Graphics Languages & Tools: A Survey

Matthew Castellana, Y. Annie Liu
Design and Analysis Research Lab
Supported in part by NSF under grant CCF-1954837 and a GAANN fellowship

Stony Brook University
Computer Science

Overview

Creating graphics and graphical applications can be tedious and time-consuming, especially for those who do not specialize in the area. To better understand the issues, as well as develop possible solutions, we survey key graphics languages and tools and divide them into three categories:

- Dedicated Graphics Languages
- Graphics Libraries and Interfaces
- Interactive Graphics Environments

We identify four features that are key to a powerful, high-level graphics language.

KEY FEATURES OF A POWERFUL GRAPHICS LANGUAGE

Rich graphical primitives and models - A rich library of graphics primitives capable of constructing detailed models.

Ease of scripting - Language support for specifying commands and actions for dynamic and interactive applications.

Concurrent objects - Language support for easy modeling of complex concurrent objects.

Declarative constraints - Language support for specifying spatial and temporal relationships declaratively.

Image Source: (CC) Blender Foundation | cloud.blender.org/spring

As the complexity of computer graphics has evolved over time, so too has the complexity of their creation.

Dedicated Graphics Languages

Self-contained standalone programming languages for creating graphics applications.

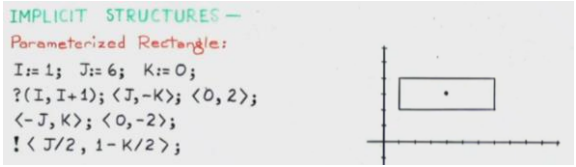
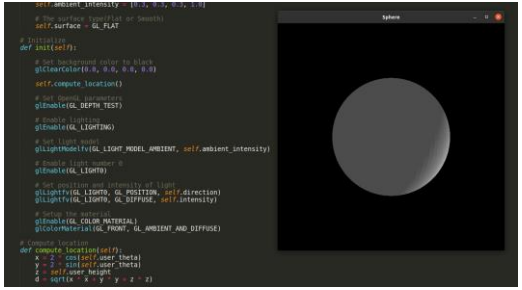


Image Source: Professor A. Kaufman, Stony Brook University

Graphics Libraries and Interfaces

Libraries with interfaces that extend general-purpose programming languages to provide graphics functionalities.



Interactive Graphics Environments

Programming environments that rely on a graphical user interface to create graphics, possibly providing scripting capabilities.