Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 224 (2009) 3–14
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Animalipse - An Eclipse Plugin for AnimalScript

Guido Rößling[1]

CS Department
TU Darmstadt
Darmstadt, Germany

Peter Schroeder[2]

CS Department
TU Darmstadt
Darmstadt, Germany

Abstract

The algorithm animation language AnimalScript, while highly expressive and versatile, is not easy to edit with no editor support. We have developed an Eclipse plugin for editing AnimalScript that includes a text editor, outline, and code assist. We expect that this plugin will make the editing process much easier and faster. The paper presents both technical aspects of the development and the resulting plugin.

Keywords: AnimalScript, Animal, Eclipse, Plugin, Animalipse

1 Introduction

Animal [5] is a versatile system for creating, modifying and presenting algorithm animations and visualizations (AV content). As far as we know, it is the only AV system that allows users to create AV content using all of the following approaches:

- *visually* using drag and drop in a novice-friendly graphical user interface [8],
- *textually* using the highly expressive AnimalScript language [7,9],
- employing a new Java-based generation API,
- using a set of external applications for generating context-specific animations for trees [10] as well as for graphs and graph algorithms [3,11],

[1] Email: roessling@acm.org
[2] Email: sevenclev@gmx.net

doi:10.1016/j.entcs.2008.12.044

- as well as using one of the currently more than 200 animation generators of the built-in *generator* framework [6]. Note that the number of generators does not necessarily indicate the number of algorithms covered, but rather the different "flavors" for the given algorithms, such as the choice of the programming language and the output language used for the presentation.

All generation approaches except for the first are directly or indirectly based on using AnimalScript, which is in the process of taking over the role of the preferred representation of Animal AV content from the built-in ASCII notation. The reasons for this development are the human-readable notation of AnimalScript, the ease with which it can be generated from programs and edited manually, and the expressiveness of the language. Since 2008, Animal also includes an integrated display of the BNF-based definition of the AnimalScript notation, as well as (since 2006) a small text editor for directly entering or modifying AnimalScript input and visualizing the results.

AnimalScript files contain one command per line, such as a definition of a new graphical object or a transformation of some objects. The animation is organized in steps, each of which can contain one or more commands. If multiple commands are used in a step, the step is surrounded by curly braces { }. Please see [9,7] for more information about AnimalScript.

Many of the other established AV systems also cover some of the generation approaches listed above. For example, JAWAA2 [1] and the GAIGS and JSamba [12] visualization engines used by JHAVÉ [4] also use a scripting language. JAWAA2 also offers a visual editor in its current release. JHAVÉ offers a set of content generators that are similar to the approaches offered in Animal's generator framework and can be run off the web. However, they focus on specifying algorithm parameters, and thus do not allow the definition of visual properties such as colors.

While AnimalScript can be edited easily using Animal's built-in editor or any arbitrary text editor, the comfort offered by this is somewhat lacking. The internal editor only offers rudimentary editing features; *cut, copy* and *paste* features are only supported by using the underlying operating system support. The editor does not offer a search facility, display of line numbers, indication of recognized syntactical or semantical errors, or syntax highlighting. Thus, editing a longer AnimalScript file is awkward and can become frustrating if the system indicates a parsing problem "in line 117". Despite (usually) precise information about the nature of the error, the lack of line numbers, search or "go to line" functions makes locating and fixing the error a tedious and less than enjoyable process.

We decided that his unsatisfying state needed addressing. Essentially, we saw three different approaches to provide better user support: improve the built-in editor to be comparable in comfort to the user's preferred text editor, create a new custom editor for AnimalScript content, or provide AnimalScript bindings for at least one commonly used text editor. It did not seem useful to invest much effort only to improve the built-in editor so that it would be comparable to, but still different from, a given user's preferred text editor. The same applied to creating a new custom editor. Therefore, we opted to provide AnimalScript bindings for at least