Morphic 3: The Future of GUIs

Juan M. Vuletich

CaesarSystems LLC
juan@jvuletich.org
www.jvuletich.org

Abstract

Morphic is the GUI framework in Squeak and Self. It was born simple and powerful. However, it is aging and it can be enhanced. This talk presents the current status of a development effort to redesign Morphic in Squeak. Besides a massive cleanup, there are two main ideas. The first one is to give each morph its own coordinate system. The second idea is to be completely agnostic about display size or resolution. The result are applications that are simpler to develop and look better on any display.

Keywords *Morphic; GUI frameworks; coordinate systems*

1. Introduction

Application developers constantly face the need to develop user interfaces for their applications. Many of the problems that need to be solved appear again and again. Therefore, pre made libraries and frameworks were developed to help on these problems, usually as part of programming tools. Most of them are tied to specific operating systems, and almost all of them consist of a pre made set of widgets. In a unique manner, Morphic puts widgets and graphics on equal footing. This gives unparalleled flexibility for widgets, and high level, powerful abstractions for graphic objects, in a simple, integrated framework. The Morphic framework in Squeak is the only GUI and graphics toolkit that allows building complex applications, including both conventional looking widgets and distinct, unique graphic objects, allowing the programmer to fully control de behavior and the look, up to the pixel, in a high level object oriented programming language.

2. Talk description

Morphic is the GUI framework in Squeak. It was originally developed as part of the Self project and later ported to Squeak. Designed with the ideas of simplicity and uniformity that are central to Smalltalk, it is both simpler and more powerful and general than regular window managers and widget toolkits. However, it is aging. Some of its problems were shown by its creators themselves, such as the lack of a coordinate system at each graphic object (morph). Others, such as code becoming too complicated, and the dependencies upon the applications built with it, were the result of the lack of good quality control for the code included in Squeak. Yet others were good decisions at their time, but it is time to move on. This talk presents the current status of a development effort to redesign Morphic. There are two main ideas. The first one is to give each morph its own coordinate system. These should not be restricted to Cartesian. Handling the coordinate systems applications might need in the framework itself will reduce and simplify the code of those applications. The second idea is to be completely agnostic about display size or resolution. This means the designer of a GUI should not know or be concerned about a possible size in pixels for his work, and the rendering should be done in high quality on whatever display the user has.

3. The limitations of Morphic

Any application that goes beyond widgets on a form needs to handle its own specific coordinate systems. Some examples are:

- Linear coordinate systems on graphics and photo editors, audio editors, music sequencers, video editors, page editors, etc.
- Logarithmic scales on audio editors, technical graphs, etc.
- Unusual scales like in written music (pentagrams)
- Geographic coordinate systems as in GIS.

However, none of the available tools goes beyond linear coordinate systems and affine transforms. In the Morphic implementation in Self, every morph defined a Cartesian coordinate system for the placement of its submorphs. This was lost in the Squeak version of Morphic, where every morph shares the World's coordinate system. This global coordinate system is in pixels. This means that it is the application programmer who must translate the points from the coordinate system of his domain to the Display. This

leads to lots of duplicated effort, and to many bugs and inconsistencies between applications.

4. Main design ideas for Morphic 3

The main design ideas for Morphic 3.0 are:

- Modeling of coordinate systems. A comprehensive hierarchy of 2D coordinate systems is included. They are not restricted to Cartesian or linear. Useful nonlinear coordinate systems include polar, logarithmic, hyperbolic and geographic (map like) projections.
- Separation of the handling of coordinate systems from the morphs themselves. A morph should only need to say "I want to use a log scale", instead of needing to convert every point it draws to World coordinates by himself. Every Morph defines a space and coordinate system. Its #drawOn: method and the location of its submorphs are expressed in its own coordinate system. This idea is from John Maloney, the designer of Morphic 1.0 and 2.0. Class Morph has an instance variable 'coordinateSystem' to hold an instance of the MorphicCoordinateSystem hierarchy.
- Complete independency of Display properties such as size or resolution. There is no concept of pixel. The GUI is thought at a higher level. All the GUI is independent of pixel resolution. All the rendering is anti aliased.
- Rendering is done applying "the signal processing approach to anti aliasing". Shapes and images are modeled as continuous functions. They must honor the Nyquist condition. They are sampled at pixel (or subpixel) position for rendering.
- A Morph is placed somewhere in an owner. Class Morph has an instance variable 'location' to hold an instance of MorphicLocation. A MorphicLocation specifies a position, extent, and rotation angle expressed in the owner's coordinate system. Separating them eases the moving, zooming and rotation of morphs. The user can easily move, zoom and rotate morphs. Please note that we are talking about zooming and not resizing.
- In general, morphs are not resizeable. The general operation available to the user is called zooming. The zoom is defined by the extent (width/height) in the location of the morph. The user can modify the location of any morph. However, the size of a morph is defined by its own coordinate system and this is generally not modifiable by the user. If some morph also wants to offer the user the possibility of modifying the coordinate system (and hence the size), it is its own responsibility.
- All coordinates are Float numbers. This is good for allowing completely arbitrary scales without significant rounding errors.
- The Morph hierarchy is not a hierarchy of shapes. Morphs don't have a concept of a border or color. There is no general concept of submorph aligning. A particular morph may implement these in any way that makes sense for itself.

5. About the presenter

Juan Vuletich is senior developer at CaesarSystems. He has worked as a Smalltalk developer for over ten years, including an internship with Alan Kay's group at Disney. He is an active member of the Squeak community. His contributions include the OS/2 port of Squeak, the JPEG reader, and the PhotoSqueak image processing framework. He managed the Morphic Team, and is now developing Morphic 3.0. He holds a Ms.Sc. in Computer Science from the University of Buenos Aires.