

Documentation

UbiStock Ubiquitous Computing Prototype

Christoph Beckmann

30093

[christoph.beckmann\(at\)medien.uni-weimar.de](mailto:christoph.beckmann(at)medien.uni-weimar.de)

Maximilian Schirmer

31201

[maximilian.schirmer\(at\)medien.uni-weimar.de](mailto:maximilian.schirmer(at)medien.uni-weimar.de)

Contents

I	Concept	3
1	Introduction	4
2	Related Work	4
2.1	Calm Technologies	4
2.2	Informative Art	4
2.3	Sector Monitor Application	5
3	UbiStock Prototype	5
II	User Documentation	7
4	Install UbiStock	8
4.1	System Requirements	8
4.2	Binary Installation	8
4.3	Installation from Source	8
5	Use UbiStock	8
5.1	Refresh Stock Data	8
5.2	Access more Details	9
5.3	Change displayed Set of Stock Items	9
5.4	Create a Custom Set	10
5.5	Quit UbiStock	10
6	Remove UbiStock	10
III	Developer Documentation	11
7	Introduction	12
8	Quartz Composer	12
8.1	Spheres	13
8.2	Interaction	13
9	Class Hierarchy	13
10	Inter Application Communication	15
10.1	Notifications	15
10.2	Threads	16

Part I
Concept

1 Introduction

While computers are getting smaller and affordable, they get more and more embedded into artefacts of our everyday-life. Huge displays are flooding public spaces while having only little informational content. Thus presenting detailed content comes along with special knowledge about meaning of displayed shapes, or needs more attention to realise reading long texts, for example.

So, information display in ubiquitous environments is an over-all challenge. Besides the design and presentation of contents, so called awareness-information, the content itself is the crux of the matter. After solving the primary task of *what* to display, the next step is to display the information accordingly. But how to display complex issues with shapes as ambient information visualisation?

Another fact is, most ambient displays only present information without letting the option to manipulate. At first sight it seems logical, why to adjust or edit ambient information, that users only recognise with one eye? In fact the interest to manipulate presentation or their contents persists of the advantage to getting informed deeper when recognised interesting changes on ambient displays.

2 Related Work

2.1 Calm Technologies

Calm Technologies [Weiser & Brown 1995] are objects placed in our everyday-life that denote both center and peripheral awareness information. The purpose of *Calm Technologies* is to provide information, even in a calm situation.

This approach prevents using huge displays or other devices, like cell-phones or pagers to display. Instead environmental items are used.

[Weiser & Brown 1995, Chapter: "The Periphery"] A calm technology will move easily from the periphery of our attention, to the center, and back.

2.2 Informative Art

This is a formulation to display information as art [Skog 2006], appearing as static image but with a high level of dynamic changes to reflect information changes. The presentation of information was inspired by the painter *Piet Mondriaan* and is supposed to have a decorative component.

Information that is displayed is mainly bus-traffic and weather-conditions. They also evaluated the handling with two groups of users. One group received a short introduction on what all shapes and colours mean, the other group did not. As a result, the first group was well informed, while the second group puzzled over the meaning for a long period of time. The benefit for the second group was very low, but the decorative factor counts, too.

[Skog 2006, Chapter 3.2] Those who did not understand the information could still appreciate the piece as a decorative item, which some seemed to do, considering the following comments on the piece: *Inspiring!*

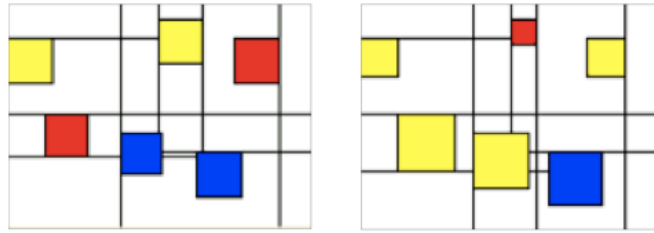


Figure 3. A weather display showing the weather in six international cities, inspired by Piet Mondriaan. The left image was generated in June 2001, the image to the right in December 2001, illustrating the differences in temperature between summer and winter on the north and south hemispheres.

Figure 1. Decorative ART with a high level of entropy.

2.3 Sector Monitor Application

Is an interface [Lin 2004] for financial institutions that compresses sales in different views. Buy and sell orders are clearly diagrammed.

Sectors are selectable, even current market activity can be displayed directly. Besides current market prices, historical trade-information and summaries may also be presented.

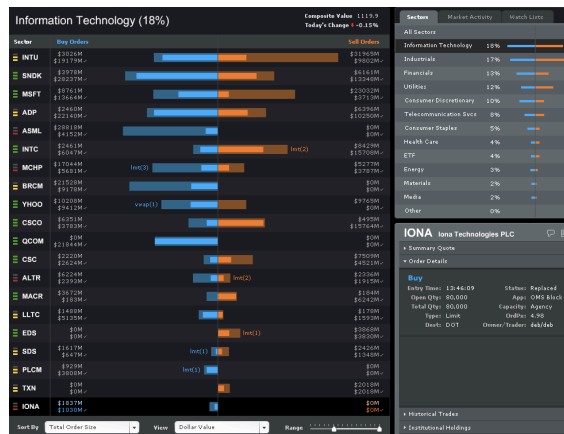


Figure 2. A Macromedia-Flash interaction prototype, giving a clue how *Sector Monitor Application* works.

3 UbiStock Prototype

Our approach is to display information as a mixture between shapes and text, that are ascertainable at a single glance. The display is mounted as central element in an office, it must be easily accessible from all workspaces. For most offices one single display won't be adequate enough.

Our display consists of different areas, on them packages are located. Areas with packages of lower priority are smaller, and there is one large presented area with the highest importance, but with less packages.

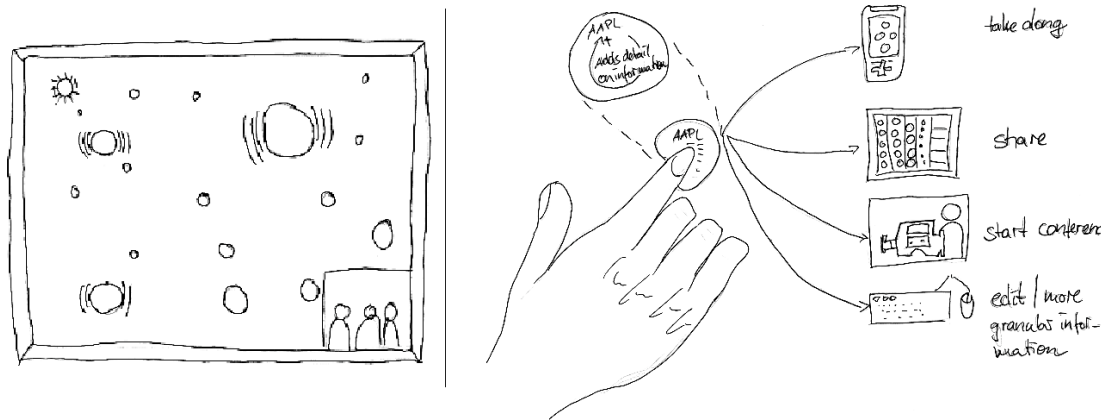


Figure 3. "Shark Pool" view with bouncing spheres and video conferencing window in the lower right corner; Manipulating information packages.

A package is represented through a circle and information or importance is mapped on size, colour. Each circle is labelled, on demand more information can be presented.

Beside the group-component, the office-wide display, there are a lot of little distributed displays that can interact between themselves. Those displays can be PDAs¹, reserved desktop space, cellular phones, etc.

Interaction between users and display is explicit asked-for. After getting interest in information changes it is annoying to search for information on personal computer. The interactive display allows to receive detailed information without searching. Therefore a graphics tablet can be used, this also implies a natural manipulating style. Actions like drag and drop information packages to preferred space on screen, or changing information granularity are easy to handle. Free-hand-interaction would be fine, too.

Beside adding or removing granularity on information, it is possible to share information-packages. Doing this is an advantage, so users can send their favourite packages to the group-display, or to other users-display.

Information presentation on a user's display depends on their preferences. This causes shape, adjustment, speed of motion, favourite colors, text-size and text-richness and certainly the level of detail.

¹Personal Digital Assistant, a handheld device that was originally designed as personal organiser

Part II
User Documentation

4 Install UbiStock

4.1 System Requirements

To use UbiStock, the following demands have to be accomplished:

- Apple Mac OS X 10.4²
- Phidget Mac OS X Framework 1.0.6³
- Apple Xcode 2.3⁴ (only required for installation from source code)
- **Recommended:** A working internet connection (to obtain live stock data)
- **Optional:** Graphics Tablet (for pen interaction)

4.2 Binary Installation

Drag the *UbiStock* icon to your desktop or applications folder and double-click to launch the application. **Note:** Installation to the /Applications folder requires administrator privileges.

4.3 Installation from Source

Alternatively, you can build *UbiStock* from source. To do so, open the included Xcode project (*UbiStock.xcodeproj*) in Xcode and choose the preferred target (Set Active Target: UbiStock RFID) from Project menu. After that select Build and Run from the Build menu.

5 Use UbiStock

Note: *UbiStock* is a prototypical application and is designed for studies in the field of Ubiquitous Computing. Although it has been tested and designed for stability, unexpected behaviour or application crashes cannot be excluded. Limitations like the number of available spheres or the underlying financial service result from trade-offs during the prototype design process.

5.1 Refresh Stock Data

Stock data is retrieved and processed every 20 seconds. Changes will be displayed using sphere colour and pulsation frequency.

Sphere Colour A sphere's colour is mapped to the corresponding stock item's change in relation to the previous closing trade. **Red** means the recent trade is lower than yesterday's trade while **green** stands for an increased trade.

²<http://www.apple.com/macosx/>

³<http://www.phidgets.com/modules.php?op=modload&name=Downloads&file=index&req=getit&lid=19>

⁴<http://developer.apple.com/macosx/>

Sphere Pulsation Frequency If a sphere is bouncing very slowly, the change value less than or equal to two percent. A change between two and four percent results in a medium pulsation frequency. Spheres with more than four percent change will bounce very quickly. Spheres stop bouncing after they have been clicked to provide a detailed overview. This feature helps to distinguish recently refreshed stocks from already noticed ones. Consult 5.2 for further information.

5.2 Access more Details

Clicking on a sphere stops the sphere's movement and reveals all available data, including the trade, change and a chart image. A second click will hide these information. The sphere will start bouncing again when new stock data has arrived.

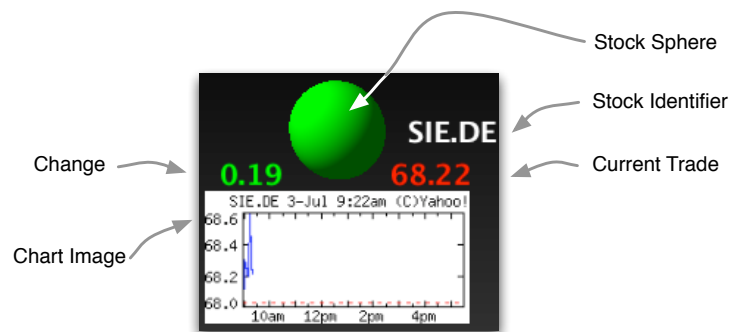


Figure 4. Stock Sphere providing all gathered information.

5.3 Change displayed Set of Stock Items

There are generally two ways for changing stock sets. You may select your preferred set by clicking on the button in the lower left corner, which reveals the stock set switcher. The stock set switcher indicates the currently selected stock set and allows to change by clicking on one of the other sets.

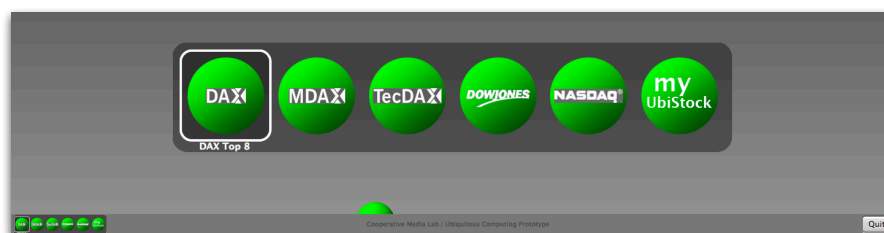


Figure 5. Stock set switcher showing currently selected and available stock sets.

The second way involves the RFID reader. Each RFID tag is associated with a stock set. Placing a different tag on the reader will automatically switch to the associated set.

5.4 Create a Custom Set

Follow the steps described in 5.3 to select the custom stock set. A sheet will be presented to select the desired stock items. **Note:** not more than eight items may be selected.

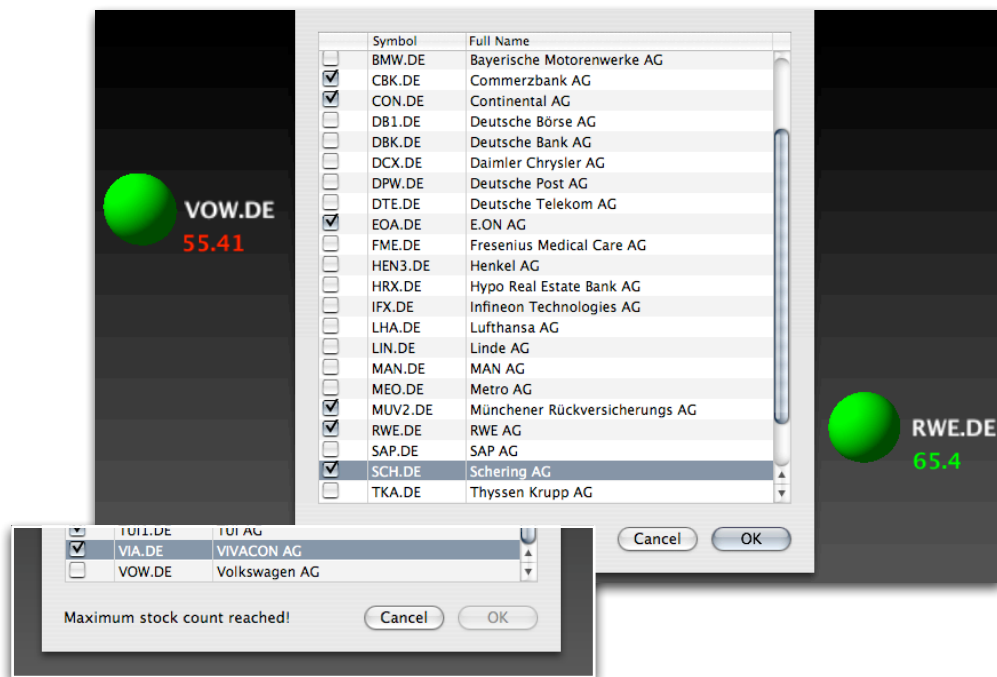


Figure 6. Custom set selection presenting list of selectable stock items. Giving advice, if prototype limitation of eight items has been reached.

5.5 Quit UbiStock

The `Quit` button in the lower right corner will close the application, using the `Command-Q` keystroke is also possible.

6 Remove UbiStock

Move the *UbiStock* application to the trash and delete the corresponding cache folder (`~/Library/Caches/UbiStock/`). If the Phidget Framework is no longer required, consult the Phidget Framework documentation for uninstallation instructions.

Part III
Developer Documentation

8.1 Spheres

The composition contains exactly eight spheres that can be enabled or disabled individually. This limitation has to be reflected by the application, so that users may not build custom sets with more than eight stock items.

Each sphere features a number of visible text labels and a chart image with the recent intraday chart. The sphere's position and movement are defined by two low frequency oscillations, x-axis modulated by sinus, y-axis modulated by cosinus. *Using different periods, offsets and phases for each sphere guarantees an unequally distributed alignment.*

Labels are positioned relatively to these coordinates using offsets. One label, defining the name of the stock, is always visible while the others are toggled dynamically on demand.

These demands are:

- Changes in data values
- Clicking on a sphere [cf. Interaction]

8.2 Interaction

Quartz Composer does not offer any means to determine if an element has been clicked or if the mouse cursor is currently over this element. We tried to implement such means using a method that checks if the current mouse pointer position is inside a rectangle that is moving synchronously with the sphere. The size of the rect matches exactly the sphere's size.

We later discovered that Quartz Composer will not allow cycles so our plan to stop the sphere's movement on mouseover was not realisable. To overcome this weakness, we changed our approach to using a single rect following the mouse pointer. By means of this rect, it is now possible to check if the mouse cursor is over a sphere and to stop it upon clicking.

9 Class Hierarchy

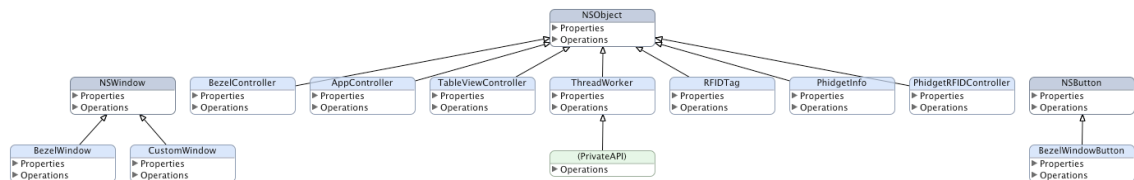


Figure 8. UbiStock Objective-C class hierarchy

CustomWindow Implements a fullscreen window that is displayed on the *main screen* without any surrounding borders. It also is responsible for fading in the *Quartz Composition* on start-up.

AppController The `AppController` class can be seen as the core class of *UbiStock*. It implements all features concerning the displayed data.

First, in a thread-environment, the corresponding stock data is gathered as csv-file from the *Yahoo Financial* service⁵. This file is parsed and all relevant data is processed.

In the end, the processed data is sent to the *Quartz Composition*.

Secondly, the `AppController` handles stock set changes. This includes the handling of predefined sets, that means cancelling the currently active thread, changing the dictionary and restarting the thread, as well as the selection panel display for creating a custom stock set.

For that purpose, notifications (cf. 10.1) are exchanged with the *TableViewController*-Class. After processing user input, a common set change, as described before is invoked.

BezelController This class manages the set switcher, including fade-in and -out, and mouse-over-effects. `BezelController` uses helper classes named `BezelWindow` and `BezelWindowButton`.

TableViewController The table inside the custom panel for custom stock item selection is connected with the `TableViewController`. For that a plist-File, containing all available stock items is read out and displayed as list. If a custom set already exists, corresponding stock items are marked as selected.

If users select an item, an internal count is increased. Due to *Quartz Composer* limitations, see section 8, an advice is presented, when maximum stock count has reached.

Another responsibility of `TableViewController` is, to save the selection persistently. Initiated through the `AppController` (if OK-button was clicked), all selected stock items are stored in an `.plist`⁶ and saved out to the file system. This procedure is strongly connected with notifications.

PhidgetRFIDController `PhidgetRFIDController` connects an *Phidget-RFID-reader*⁷ with *UbiStock*.

After reading a RFID-tag, a notification is sent to the `NotificationCenter`. The notification also delivers the tag's ID as payload. The `AppController` is an observer for that notification type and receives the tag ID. Every tag ID is associated with a set of stock data. If the custom tag is placed on the reader, the custom-panel will appear to allow selection of custom stock items.

⁵Yahoo Financial service: <http://de.finance.yahoo.com>

⁶Property List

⁷See documentation: <http://www.phidgets.com/documentation/1023.pdf>

10 Inter Application Communication

10.1 Notifications

Unlike the original OOP⁸ approach which relies on sending direct messages to instances of classes, *UbiStock* makes use of Cocoa's⁹ `Notifications`. Notifications are not exchanged directly between objects, in contrast, they are directed to the application's `NotificationCenter` where other software components may register for notifications.

When subscribing for a notification, the observer defines a method that is invoked when the notification occurs.

The following example explains this process. When a user selects the custom stock data set either by placing the associated RFID tag on the reader or by clicking the custom button in the stock set switcher, the `AppController` initiates displaying the custom panel. Because the table view inside the custom panel is connected with the `TableViewController`, it automatically receives the table data from the controller's data source methods. The `TableViewController` also checks if the maximum number of selected items has been reached.

Clicking the OK button invokes the `AppController` again, which will now request processing of the selected data by sending the `CMLProcessPanelRequest` notification. The `TableViewController` is registered as an observer for this notification and reacts by saving the data to an XML file. It posts the `CMLProcessPanelFinish` notification when processing has finished.

For this notification, the `AppController` is registered and reacts by closing the panel and updating the displayed stock data.

The table below shows all notifications, where they are posted and which component is registered as observer accordingly.

AppController	TableViewController
Observer	Observer
CMLProcessPanelFinish CMLBezelButtonClicked CMLRFIDnewTag CMLdataSetChanged	CMLProcessPanelRequest
Notifications	Notifications
CMLPanelCancel CMLProcessPanelRequest CMLdataSetChanged	CMLProcessPanelFinish

⁸Object Oriented Programming

⁹Cocoa Application Programming Interface <http://developer.apple.com/documentation/Cocoa/index.html>

BezelController	BezelButtonWindow
Observer	Observer
CMLMouseOverButton CMLMouseOutButton CMLPanelCancel Notifications	Notifications
CMLBezelButtonClicked	CMLMouseOverButton CMLMouseOutButton
PhidgetRFIDController	
Observer	
gotAttach gotDetach gotTag Notifications	
CMLRFIDnewTag gotAttach gotDetach gotTag	

Table 1. Notification overview

10.2 Threads

Because updating stock data and manipulating interface elements with effects like the fade-in and fade-out of the stock set switcher or the panel animation would interfere with the Quartz Composer rendering, *UbiStock* uses at least two threads most of the time.

The `ThreadWorker`¹⁰ class allows termination of a working thread at any time and handles locking of resources shared by different threads or processes.

¹⁰Public domain, by Robert Harder: <http://iharder.sourceforge.net/macosx/threadworker/>

References

- [Lin 2004] Lin, A. Andrew Lin, Sector Monitor Application [online]. 2004 [cited 20/05/2006]. <http://androo.com/portfolio/projects/sectormonitor/index.html>.
- [Skog 2006] Skog, T. (2006). AMBIENT INFORMATION VISUALIZATION. PhD thesis IT-university Goteborg ISSN 1651-4769;3.
- [Weiser & Brown 1995] Weiser, M. and Brown, J. S. Designing Calm Technologies [online]. 1995 [cited 20/05/2006]. <http://www.ubiq.com/weiser/calmtech/calmtech.htm>.