular properties are determined by the simulation engine. This approach is used when there are many objects which are animated simultaneously and may have complex interactions between them. An example of such are particle systems which are used to model water, fur, and smoke. An end

* Corresponding author. Tel.: +1 541 758 4624.
E-mail addresses: christoph@neumannhaus.com (C. Neumann), metoyer@eecs.oregonstate.edu (R.A. Metoyer), burnett@eecs.oregonstate.edu (M. Burnett).

1045-926X/$ - see front matter © 2008 Published by Elsevier Ltd.
doi:10.1016/j.jvlc.2008.04.005

user could use extension mechanisms, such as macros or plugins, to customize a keyframe-oriented, general purpose animation environment to add domain-specific abstractions to support simulation-oriented authoring. However, developing macros and plugins often requires the user to learn another programming language (likely a scripting language) which represents information quite differently than the content authoring environment.

The research community has turned to visual programming languages to lower the barriers for end users to create domain-specific, interactive, animated content. Novices can use a visualization tool such as OpenDX [1] (a visual dataflow language) to create their own interactive visualizations that would otherwise require a strong background in a general purpose language such as C and a graphics API such as OpenGL. Agentsheets [2,3] allows end-user programmers to create domain-specific visual abstractions and use them to create simulation oriented, animated content. We believe rule-oriented visual programming environments hold promise for allowing end users to create computation-driven content such as what is required for visualization or animation.

We explore end-user creation of rule-based, computation-driven, animated content through our Interactive Football Playbook (IFP). The IFP is a simulation-driven, strategy-oriented approach to allow coaches to program animated football simulations. Fortunately, football coaches rely on a fairly standard symbolic language for specifying plays on static media such as paper or a whiteboard. The IFP builds on this standardized symbolic language and augments it with primitives and language enhancements that allow the coaches to specify strategy information for virtual players. Animations are then rendered by executing the programmed scenario and visualizing the parameters for the virtual players in the scene (Fig. 1).

Our approach is novel in a number of ways. Firstly, we contribute an approach for end-user strategy programming. Our approach differs from prior end-user programming work because we associate high-level, *strategy-oriented* rules with a virtual agent rather than using global, imperative rules to modify the *state* of the agents. The coaches' current use of static diagrams involves formulating a strategy in the coach's mind, translating that strategy into a drawn diagram, and then using the diagram as an aid to communicate the strategy to the football players. Only simple parts of the overall strategy are captured in the static diagram notation. Our approach elevates the semantic level of the captured information by allowing coaches to express more strategy information in the IFP than in the static playbook notation. For example, the coaches can use our notation to specify distance and sequencing information whereas in a playbook, that behavior is described using English. This richer language allows the IFP to generate animated simulations rather than simply representing static diagrams. Effectively, coaches author by using strategic concepts to create rather than simply scripting out animation actions. The outcome is more than a set of fixed animations; it is a repertoire of executable scenarios which exhibit specific strategies responding to the states encountered in the scenarios.

Our second contribution is our novel approach to rule bending. Like Repenning's work [4,5], we address the issue of rule explosion, but Repenning focuses on managing discrete permutations of rules. We apply rule bending in a continuous fashion.

We also contribute a notational device to express parallel and sequential "and" within a visual programming environment. Boolean expressions have been notoriously problematic for end users [6], so our device allows the user to express sequential and parallel "and" without the confusion that stems from using the overloaded English word "and". Prior work focused on logical "and" used in conditions for selection, but not on "and" used with regard to execution flow.

Finally, we contribute further evidence of the usefulness of the Natural Programming design process [7,8]. We successfully used the Natural Programming design process for creating the IFP and found the process to be

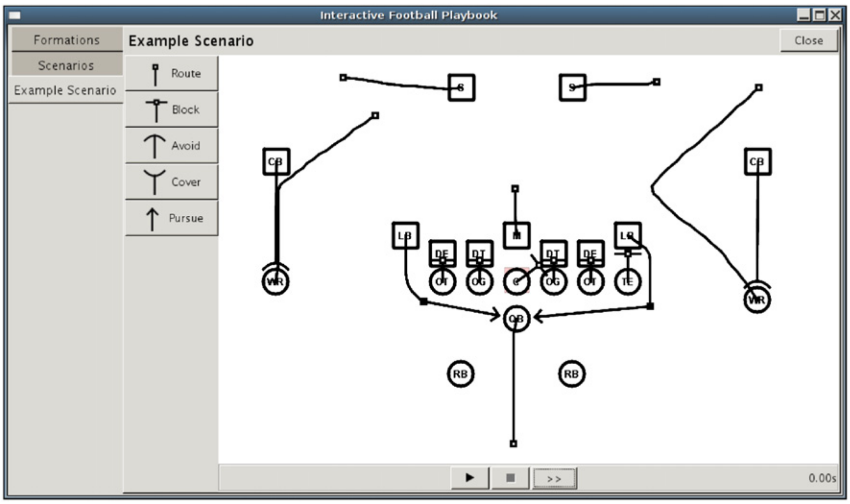

Fig. 1. A screenshot of the Interactive Football Playbook.