Integrating Graphic Designers into the Embedded Design Cycle

A Crank Software White Paper

January 2010

Abstract

The last few years have seen a significant increase in the number of embedded products containing an integrated graphical display. This provides consumers with a much more visually appealing presentation of information but, at the same time, introduces additional challenges for the embedded development team. Product requirements now must also include details about the visual presentation of the product, as well as the system level specifications. Often the graphic design team is detached from the standard software development team, leading to an “over the fence” style of development. Graphic artifacts such as images and screen designs are handed off to software developers who, in turn, must interpret them and attempt to make them fit into the rest of the system design. The two teams are often quite disconnected leading to inevitable integration problems. This white paper examines some of the technical issues that graphic designers and system engineers need to be aware of when designing a graphical product for the embedded market and look at tools to facilitate the graphic design process for embedded systems by assisting with integration and highlighting technical issues early in the development cycle.

Crank software inc.

The user interface design dilemma

The design and development environment for embedded products has changed significantly in the last ten years. The landscape of embedded devices has changed, due in large part by the advances in consumer electronics; in particular, the cellular telephone market. Now, an embedded device may still have a small form factor, but its system capabilities are richer than the desktop pc's of ten years ago [1]. Products that were originally using 8 and 16 bit micro controllers have been widely replaced by cheaper and more powerful 32 bit processors. Embedded devices that were once considered to be CPU and memory constrained now have the ability to run much more sophisticated applications.

Along with this change in processor and memory capabilities comes a much greater challenge: Evolving a product's user interface (UI) or user experience.

Adapting a product to use a different processor or make better use of available memory to improve performance falls within standard software engineering practices. There are a wide variety of software tools, packages, and support libraries that can be used to facilitate the migration of source code from one configuration to another.

The resources available to help update or adapt a product's UI are not so bountiful. For embedded products there are often two dilemmas. The first is how to create the initial UI. For many embedded products, consumer or otherwise, UIs are being created where previously there were none.

The second dilemma arises as the product evolves. What is the most effective way to evolve a UI in sync with a product's evolution? For those embedded products that provided user feedback, it may have previously been in the form of a line oriented display or terminal interface. These technologies are decidedly unsophisticated in their graphical complexity providing a very functional interface to end users, but none of the richness that today's iPhone generation expect.

The "iPhone effect" [2] is the term that is being applied to describe the introduction of display technologies to a myriad of products that had previously had only rudimentary UIs. While there is a strong desire for rich interactive displays, it is the ever decreasing price of LCD and touch screen displays, pushed by an ever-expanding cell phone market, which

allows this desire into reality. Adding an LCD, and the corresponding graphical user interface (GUI), to a product has become an effective and inexpensive technique used to re-brand older product lines. For new products, the addition of a graphical display technology and a touch screen interface can provide a significant differentiator to a new product entering an existing market.

The challenge remains; however, in the actual design and implementation of the GUI and its integration into the rest of the product's hardware and software development process.

A new term, user experience or UX, has emerged in product development circles in recent years [3]. The intent of the term is to highlight the fact that a product's UI is more than just a set of assorted widgets on a screen. A product has an operational flow associated with it that is the sum of the graphical interface, the presentation of the data, and the methods by which the user interacts with the system. This constitutes the UX and when most UIs are being designed, they are being designed with a particular UX in mind.

UX is one reason that the addition of a UI to a new product, or the updating of an existing UI, becomes important to consider as part of a product's software development process and not considered as outside of the general engineering and product development cycle. A graphical interface is more than simply a set of assorted widgets provided by a graphic designer to a software developer.

Graphic design in the product development process

There are two popular models for integrating graphical content that make up a product's UI and; as such, the role of the graphic designer in a product's development lifecycle.

The first model is to integrate the individual assets using standard graphical libraries and widgets, and then use the image content from a graphic designer to skin the standard controls. This technique usually has software developers implementing the UI, guided by images reflecting static snapshots of how the UI should look as imagined by a graphic design artist. Software developers are used to implementing the UI in this situation. The access to the standard embedded graphical libraries is programmatic rather than declarative and the tooling available, if any, is often targeted at software developers.

In this technique for implementing the UI, the graphic designer plays a passive role. They are actively engaged initially to help form the product vision and design the overall UX, but as the product moves from conception to production they play a reduced role in the hands-on development of the product.

Often a project cannot afford to have a graphic designer who is dedicated to simply supervising the implementation of a UI. In order for this technique to result in a product that's UI is a true reproduction of the original design, it requires constant validation of the transformation that the software developers are performing. The software developer is responsible for mapping the implementation of a static image to an interactive product and in that process there are many areas where choices need to be made.

Unless the graphic designer is readily available, the software developer will use his or her best judgment to implement functionality that is unspecified. The magnitude of these decisions depends on the quality and detail of the specification.

If the designer is not a dedicated member of the development team and readily available, then the alternative is that a UI specification must be created that is so detailed, that it in fact represents a complete abstract implementation of the product. This is unrealistic for most consumer products that have both small development teams and shortened time to markets. Just as unproductive is having a graphic designer who cannot contribute to the development and enhancement of a product as it is evolving without having to engage with another resource, such as a software developer.

As a result, over the last few years as UIs have proliferated to more and more products, product development teams are re-evaluating how their teams create, develop, deploy, and maintain their UIs.

Integrate the technology

If throwing static UI designs "over the wall" to software developers to interpret and re-implement is not an effective or scalable technique, what is its replacement? One approach being applied is to move the technology being used by the graphic designers directly into the embedded products, rather than spend the time and resources transforming UIs created by graphic designers through a standard graphics/widget library.

Examples of technologies that are being pushed into embedded products in this fashion include the integration of Adobe Flash, Microsoft Silverlight, or web-browser based UIs using the WebKit.org or Mozilla.org engines.

Rather than delegating the graphic designer to a sideline supervisory role during product development, this technique allows the designer to participate as an active member of the development team. As user feedback is received or as requirements change, the designer can adjust the UI elements quickly because they are working in a familiar environment. The final UI has the potential of being much more aligned with the original vision, including the subtle but important details that would normally be lost during the transformation by a software developer.

The software development costs associated with this approach are not inconsequential however. Many of the design tools used by graphic designers are designed for optimal use in a 'pre-production' type of computing environment. The CPU and memory resource consumption used by the environment is often an insignificant concern since they run on desktop systems rather than embedded systems [4]. The effort required to port these technologies to an embedded platform can be significant. Not only must the technology work in a resource-constrained environment, but it may have to be additionally tuned for performance, power management, or real-time behaviour with respect to other device activities.

It is also a concern that in a fast moving technology area such as the consumer device space that this effort may have to be undertaken for each change in operating system (OS) or CPU architecture, significantly diminishing the cost benefits of 'reclaiming' a software developer to concentrate on system development rather than UI generation.

Graphic design for embedded

Neither of these two approaches to developing and maintaining a product's UI is ideal. Neither alternative leverages the full capabilities of the graphic designer to make them an active and productive member of the development team, while at the same time providing an optimal technical solution targeted specifically at embedded and consumer products.

A large part of the challenge is the fragmentation of available tools and technologies. There are few, if any, solutions that allow efficient end to end graphic designer participation in a product's development cycle. There are tools for content creation, tools for content management and revision control, tools to help with product build automation, tools for embedded system development and optimization, and tools to help with platform migration and updates. A software developer can effectively produce code and have it flow easily through to the end product and be confident that it will produce the desired effect. There is no such confidence afforded to a graphic designer. In order to ensure that accuracy in the original intent of the UI is replicated in the final product and is efficient, and that system resources are properly appropriated, a laborious and time consuming path must be followed.

What should an ideal development environment/system/process provide to ease this burden and improve the effectiveness of a graphic designer on a product development team? At a minimum, the following points should be considered:

- The system will allow graphic designers to work in an environment that they find familiar, and use the tools that they are comfortable with, and that make them most effective. This means allowing development on desktop systems and allowing them to use desktop graphic design packages, such as Adobe Photoshop, for content development.
- The system will tie directly into the tools that the graphic designers are using such that a graphic designer can transform their static UIs into "animated" UIs. This could be implemented as an export format integrated into the existing tools or the ability to import native content as produced by graphic design packages. Animation of static graphic content could be facilitated through the use of standard naming conventions for objects, such as using up/down name hints for button states.
- The system will highlight design errors that are specific to the performance and operation of the UI in the embedded environment early in the design cycle so that the graphic designer can adjust the design accordingly. Issues such as incorrect image colour depth or the use of extraneous alpha channels can have a significant performance effect on systems where computing resources are already being stretched.
- In addition to warnings about performance impacting design decisions, the system should allow the designer to simulate the UI of the device. This will model what is frequently done when Adobe Flash is used as a mock-up tool; however, by making

the system aware of the limitations of an embedded environment a much more representative simulation can be performed.

- The system will allow rapid and scriptable deployment of a UI design to a particular embedded target configuration. Similar in nature to the compilation phase for source code, this will provide a platform (CPU, OS, render and display configuration) optimized solution using the same input design elements.
- A common model for isolating UI design changes from the underlying system and vice versa is through the use of a Model-View-Controller (MVC) paradigm. The environment for mapping behaviour to graphical elements should assist, and potentially enforce, this separation.
- The system will facilitate collaborative development so that the UI can be developed in parallel by multiple graphic developers and efficiently merged. The ability to merge content, display changes, and resolve conflicts is a key function in distributed software development teams.
- The sum of this functionality describes a very power environment that would empower graphic designers to create innovative UIs and participate actively in product development, aligning their involvement with the rest of a product's software development team.
- In addition to the productivity benefits of allowing the graphic designer to focus on UI development for the final product, rather than an early prototype, the product will benefit from having a more refined UI. Being continuously involved, the graphic designer can more easily adjust the UI based on feedback of the overall UX that is provided to the product development team, turning what would potentially be a long feedback cycle into something much more aligned with today's agile development techniques.

Conclusion

More and more embedded products are including graphical display technologies, whether as an evolution of existing designs or as a completely new approach to interfacing with the user. To properly incorporate these displays into a product, graphic designers are being employed to design rich UIs to appeal to today's consumer.

Examining the most common techniques for incorporating UI design into the product development process, the inclusion of a graphic designer to the team represents a significant

cost to the project. In order to eliminate this inefficiency, a paradigm shift is required in how graphic designers and the UIs that they create are incorporated into the product development process. In particular, this means developing an environment that aligns the effort and workflow of a graphic designer with that of the software developers on the product team.

The result of this alignment will not only be a more efficient development process, but the release of products whose UIs are richer, more refined, and highly capable while still being able to run on inexpensive hardware configurations.

References

If you are interested in reading more about this topic, try these websites.

[1] Computing Power
http://en.wikipedia.org/wiki/Moore's_law
http://digitalcontentproducer.com/dcc/revfeat/disappearing-desktop-0409/
[2] iPhone Effect
http://www.tomsguide.com/us/pictures-story/104-iPhone3GS-Winners-Losers.html
http://www.youtube.com/watch?v=_ZyznnuBZdY
[3] User Experience (UX)
http://en.wikipedia.org/wiki/User_experience_design
http://www.upassoc.org/upa_publications/user_experience/
http://mashable.com/2009/01/09/user-experience-design/
[4] Desktop Technologies Adapted for Embedded
http://www.cultofmac.com/adobe-gets-bitchy-about-flash-and-iphone/20378
http://arstechnica.com/microsoft/news/2009/09/latest-windows-embedded-ce-includes-silverlight.ars

About the author of this white paper

Thomas Fletcher, Vice President of Research & Development, works for Crank™ Software. Thomas has worked in the field of real-time and embedded software development for more than 10 years and is a frequent presenter at industry conferences. He is a technical subject matter expert and thought leader on Embedded System Architecture and Design, Real-time Performance Analysis, Power Management, and High Availability.

About Crank Software

Crank Software is an innovator in embedded UI solutions for consumer electronics, automotive head units, and industrial devices. Compared to traditional electronic design automation tools, Crank Software's UI products and embedded development services enable R&D teams to more quickly develop rich UIs for resource-constrained embedded devices like in-car graphical displays and animated GPS systems.

CRANK SOFTWARE

Headquarters
Suite 200
700 March Road
Kanata, ON, Canada
K2K 2V9

Call
+1.613.595.1999

Email
info@cranksoftware.com

Visit on the web
Cranksoftware.com