Hardware Acceleration for Interactive, High-quality Graphics in Live Programming Environments

Master's Project Proposal, Summer Term 2020
Software Architecture Group
Prof. Dr. Robert Hirschfeld

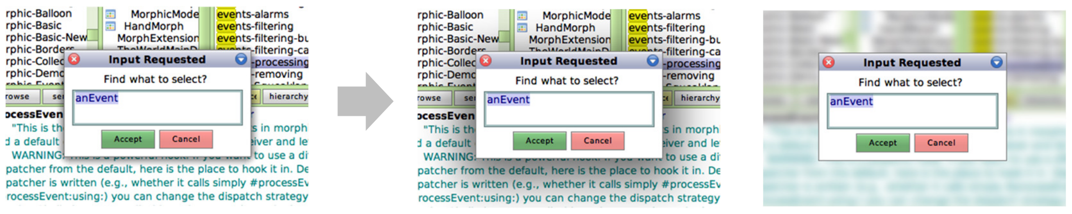
Project Scope and Goals

In this project, students will explore the design space of hardware-supported 2D rendering in direct-manipulation, live-programming systems using the Squeak/Smalltalk system with Morphic as graphics framework. The following goals form the starting point for this project and are likely to be refined and extended:

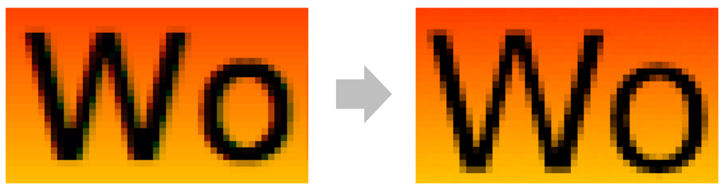
- *Explore* interdependencies between tools for live programming and mechanisms for hardware acceleration. Document the opportunities and limitations of that boundary. Consider the environment's capabilities for direct manipulation, exploration, and liveness.
- *Design* and *implement* a concept of layers and layer composition as an extension to Squeak's (and Morphic's) canvas-based drawing abstraction.
- *Implement* and *evaluate* (at least) three exemplary compositors (for layers) to integrate hardware-specific features for improving Squeak's rendering quality and performance: (1) CPU-based, (2) OpenGL-ES-based, (3) Metal-based.

As a starting point, the following scenarios should be used to experiment with and evaluate the proposed solution:

- *Improve* the quality of window shadows and enable efficient background blur:

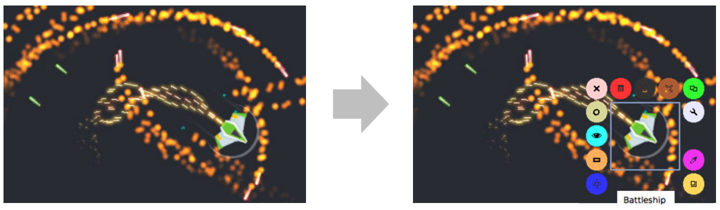


- *Improve* the quality of font rendering with focus on sharpness and correct sub-pixel AA:



The font is DejaVu Sans.

- *Improve* the performance when applying arbitrary transformations to thousands of morphs while preserving Morphic's direct-manipulation capabilities:



The halo implies direct manipulation.

Context and Motivation
Since its original implementation in Smalltalk-80, BitBlt has been responsible for combining bitmaps with color information, called forms, to composite the system's graphical output. There are combination rules to, for example, configure the transparency color during the "blitting" process between forms. It's a simple and effective way to decouple the system from the specifics of its host environment (Windows, Linux, macOS). All involved objects and data structures are accessible so that users can inspect and debug intermediate results. The system supports an exploratory working practice.

Almost since the advent of Squeak in 1996, the Balloon2D plugin has been complementing BitBlt to enable more advanced graphical effects such as gradient fills, rounded rectangles, and Bezier curves. Finally, the canvas-based abstraction was added to Squeak as part of Morphic. A canvas allows clients to only issue high-level drawing operations without having to deal with the underlying forms and their pixel-based representation. Note that the inspection capabilities remain, which simply expose pixel information through tools to users for debugging.

However, the current design of Squeak's BitBlt, Balloon2D, and Canvas mechanism does not support the use of potentially hardware-accelerated means from the underlying host environment. This lack of support results in a negative impact on quality and performance, which in turn affects the quality of applications programmed and executed in the Smalltalk environment. Considering the examples described above, window shadows are too simplistic, a font's glyphs are pre-rendered with hard-coded subpixel AA, and Morphic's interactivity does not scale up. At the end of the day, BitBlt is just about copying memory from a source to a destination. Any high-level intent got lost at that point; there is no concept of shaders. Thus, the interface that the existing plugins (here: BitBlt and Balloon2D) establish between the Squeak image and the OpenSmalltalk virtual machine needs to be extended.

Fortunately, there are well established standards for communicating to acceleration hardware such as OpenGL for graphics cards. These standards can help designing a better interface between Squeak and the OpenSmalltalk VM to improve the rendering performance and quality. The biggest challenge is to exploit the system's capabilities for exploration and direct manipulation – and still benefit from hardware acceleration if available and applicable.

As an overall idea, Morphic's canvas should be extended to support layers, which can then be composited in different ways. Layers should keep track of transformation commands (e.g., rotate, scale, translate, sheer) and post-processing commands (e.g., blurring, color mapping). Different compositors should be implemented and then selected, depending on whether hardware-acceleration is available. The basic, CPU-bound way would largely correspond to the current BitBlt implementation.

Project Values
In this project, students are expected to follow certain practices and represent certain values so that all emerging ideas and solutions can outlive this project's timebox:

- **Explorable implementation**

  The major portion of the solution should be implemented in Smalltalk; even VM plugins should minimize C and maximize Slang, a Smalltalk subset. There should be no C code hidden in Smalltalk strings to be sent via a foreign-function interface (FFI) to an external library. This should help programmers follow an exploratory working mode and interactively debug issues with Squeak's existing toolset.

- **Backwards compatibility**

  The proposed solution should work for existing applications, too. This means that the interface between morphs and canvases should basically remain as is. Extensions are possible but should be evaluated thoroughly to not leak the intended abstraction.

- **Cross-platform compatibility**

  The proposed solution should work on recent versions of Microsoft Windows, Apple macOS, and established Linux distributions. Consequently, VM plugins should be designed for existing cross-platform interfaces such as OpenGL if possible. Note that the Metal backend is very likely to replace OpenGL on macOS in the future.

- **Community contributions**

  On a regular basis, parts of the proposed solution should be packaged as contributions to the Squeak Trunk as well as the OpenSmalltalk VM's main branch. On the one hand, this includes fixes, code refactorings, and feature additions to Squeak's base system and Morphic. On the other hand, this can include changes to existing plugins in the virtual machine.

Further Reading
D. H. H. Ingalls, T. Kaehler, J. H. Maloney, S. Wallace, and A. C. Kay, "Back to the Future: The Story of Squeak, a Practical Smalltalk Written in Itself," in *Proceedings of the 12th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, 1997, pp. 318–326.

J. H. Maloney, "An Introduction to Morphic: The Squeak User Interface Framework," in *Squeak: Open Personal Computing and Multimedia*, Guzdial, Mark and Rose, Kim, Ed. Prentice Hall, 2002, pp. 39–67.

J. H. Maloney and R. B. Smith, "Directness and Liveness in the Morphic User Interface Construction Environment," in *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*, 1995, pp. 21–28.

M. Taeumel and R. Hirschfeld, "Evolving User Interfaces From Within Self-supporting Programming Environments: Exploring the Project Concept of Squeak/Smalltalk to Bootstrap UIs," in *Proceedings of the Programming Experience 2016 (PX/16) Workshop*, 2016, pp. 43–59.

A. Goldberg and D. Robson, *Smalltalk-80: The Language and its Implementation.* Chapter 18: The Graphics Kernel. Addison-Wesley, 1983.

Organization
A group of three to five (3-5) students may participate in the project. Organization and tasks will be explored and mainly determined by the project participants. The project will be carried out at the Hasso Plattner Institute in Potsdam. Project participants are expected to communicate with the advisors on a regular basis. Communication tools include GitHub issues, projects, wiki, and e-mail. There will also be talks to prepare so that students can collect feedback from the entire Software Architecture Group. Communication is likely to be conducted in English and German. Expected results include a working software accompanied by appropriate documentation.

Contact
Prof. Dr. Robert Hirschfeld (C-E.7, hirschfeld@hpi.uni-potsdam.de)
Marcel Taeumel (C-E.11, marcel.taeumel@hpi.uni-potsdam.de)

Software Architecture Group
http://www.hpi.uni-potsdam.de/swa