Programming for Artists: A Visual Language for Expressive Lighting Design
Joshua B. Gross
School of Information Sciences and Technology, The Pennsylvania State University
jgross@ist.psu.edu

Abstract: Programming is a process of formalizing and codifying knowledge, and, as a result, programming languages are designed for generalists trained in this process of formalization. Artists, whose training focuses on skill and tacit knowledge, are marginalized by existing tools. By designing visual languages that take advantage of an artist’s skills in visual perception and expression, we can allow that artist to take advantage of the expressive potential that modern computing offers. In particular, this paper will look at lighting design for interactive, virtual environments, and augmenting an existing programming language to allow artists to leverage their skills in the pragmatics of that medium.

Keywords: Semiotics, pragmatics, domain specific language, virtual environments

1. Introduction

What does it mean to be marginalized by the move to digital? Training as an artist involves extensive immersion in the practice of producing art, but artistic knowledge can be exceptionally difficult to formalize. Artists are not theoreticians; their focus is on praxis [1]. In order to empower and engage artists, it is the responsibility of the language designer to consider how to construct a language that focuses on the needs and strengths of the artist-programmer, including the practical (outcome) orientation of the artist, as well as the visual knowledge the artist holds.

2. Domain-Specific Languages

Most programming languages are designed to be general purpose. This is not an absolute; FORTRAN and Mathematica, for example, are designed for science, engineering, and math applications. The move to OO languages shows an interest in allowing developers to define semantics ad hoc, but the languages themselves are still general.

A domain-specific language has one major disadvantage, but several key advantages. The disadvantage is that the domain centricity of the language prevents it from being applied to other domains, and the knowledge acquired in learning the language is not readily generalizable. However, the advantages outweigh this disadvantage for certain domains, and for certain (would-be) programmers. These advantages include the ability to use existing mental models of their work and to avoid the learning curve associated with general programming languages. Unless a user aspires to be a professional programmer, general languages can be daunting, due to the steep learning curve associated with learning programming [2].

3. Expressive Lighting Engine (ELE)

The example domain we are currently exploring is dramatic lighting. Magy Seif El-Nasr’s work on ELE (Expressive Lighting Engine) [3] is an excellent example of a domain-specific tool; it allows a programmer to control lighting for a virtual environment, such as a video game or interactive fiction.

However, the level of control goes far beyond placing and directing individual lights, as one can do in any virtual environment development tool, such as Unreal Tournament. ELE augments this basic type of lighting by allowing dynamic control of the lights in order to meet expressive goals, such as increasing or decreasing dramatic tension throughout a scene or set of actions.

The tool is designed for artists, but is based extensively on mathematical formalizations of the independent controls of lighting and the dependent dramatic outcomes that tuning those controls can produce. ELE currently requires its users to program in a language such as C++ or C#. Our focus is to design a visual interface to ELE for non-programmers, specifically, for lighting designers who wish to use ELE’s expressive power without learning to program.

4. Semiotics and Programming

Semiotics, the study of systems of symbols, has long been considered important in programming [4]. However, most programming language development focuses on lower levels of semiotics; syntax (how symbols are organized) is of obvious importance, but semantics (the meaning of symbols) has grown more important as language designers have recognized the importance of allowing programmers to build domain knowledge into their systems.

5. Dimensions of Control in ELE

The value of ELE is its ability to allow an artist to control the affect, or intended emotional state, of a viewer or player. ELE is a complex environment that
provides control of many different dimensions simultaneously. In addition to the three dimensions of the space to be lit, and the lighting elements themselves, ELE adds in the dimension of time, the lighting of individual characters, and the selection of colors (including saturation and brightness), to achieve the central outcome of promoting the story and its intended effect.

Because ELE is a sophisticated tool for controlling and adjusting many dimensions simultaneously, it would not be appropriate to implement a form-based GUI. This would not meet the needs or desires of the intended users, and would necessarily undercut the expressive power of the language. At the same time, it would also be inappropriate to force lighting engineers to learn to use ELE through a generic programming language.

6. Pragmatics, Behavior, and Effect

Pragmatics is the branch of semiotics that deals with context and behavior, two crucial aspects not considered by syntactics or semantics. This level of analysis is crucial to developing tools for artists, because art is so heavily immersed in practice and action, and because art is valued on its ability to communicate.

Lighting is an excellent example of pragmatic semiotics. It is used extensively in theater, film, and games to control moods. In black and white film, for example, designers used light, shadow, and contrast to evoke moods so powerful that an entire genre, film noire, is based on these principles.

The goal of pragmatics is to model behavior that has an intended effect or outcome. This is an appropriate framework for a language for expressive lighting design, in that the domain relies heavily on practice (behavior) and affect (intended emotional outcome).

One example of this pragmatic approach would be to allow the designer to specify, for a particular scene, a shift from relaxation to dramatic tension. The graphical tool will simultaneously increase saturation and contrast, while decreasing overall brightness, without forcing the user to quantify or formalize these actions. Given the intense processing demands of virtual environments, the tool could also represent (visually, of course) an approximate order of magnitude of complexity of the particular control scheme, to give the designer a view of potential performance issues.

7. Research Approach

Any research to develop a domain-specific language must begin with immersion into that domain. In addition to an extensive literature review, we will be working with interactive, virtual environments in research settings at two universities. The purpose of this research is to gain a thorough understanding of the attitudes and impressions about current tools. In particular, we will look for aspects of existing systems that allow these designers to express themselves creatively without taxing them extensively to learn a generic tool.

When the data has been collected and analyzed, we will use existing software engineering and HCI techniques, including scenario-based design [5], cognitive walkthroughs [6], and system mockups, to test the learnability and usefulness [7] of the proposed visual language.

Ideally, the process of developing an interface to ELE will be an iterative, creative process that involves participants with skills in programming language design, virtual environments, lighting design, semiotics, and HCI.

The final phase is validation, and this will have two measures; first, we can perform controlled experiments to measure the usability of the system, but we can also see if the interface increases the number of people who incorporate ELE into their work. Ideally, the latter will be the focus.

8. Directions for Research

As mentioned above, this abstract is designed to summarize a large-scale and long-term project to research, design, and implement a visual language for lighting designers that gives them full expressive access to ELE.

However, the goal is not only to create a domain-specific language. In addition, we seek to gain an understanding of some key issues of domain-specific visual languages: to understand the value of visual expression in a language (especially for visually-oriented programmers) and to determine the effectiveness and appropriateness of pragmatics as a framework for the development of these languages.

References