Inner View and Net Weight: Environments for Personal Health Exploration

Bradley E. Geilfuss, Jr.
S.B., Architecture
MIT 1994

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning, in partial fulfillment of the requirements for the degree of Master of Science in Media Arts and Sciences at the Massachusetts Institute of Technology

June 1999

© Massachusetts Institute of Technology, 1999. All rights reserved.

Author
Bradley E. Geilfuss, Jr.
Program in Media Arts and Sciences
May 7, 1999

Certified by
Michael Hawley
Assistant Professor of Media Arts and Sciences
Alex Dreyfoos Jr. (1954) Career Development Professor of Media Arts and Sciences

Accepted by
Stephen A. Benton
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences
Inner View and Net Weight: Environments for Personal Health Exploration

Bradley E. Geilfuss, Jr.

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning on May 7th, 1999, in partial fulfillment of the requirements for the degree of Master of Science in Media Arts and Sciences at the Massachusetts Institute of Technology

Abstract

Health monitoring technology is going to change the way health care and health awareness work. While there are many monitoring systems and technologies on the market or under research, few of these systems or technologies provide both the ability to communicate their data to broader infrastructures and the ability to visually educate the user about what that data means. Two prototype objects—Net Weight and Inner View—are presented in this thesis. Net Weight is a scale that records weight data and that communicates with Inner View, a mirror outfitted to augment the user’s reflection with visualized data from health monitoring devices.

Thesis Advisor:

Michael Hawley
Assistant Professor of Media Arts and Sciences
Alex Dreyfoos Jr. (1954) Career Development Professor of Media Arts and Sciences
Inner View and Net Weight: Environments for Personal Health

Thesis Committee

Thesis Advisor

Michael Hawley
Assistant Professor of Media Arts and Sciences
Alex Dreyfoos Jr. (1954) Career Development Professor of Media Arts and Sciences
MIT Media Laboratory

Thesis Reader

John Maeda
Assistant Professor of Design and Computation
Sony Career Development Professor of Media Arts and Sciences
MIT Media Laboratory

Thesis Reader

David Kelley
Chief Executive Officer
IDEO Product Development
Acknowledgements

For inspiration and guidance, thanks to Michael Hawley, David Kelley, and John Maeda.

Thanks especially to Essam Abadir, Charlotte Burgess, Kristin Hall, Matt Lau, and Maria Redin for the many little things.

Above all, thanks to my family for putting up with never hearing from me except to bail me out of a rent payment or to help me pay for a new tire, and to Sandy Choi, for her love, music, endless patience, and support.

Apologies to Boeing for keeping him in my office instead of outdoors where the squirrels were.
Table of Contents

1.0.0 Introduction ......................................................................................... 7
1.1.0 Scenario ............................................................................................. 7
1.2.0 Structure of this Thesis ....................................................................... 7

2.0.0 Background .......................................................................................... 9
2.1.0 Introduction .......................................................................................... 9
2.2.0 Moving Towards Personal Monitoring ................................................. 9
2.3.0 New Paradigms ..................................................................................... 10
2.4.0 Internet Medical Support ....................................................................... 10
2.5.0 Recent Progress in Bio-Monitoring ....................................................... 11
2.5.1 Polar ..................................................................................................... 11
MarathonMan ............................................................................................... 11
2.5.3 US Army Rangers ................................................................................ 12
2.5.4 1998 Everest Expedition .................................................................... 13
2.5.5 FitSense ............................................................................................... 14
2.5.6 FitLinxx ............................................................................................... 14
2.5.7 Toto Toilets ....................................................................................... 14
Tanita Scales ............................................................................................... 15
Ventus ........................................................................................................ 15
2.5.10 AirChart / LifeChart ......................................................................... 15
2.5.11 LifeLink ............................................................................................. 16
2.5.12 Guardian Angel .................................................................................. 16
2.6.0 What is Missing? ................................................................................ 17

3.0.0 Inner View ............................................................................................ 19
3.1.0 Introduction ......................................................................................... 19
3.2.0 Built Prototype .................................................................................... 19
3.3.0 Inner View System Architecture .......................................................... 22
3.3.1 Device Level ....................................................................................... 23
3.3.2 Agent Level ......................................................................................... 23
3.3.3 Display Level ...................................................................................... 24
3.4.0 Demonstration ..................................................................................... 24
3.4.1 How Data Feels .................................................................................. 24
3.4.2 Polar Heart Rate Monitor .................................................. 25
3.4.3 Polar HRM Agent .............................................................. 25
3.4.4 Pill Bottles and History .................................................... 26
3.4.5 The Magic Mirror .............................................................. 27
3.4.6 Interval Research Corporation’s Magic Mirror .................. 28
3.5.0 What Will the Future Look Like? ......................................... 28

4.0.0 Net Weight ..................................................................... 30
4.1.0 Introduction ............................................................................ 30
4.2.0 Built Prototype .................................................................... 30
4.3.0 Toe Scanning ........................................................................ 31
4.4.0 The Net Weight Agent .......................................................... 33
4.5.0 Weighing the Future ............................................................. 34

5.0.0 Continuance and Conclusions ............................................. 35
5.1.0 Continuing the Current Prototype .................................. 35
5.1.2 Glucose Sensors ................................................................. 35
5.1.3 Net Weight: the Next Prototype ................................. 36
5.1.4 Inner View: the Next Prototype ................................. 36
5.2.0 Conclusions ............................................................................. 36
5.2.1 Preventing Disease .............................................................. 36
5.2.2 Monitoring the Elderly ......................................................... 37
5.2.3 Staying Out of the Hospital .................................................. 37
5.2.4 Health at Home ................................................................. 37

6.0.0 References ..................................................................... 39

7.0.0 Appendix ......................................................................... 42
1.0.0 Introduction

1.1.0 Scenario

Shortly after noon on April 19th of 1997, I stepped over the starting line of the Boston Marathon in Hopkinton. Looking around, I noticed many of the other runners were looking at me. More specifically, most of the runners around me were looking at the fanny pack I was carrying, with wires leaking out and into my singlet. I was running the first wired marathon. Sensors on my body were recording data about me—where I was, how fast I was going, how warm it was in my stomach, and how fast my heart was pumping.

At the end of the day, when the race was over, the tape and sensors were removed, and I had taken a few aspirin, I was excited to see the results. Sadly, I was disappointed. An Excel graph showed me the route of the marathon (but I already knew what that looked like.) More graphs showed me my heart rate and my cadence. Something was missing for me. After quite a while, I realized what was missing was actually a few things.

One thing missing was a medical comprehension of the data. I did not have the skills to understand the data in all of its completeness. Another was the timeliness. By the time the graphs were done, the race was long over. I had showered, eaten, and even slept. Most of all, the entire process was unnatural. If I was going to continue to run with sensors, I was not going to continue to go back to my PC after every run to look at spreadsheets, especially not if I was going to do it everyday. It became a mission for me to explore what I was going to do with the results from this technology—where should the data go, and what could we do with it?

1.2.0 Structure of this Thesis

This thesis describes a prototype personal health monitoring constructed during the past year as part of research done within the Personal Information Architecture group at the MIT Media Lab. Section 2 introduces background information about emerging trends in health care and discusses how personal health monitoring—the ability to read vital signs from our bodies outside of the hospital will be and has been changing the doctor/patient relationship. Section 3 describes Inner View, a mirror with the ability to superimpose visual information gathered from health monitors and the
Internet onto users’ reflections. Section 4 describes *Net Weight*, a body mass scale that can communicate its data to Inner View. Finally, Section 5 will discuss conclusions and future work for this thesis.
2.0.0 Background

2.1.0 Introduction

_The most important part of an art is to be able to observe properly._

- René Laennec, Inventor of the Stethoscope.

René Laennec pioneered the practice of monitoring vital signs in the early 1800’s, when he first placed a hollow wooden tube between a patient’s chest and his ear so that he could listen for anomalies in a patient’s heartbeat. The number of tools used to monitor a patient has increased greatly since then. In 1999, a typical intensive care unit contains stacks of vital sign monitors which constantly measure a patient’s blood pressure, breathing rate, pulse rate, temperature, and the levels of many gases in the bloodstream [1]. While these stacks of monitors do the job, they are generally not portable, and they often require invasive techniques to acquire samples from a patient. The bio-medical engineering industry has been working to make sensors smaller and, when possible, non-invasive. The following chart shows the states of three important vital signs sensors, and how they have progressed over the last 60 years.

<table>
<thead>
<tr>
<th>sensor</th>
<th>1960</th>
<th>1980</th>
<th>2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse rate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blood Pressure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blood Oxygen</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

_Progress in health sensors._

2.2.0 Moving Towards Personal Monitoring

As health sensors become smaller and more cost-efficient, they are finding places in the consumer market. Since the mid-80’s, athletes have had the ability to monitor their heart rate from chest straps transmitting to wristwatch memory/display units. These sensors are less sophisticated than what would be found in a hospital, but they serve a different purpose—whereas hospital monitors determine whether or not a patient is out of risk, an athlete’s monitor becomes a benchmark for how healthy he or she may be. Runners commonly judge themselves not by how far or how fast they run, but by how high they can push their heart rate and how long they can
sustain it. Progressively, personal health monitoring will be capable of both giving doctors more complete and more reliable information about our health and giving people the information that can help them proactively improve their health.

### 2.3.0 New Paradigms

In the 1997 Boston marathon, researchers at the MIT Media Lab debuted a prototype personal health monitoring system that could monitor heart rate, cadence, core temperature, and location [2]. A later prototype system, that could additionally measure oxygen levels in the bloodstream, was adapted for climbers who were attempting to summit Mount Everest in 1998 [3].

The sophistication of these systems begins to blur the line between consumer and professional health monitoring. How long will it be before unobtrusive sensors for our bodies and for our environment are as sophisticated as those found in an intensive care unit? How will this transform the patterns of medical practice?

A century ago, physicians would make house calls. Once health monitors became necessary to physicians, patients were required to go to the hospital rather than have the physician come to them, as many of these monitors were not mobile. Some experts believe that the house call may return, as hand-held monitors are introduced that can perform all of the readings conventionally taken in the hospital [4]. Inexpensive consumer sensors will enable physicians to regularly diagnose patients without necessarily being in the same location.

### 2.4.0 Internet Medical Support

Intel President Andy Grove observed that 63% of doctors use e-mail. Of those doctors, 13% use e-mail to talk with their patients. Of those doctors, only 2% of doctors find e-mailing their patients is useful, yet 65% of all patients use the Internet regularly. There is an obvious in these numbers, yet a large number of patients have begun to use the Internet to find answers to their medical questions regardless of the relatively small number of doctors who share their expertise on line [5].

Internet sites like *Ask Dr. Weil* (http://cgi.pathfinder.com/drweil/) and *Go Ask Alice!* (http://www.goaskalice.columbia.edu/) enable patients at home to ask questions about their health and to receive personal answers directly
from a real physician. DiagnosticDoc (http://www.diagnosticdoc.com/) let patients and physicians contribute to message boards on specific health issues. Internet startup Healtheon (http://www.healtheon.com) has seen dramatic jumps in its stock value as the market speculates positively on the future of the Internet as a solution to storing and accessing an unmanageable number of medical records in the United States [6]. The Internet points to the promise and hope of health care leaving hospitals and connecting to the home.

2.5.0 Recent Progress in Bio-Monitoring

The following sub-sections describe a number of initiatives to make consumer level health-monitoring a reality. These include bio-monitoring systems capable of recording vital signs during a number of rigorous activities, individual sensor suites for monitoring specific signs, and infrastructures that implement health monitoring.

2.5.1 Polar

In 1982, a Finnish company, Polar, introduced a heart rate monitor that is still the most popular consumer bio-monitoring device on the market today. Since then, Polar has expanded their product line to a suite of software, training guides and other tools to accompany their monitors for amateur and professional training.

A Polar system consists of a strap worn around the athlete’s chest and a wristwatch receiver. The Polar chest strap measures voltage signals from the heart and interprets them as pulses of the heart. The receiver functions both as a storage unit as well as a display unit. While a Polar monitor can record data and store it on a user’s home PC, it does not currently make that data available to the user’s doctor or communicate with any other health database [7].

2.5.2 MarathonMan

In 1997, researchers in the Personal Information Architecture group at the MIT Media Lab set out to build a bio monitoring black box for use in the Boston marathon. This device was capable of recording heart rate, foot cadence, location (GPS), and core temperature using commercially available

---

1 Airliners contain devices that record anything that may help determine what went wrong following a crash. These devices are called black boxes.
2 The Global Positioning System allows handheld receivers to determine location on earth via satellite signals.
sensors. The data being recorded by this device was relayed in real-time to a network location where Internet users could follow the runners' progress.

This system was perfected during the San Francisco and New York marathons that same year, when it became clear that it was possible to assess failure of a runner from the real-time data [2]. No automated feedback was available to the runners while they were running the marathon. The data was relayed to a remote station where a controller could speak his interpretation of the data to the runners via cell phone.

GPS. Point A is at mile 13 (halfway)—where the runner dropped out. This is the same as point C to the right.

Steps versus Time. Note the breaks in the runner's stride at points B and C.

Michael Hawley's foot failure in the 1998 San Francisco Marathon.

2.5.3 US Army Rangers

The United States Army lost two soldiers during training in 1996 due to hypothermia. Having successfully completed and tested the marathon monitoring system, the Personal Information Architecture team joined with researchers from the US Army Natick Laboratory to create a system capable of monitoring soldiers in training in hopes of preventing such tragedies in the future.

This system was almost identical to the marathon system except that it was housed in a more robust casing to withstand more difficult conditions and it was unable to broadcast its data. The Army's goal was to gather data for

---

3 Polar heart rate monitor, FitSense prototype cadence sensor, Trimble GPS, and a Body Core Temperature Monitor engineered by DARPA.
future analysis, and therefore storing all of the data on the sensor pack was acceptable.

One interesting technology added to this system was a mathematical inference engine\(^4\) written by Matthew Gray at the MIT Media Lab [8] that enabled a computer to make reasonably accurate assessments about what particular activity a soldier might have been engaged in based on his monitor data.

![Graph showing probability of soldier being in an assault situation over time.](image)

*Mathematical inference engine predicts soldiers' activity.*

### 2.5.4 1998 Everest Expedition

Since Sir Edmund Hillary first reached the summit of Mount Everest in 1953 [9], scores of climbers have continued to attempt the same climb every year. In 1997, the pursuit of this legendary summit turned tragic when 5 climbers lost their lives on a single night [10]. Questions arose as to how such tragedies could be avoided with technology.

Researchers from the Media Lab teamed up with researchers from Yale University to test health monitoring and telemedicine systems\(^5\) for use in remote locations during the May 1998 summit season on Everest. The monitoring system used on Everest collected data about blood oxygen levels

---

\(^4\) Matthew Gray has documented his research at [http://www.media.mit.edu/~mkgray/army/](http://www.media.mit.edu/~mkgray/army/).

\(^5\) The Yale University team tested a prototype portable 3-D ultrasound system. [http://yalesurgery.med.yale.edu/telemedicine/telemedicine.htm](http://yalesurgery.med.yale.edu/telemedicine/telemedicine.htm)
as well as temperature inside, on the surface of, and around the climbers. The Everest system also gathered non-bio data such as location, humidity, and ambient light.

The Everest system did not have an interface allowing the climbers to see their vital signs. The data was relayed/recorded to remote workstations where doctors could potentially interpret the data for them. In this case, the real-time relaying of that data did not work due to transmission problems, and therefore the data was not available to anyone until after the climbers had returned [3].

2.5.5 FitSense

FitSense, a startup in Wellesley, MA, engineered a commercial system that will monitor the vital signs of athletes, display that data on a wristwatch, and relay that information to the Internet. Their product has not yet reached the marketplace, though many of its components were tested as parts of the Media Lab marathon system.

The FitSense system uses a wireless networking system for connecting each of the sensors distributed over an athlete’s body, storing that data and transmitting that data. The current prototype requires hardware to be attached to the user’s sneaker and mounted elsewhere on the body [11].

2.5.6 FitLinx

FitLinx is currently marketing a system to be attached to exercise machines. Their Training Partner networks the machines and enables them to collect repetition data. The FitLinx system gathers data only from the machines—not from the people using them. It remembers how many pounds you lift or how many minutes you run on the treadmill, but not what your heart rate was or how much oxygen is in your blood [12].

2.5.7 Toto Toilets

Toto Corporation markets a toilet in Japan that performs an analysis of the user’s urine. The toilet’s armrest records body temperature, pulse and blood pressure. Collected data is sent by modem to a doctor’s office for analysis, the results of which are available on-line when the user has returned to his PC. This toilet is available only in Japanese markets right now [13].
2.5.8 Tanita Scales

One common benchmark of overall health is weight and body fat composition. According to the latest government statistics, over 30% of the American population are overweight. One of the difficulties in assessing weight is that a single user's weight may vary by upwards of 5 pounds in a single day.

Tanita Corporation sells a line of scales that measure the electrical resistance through the user's body to determine body fat composition. Many of Tanita's scales can record data and communicate with desktop PC's. The scales show the user's current weight, however they do not have displays complex enough to show history, nor the ability to bring any other information to the user [14].

2.5.9 Ventus

Many asthma patients need to take regular inhalations of a preventive medicine. Often patients miss inhalations or they take too much medicine.

Working at the MIT Media Lab, Joseph Kaye has produced an "intelligent" inhaler called Ventus. The Ventus inhaler remembers how many doses have been disbursed each day and when the doses were disbursed. Each inhaler is programmed to know when the patient should be using the inhaler and can give warning when the inhalation time arrives. The inhaler is connected to a PC at the end of each day so that its history of use can be stored or its schedule adjusted [15].

2.5.10 AirChart / LifeChart

ENACT Health Management Systems released the first home monitoring system for asthma patients in November of 1998. ENACT's system requires patients to blow into the AirWatch device, a pocket sized breathing monitor. The AirWatch monitor displays a real-time reading to the patient and stores its test results until it is connected to a PC. A co-product, the LifeChart network, transmits all of the data to the patient's pharmacist or doctor as well as returns the latest news about asthma to the patient [16].

---

*The Things That Think consortium at the Media Lab has conducted ongoing explorations of object intelligence—by allowing using computers to see and act upon relationships between networked objects.*
2.5.11 LifeLink
Ambulances in San Antonio, Texas have been using wireless connectivity to make it possible for physicians and EMT's to know about patients' vital signs and medical histories en route to hospitals. Using the LifeLink system, waiting physicians receive video images of the patient's body while in the ambulance. Physicians can control the video camera remotely to zoom or pan over the patient. Furthermore, vital statistics are also gathered and transmitted with the video signal. While this system is not a consumer system—nor is it intended to become one—it certainly illustrates the promise of telemedicine [17].

2.5.12 Guardian Angel
There are many initiatives to unify the way patient records are stored and accessed using the Internet. The Lab for Computer Sciences at MIT has been developing the Guardian Angel Patient-Centered Health Information System since 1995. The system architecture is designed with a bent towards communication with PDA's and other distributed computing devices that may monitor patients outside of the hospital. The Guardian Angel system would facilitate the keeping of records relating to all health information as well as have the intelligence to check and interpret the data it gleans. As stated in the Guardian Angel research manifesto:

"We believe, with many others, that active monitoring of accurate and comprehensive information about an individual's lifetime medical history can greatly improve the quality of medical decision making for that person, reduce errors in health care, and allow people to make better personal decisions and commitments about their care." [18]

---

7 Personal Data Assistant.
### Recent projects in personal health monitoring.

#### 2.6.0 What is Missing?

One feature missing from most current health monitoring systems is compelling and clear visual output. The few health monitors that do allow us to see their data in real-time are limited by unsophisticated displays, hardwired to show us only the smallest portion of their data they receive.

While it is obvious that physicians would benefit from knowing more about the state of their patients’ health, patients too can benefit from this knowledge which would enable them to proactively improve their health.

Another shortcoming of many health monitors is their lack of network access. As health care incorporates some of these technologies, it will be important to determine how these devices will be networked together. Devices that can sense us but which can not contribute what they know to a larger database are limited in their use.

The prototype system described throughout the following sections addresses these shortcomings. It has a display system that is flexible enough to let us see any data about ourselves, from any source. Its architecture allows any number of sensors to be added to it, and the information from those sensors is made available to other networked devices and services.

<table>
<thead>
<tr>
<th>Project</th>
<th>Bio-Input</th>
<th>User Display</th>
<th>Network access</th>
<th>On the Market</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polar Heart Rate Monitor</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Closed display architecture</td>
</tr>
<tr>
<td>Marathon Man</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>Data interaction over WWW</td>
</tr>
<tr>
<td>US Army rangers</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1998 Everest Expedition</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Data interaction over WWW</td>
</tr>
<tr>
<td>FitSense</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>pre-beta</td>
</tr>
<tr>
<td>FitLinxx</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Records device history only.</td>
</tr>
<tr>
<td>Toto toilet</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Not available in U.S.</td>
</tr>
<tr>
<td>Tanita scale</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ventus</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>Records device history only.</td>
</tr>
<tr>
<td>AirChart / LifeChart</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Only monitors breathing.</td>
</tr>
<tr>
<td>LifeLink</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>Not for Consumer use.</td>
</tr>
<tr>
<td>LCS Guardian Angel</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>Protocols only.</td>
</tr>
</tbody>
</table>

---

17
Furthermore it is built as an environment—a space for interacting with the data that describes our health and observing how that data relates to our bodies and our health. It is a space that is not dedicated to information, like the home PC currently is. One of the lessons learned from previous research on the Media Lab MarathonMan project was that examining bar graphs on a workstation monitor was useful to scientists, but hardly compelling or revealing to the casual, consistent user. After coming home from a run, sitting down at the PC simply wasn’t a natural progression. Instead, runners wanted to head straight for the shower. This thesis proposes a system through which a jogger’s run could be deconstructed while he or she undressed, giving the jogger a chance to think about the results in the shower. The system described here makes it possible to access the information away from our computers and within the spaces that we normally inhabit.
3.0.0 Inner View

3.1.0 Introduction

In searching for an environment to examine how people can interact with health monitor data, one of the most immediate objects to come to mind is a bathroom mirror. There are fundamental reasons for this:

First, many immediate signs of health do not require health-monitors—just our eyes. We can see how toned or how flabby we are, how wrinkled our skin is, or how bald our heads are. With training we can read our posture or identify rashes. Since many of us look in the bathroom mirror after we undress and before we get in the shower, the bathroom mirror gives us a chance to see and explore without being hidden behind our clothes.

Interestingly, people often depend on mirrors to see parts of themselves they cannot normally see, such as their backs but this is hard. This is conveniently parallel to the way health monitors let us see things about ourselves that we normally cannot see. Because Inner View is connected both to health sensors as well as to cameras, and because it has the ability to augment what we normally see in the mirror with other information, it can let us see things like our heart beating or our lungs expanding.

Visual insight aside, bathroom mirrors often hold health-related products behind them. How convenient it would be if our bathroom mirror was conscious of the bottles of medicine and beauty products it was holding. Imagine a mirror that could tell you what medicines belonged to who, when they expired, and what warnings should be heeded. Inner View is capable of this.

3.2.0 Built Prototype

A few design options were considered prior to building the Inner View prototype. The first option considered was to use a camera and projector as a faux mirror. The camera would be pointed at the user through a hole in a rear projection screen, or from a correctable angle on the periphery of the screen. The advantage of such a setup would be that graphical data could be merged with the camera’s video-stream as seamlessly as pixels and processing allow—and, as Hollywood has shown us in the past decade, video manipulation can be pervasive.
There are a few downsides with this setup. Most significantly, the projection would necessarily have to work around the camera so as not to obstruct rearward projection. Also, the video projection would easily be recognizable as video rather than a true mirror because of issues of pixelation, distortion, etc. which are much more noticeable in a scenario where your eyes are very close to the projection surface. Finally, the realism of the image is limited by the quality of the camera and the projection technology.

A second option considered was to use a camera in conjunction with a large flat panel display. At the time of this writing, Toshiba had just started shipping a 40.0" x 24.9" x 3.5" plasma display panel\(^8\). Unfortunately, large flat panel displays are prohibitively expensive, have limited pixel resolution, and would be difficult to modify for the desired use here. Placing a camera, infrared, and other visual sensors behind the panel become difficult tasks. Furthermore, the display resolution is coarse so that the pixelation and distortion issues inherent in the previously described setup would still exist.

\(^8\) A Toshiba PD42WI Plasma display currently costs $14,995.00 and has a pixel resolution of 852 x 480.
The actual Inner View prototype implementation uses a sheet of half-mirrored glass with projection on the rearward side. A half-mirrored glass is both reflective and translucent, depending on the difference in lighting levels between the front and back of the mirror. With Vellum affixed to the back of this mirror, ambient light is reduced from the rearward side to the point that the glass behaves more or less like a regular mirror. When projecting on Vellum, its surface becomes bright enough that projection images become superimposed on the reflected image.
One main advantage of this setup is the simple fact that it is a real mirror and as such, it satisfies our expectations of what a mirror is and does physically. Another advantage is that because the reflection is real, no processing is necessarily required to augment a video signal with other information. Desired images can simply be projected onto the reflection.

There are a few disadvantages. First, luminance is lost through the mirror, leaving colors dull. Also, the pixelation of the image contrasts greatly with the clarity of the natural reflection. Moreover, parallax becomes an issue if augmented images are to be tightly integrated with the reflection image. Finally, rearward projection requires a long distance between the projector and the projection surface in order to cover a reasonable area of the projection surface. To maximize this distance without making the depth of the setup too awkward, a second mirror was used to reflect the image onto the half mirror, so that the projector can sit 90 degrees from the surface. In a real situation, depth into a wall would be much less achievable than depth along a wall. Regardless of the disadvantages, the expectation users have for a digital mirror to be a mirror and the ability of this setup to satisfy that expectation were overwhelmingly in favor of making this the most ideal setup.

### 3.3.0 Inner View System Architecture

This section describes the Inner View software architecture. The software architecture has three levels—a device level, an agent level, and a display level. By separating the devices, the data interpreting agents, and the display engine, it is easy to deal with the addition of any device, to change the way a device is utilized, and to re-implement the final output of the system.
3.3.1 Device Level

The device level includes code for reading from the prescribed devices. The device level is unaware of the type of devices that are being read from. *SerialReaders* are told the baud rate to read at and the UNIX device name to read from, and continue to do so until the user resets the system.

3.3.2 Agent Level

Agents are semi-autonomous software entities that act upon data that they see [19]. The Inner View agents examine the incoming data and determine what kind of data they are and how that data should be parsed. Some agents may also write the parsed data to remote locations or manipulate the display level to reflect the new data.

Inner View is currently running on an SGI O2 with an R5000 processor. While this system runs a relatively slow implementation of the Java Virtual Machine, it is optimized for vision and rendering, both of which are more sensitive to processor speed than the code required to read and to interpret the health data.

The code for Inner View is written in JAVA, so as to be easily integrated with the *Hive* architecture currently being developed at the MIT Media Lab. *Hive* enables many disparate hardware devices to be connected easily over a robust network. (See page 35.)
3.3.3 Display Level

The display level is a graphics toolkit tailored to present data information and allowing for real-time manipulation of that data. The current display code allows images to be added to the display and for those images to take on animated behaviors. For example, an image of a heart can be placed in the display, then panned, zoomed, and scaled to make it seem to follow the user or to beat. Images can be removed from the display or updated with a single method call.

3.4.0 Demonstration

In choosing demonstration material for Inner View, three distinct modes of representation were considered. First, how can the user’s image be understood and interpreted in the space of a mirror? Second, how can physical objects be interpreted? Third, what kind of latent information can be accessed and displayed on a mirror?

<table>
<thead>
<tr>
<th>Modes of Representation</th>
<th>Demonstration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpreting your image</td>
<td>As you brush your teeth you see faux plaque germs get brushed away.</td>
</tr>
<tr>
<td><em>What does Inner View see?</em></td>
<td></td>
</tr>
<tr>
<td>Interpreting physical objects</td>
<td>An image of a heart represents a heart rate sensor. The heart pulses in sync with the data stream coming from the sensor.</td>
</tr>
<tr>
<td><em>What do Inner View's sensors sense?</em></td>
<td></td>
</tr>
<tr>
<td>Data from external sources</td>
<td>Warnings and expiration dates about medicine in a medicine cabinet are displayed.</td>
</tr>
<tr>
<td><em>What does the Network know?</em></td>
<td></td>
</tr>
</tbody>
</table>

*Modes of Representation.*

3.4.1 How Data Feels

A well schooled musician can read a tempo marked 112 beats per minute (bpm) and have a very clear understanding of exactly how fast he should be playing. This kind of innate sense of timing is acquired through years of experience. To the average person, 112 bpm is not as immediate because he likely has to compare it to something—for example a clock—before he can begin to feel what that tempo might be.
How much more informative is labeling a user’s heart rate as being 112 bpm? The goal behind this demonstration is to represent a heart rate data-stream in a way that provides something you can feel or experience and thereby helps you to acquire an intuitive sense of what it is.

3.4.2 Polar Heart Rate Monitor

To show this, I chose to connect a Polar heart rate monitor to Inner View. The Polar heart rate monitor consists of a receiver hidden behind the mirror and a chest strap worn by the user. Imagine that if you held the chest strap in front of the mirror you saw a little heart superimposed on the chest strap. As you place that strap on your chest, the heart follows the strap to your chest and then begins to beat in sync with the pulse rate it is reading.

The Polar heart rate monitor system has been used extensively within body monitoring research at the lab because it is wireless, accurate, stable, and easy to work with. To connect the Polar receiver to Inner View, an iRX board containing a PIC chip was programmed to read the receiver data. The receiver already outputs ASCII values, so the only necessary task of the iRX board is to adjust the output signal to TTL levels and to output them serially [20].

3.4.3 Polar HRM Agent

The Polar HRM Agent reads the values and then causes a graphic heart in the mirror to pulse. The heart only pulses per reading, and modulates between zero dimensions and a full size image. In this way, if the monitor is removed from the user’s chest, the Heart stops at its zero dimensions and thus appears to have disappeared seamlessly.

It is the task of the Polar HRM Agent to store and access long term heart rate data as well. The format for storing heart rate data is similar to the format used by Net Weight to store weight data. Samples are stored every pulse.
Polar data fields: device / user ID / date/time (GMT) / average heart rate for one minute.

3.4.4 Pill Bottles and History

A bathroom mirror with image augmentation ability, network access, and sensors could know what things were in its shelves and it could tell you what it learned about those things from other networked information sources. This demo requires RF tag reader and separate RF tags for each object to be identified on the shelves, agents for identifying the tags, and an agent for augmenting the image. Because Hive already implements RF identification and because the resulting demonstration may be fairly obvious, this demo is being left as a future project once the Inner View code is adapted to communicate with a Hive server.
3.4.5 The Magic Mirror

The magic mirror in Disney's 1937 release *Snow White and the Seven Dwarfs* has become a cultural icon in America [21]. The notion that a mirror could be more like a window through which some magical entity could see us and show us things is a powerful fantasy.

Can Inner View fulfill such a fantasy? Inner View has full display capabilities. It also has the ability to sense people, and, with a hidden camera, it can see them. Because Inner View already utilizes a powerful computer to manage its display software, it seems logical to examine an interactive magic mirror. Although there was not time to complete a fully interactive system by the time this document was written, the following storyboard presents a likely scenario for such a tool:
3.4.6 Interval Research Corporation's Magic Mirror

One of the reasons completion of this demo is not being pursued rigorously is that it will involve addressing artificial vision problems that are beyond the scope of this thesis, and more importantly, problems that are being solved elsewhere by more directed efforts. Interestingly, Interval Research Corporation has already constructed a “magic mirror” demonstrating their artificial vision technology. *Mass Hallucinations* uses cameras to track users’ faces. It then applies graphic distortion to the areas where those faces are in the image, based on the depth they are perceived to be from the mirror. The distortions happen in real time, fulfilling their goal of creating a system through which users can naturally control a display without use of any mouse or keyboard [22].

3.5.0 What Will the Future Look Like?

While the Inner View prototype has been designed to be a bathroom mirror, it could have been a mirror anywhere. It could have been a mirror in a gymnasium, a weight room or in a hallway. Or it could not have been a mirror at all: perhaps its data could have been presented on the front of a refrigerator. One goal in creating Inner View has been to show that we should be able to access information from almost anywhere in our environment, just as mirrors are objects that we simply choose to hang on any wall we please, or to build into our cars.
Some day perhaps there will be unexpected visual interactions in any piece of glass— or even paper— that we desire. In the meantime, one of the most satisfying experiences in creating Inner View has been seeing numbers of passersby stop next to it out of the astonishment that graph images were dancing within a piece of glass that is very similar to what they see around them almost any day.
4.0.0 Net Weight

4.1.0 Introduction

When a person stands on a scale, it is likely that his thoughts are about how fat or thin he is. For some people this may be vanity, for others it is critical to their survival. Obviously, for most it is something in between.

The most commonly trusted device for measuring weight is the scale. Ironically, the single measurement that most people use a scale to make—an isolated point of weight data— is unlikely to be generally useful. To begin with, the average human being’s weight varies by as much as 5 pounds during a single day. More importantly, in cases where weight is either a measure of healthiness or the result of a special diet, real fluctuations in health happen over days or weeks. Daily average is more likely the necessary statistic, and gathering daily averages over long periods of time is the most meaningful function to a person. An ideal scale then would measure weight and remember weight. This way weight could be examined by daily average, or normalized for time of day, etc.

4.2.0 Built Prototype

The Net Weight prototype is a modified commercially available bathroom scale. The platter of the scale is a quarter inch thick plate of glass, supported from the center. The center is seated on the end of a steel cantilever. The deflection of this cantilever is measured to determine the user’s weight. In order to “tare” the scale, a microphone listens for audible “bumps.” Upon hearing a bump, it wakes up, measures the zero state, and then waits for the user to stand upon the scale. We added a serial output to the scale. Measurements from the scale are read serially at a fast enough rate (9600-baud) that the deflection can be represented in real-time. The data is read as a serial source and then parsed by a Net Weight Agent.

---

9 Evidence suggests that heart risk patients may show noticeable gain in weight prior to heart failure.[23]
4.3.0 Toe Scanning

The choice to use a scale with a glass platter resulted from both the need of a local identification service as well as from the desire to explore other things a scale could do intrinsically. In particular, what is that a scale can see and know other than how much weight is sitting upon it—especially given that it is in direct contact with the user? Because the user stands on a glass platter, it is possible use a camera to capture images from beneath the scale. With the focal point of the camera set to the plane of the platter’s surface, it is possible to capture an image of the foot of anyone standing on it. Would it be possible to do a rudimentary “toe scan” from such a setup?

The bottom of the platter is a plane 2.25 inches from the level of the floor. Finding a digital camera that can do exacting macro work with a wide-angle view that needs less than 2 inches distance from the target (assuming that the length of the camera is no more than a quarter inch thick) was not possible. Two alternatives were probed. The first alternative was simply to raise the scale platform another 4 inches from the floor. This proved awkward, as standing on the scale felt very unnatural. The second alternative was to place a mirror was placed at 45 degrees beneath where the user’s toes would are. This is less awkward, but makes the setup less ideally hermetic. Using the latter setup, the following images were taken of several users’ toes:
What the camera sees from beneath the scale and four different users.

We hypothesized that, for a small group of users (a family of five), we would be able to discern any given user by comparing, both their weight and their foot trace.

After working with this system for several weeks, it became apparent that there were some mechanical problems that needed to be solved before the identification could be made. Our ability to obtain accurate toe scans was hindered by our inability to control the ambient lighting around the scale, as well as by variations in humidity causing test users' feet to spread out more or less on the platter. Because creating a new identification was not within the central focus of this thesis—especially since Hive already has other devices available for user identification—this research was eventually dropped. It seems likely that such an identification system could work if the mechanical problems were solved.
Multiple scans of the same user/setup on different days.

4.4.0 The Net Weight Agent

The Net Weight Agent does three things: it parses data from a data-stream, it attaches other information to that data (time, date, and user) and it adds or manipulates objects in the Inner View display. It is also the job of the Net Weight Agent to read and write data over the network to remote server.

Net Weight Agent flow chart.

When the Net Weight Agent reads a data point, it first determines what the date and time of the data point’s recording are from the computer’s clock. Then the data point and associated information are written to a file located on the network. At startup, that file is read in its entirety. The Net Weight Agent averages each data point with all data points for the same day for each individual. The Inner View representation is given a years worth of data points (or whatever data there is if less.) The data is presented as a simple line graph. The most current data point is represented as a bar at the end of this graph.
Author's weight for 2/99 through 5/99. The gray data to the left is computer generated to fill out the year for representational purposes.

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTWT</td>
<td>GEILFUSS</td>
<td>Apr 09 10:56:12</td>
</tr>
<tr>
<td>NTWT</td>
<td>GEILFUSS</td>
<td>Apr 09 10:56:32</td>
</tr>
<tr>
<td>NTWT</td>
<td>GEILFUSS</td>
<td>Apr 09 10:57:16</td>
</tr>
<tr>
<td>NTWT</td>
<td>GEILFUSS</td>
<td>Apr 09 10:57:33</td>
</tr>
</tbody>
</table>

*Net Weight data fields: device / user ID / date/time (GMT) / weight.*

The data stream from Net Weight is fast enough that real-time representations of the weight will show accurate fluctuation similar to an analog needle. From a design standpoint, being able to represent the stream as something that moved from zero to $n$ is in keeping with observing a very tangible representation of the data stream as described in the previous section. The most current data point is displayed as the real-time output of the scale.

### 4.5.0 Weighing the Future

The average scale does what we expect it to— it tells us how much we weigh. Net Weight's strengths are that unlike a scale it can *remember* what it tells us, and that it can remember things we did not realize it knew, like what the bottom of our feet look like. More importantly, the things it remembers can be communicated to other objects around our environment. Some day perhaps every object will have sensors enabling them to remember the few things they "see". Some day perhaps every object will be able to communicate what they sense to other objects. In time it is likely that we will take much of this for granted the same way we can now take it for granted that a good sofa cushion remembers the shape of our bottom or a good shoe remembers the shape of our foot.
5.0.0 Continuance and Conclusions

5.1.0 Continuing the Current Prototype

As a prototype, and as continuing work, there are improvements that should and will be applied to Inner View and Net Weight. The following sections detail some likely future directions.

5.1.1 Hive, Kitchens, and Bathrooms

A significant development at the Media Lab in 1998-99 has been the creation of Hive, a toolkit for connecting “smart” objects to a global network. Hive includes protocols and libraries for enabling many disparate objects to communicate over a robust network model. The system also allows users to tailor the output of events on their individual desktop systems [24].

Media Lab students Matthew Gray and Andy Wheeler have been applying Hive to the task of creating an intelligent kitchen. The Counter Intelligence kitchen connects many sensors and appliances so that it can keep track of what ingredients and how much of each ingredient is contained within its cupboards, access recipes from the network, and control appliances.

Because Counter Intelligence uses Hive, it has access to many devices that are not normally attributed to the kitchen—for instance a child’s talking toy. Counter Intelligence can potentially use the toy as a mode of interaction to teach a child how to cook using only “child safe” recipes. The toy may playfully tell the child where to find the right ingredients, which bowl to pour them into, when to stop pouring [25].

As a future project, rewriting some of the Inner View/Net Weight software to take advantage of Hive could be beneficial. What one eats is an obviously significant factor towards health. The ability to access what a kitchen may know about your consumption habits, or the ability to make recommendations to an intelligent kitchen of recipes may be beneficial to persons of particular health risks and could be a powerful application.

5.1.2 Glucose Sensors

Diabetics must keep the level of glucose in their bloodstream within very tight boundaries, undergoing pinpricks several times a day to get blood samples for analysis by at-home sensors. In an effort to ease this process,
dozens of companies are trying to bring non-invasive glucose sensors to the market. Because diabetics must be hyper-aware of their health state and keep careful track of their diet, a system with sensors and access to nutritional information may be extremely valuable to them [26].

5.1.3 Net Weight: the Next Prototype

A future consideration for Net Weight is to explore recording the user’s current image with each weight recording. The goal of this exploration would be to see if it is beneficial to play back weight information with their correlated images. Most likely only the user’s face would be stored because clothing may mask weight effects on user’s bodies. This may be a very beneficial tool towards understanding how weight affects our bodies, however it touches very close to debates about weight, cultural expectations, and self esteem.

5.1.4 Inner View: the Next Prototype

In its current state, Inner View shows the potential for rich health interactions and learning possibilities. Many users have expressed appreciation simply for the images dancing inside of a real mirror, before even discovering that it could sense or communicate with them.

While it is currently unrealistic that the average consumer could afford a mirror with expensive display technology and the latest graphics workstation behind it, display technology is advancing rapidly. It is not too farfetched that a future incarnation of Inner View could be created from an inexpensive large flat panel display and an array of embedded minute video cameras for health conscientious consumers.

5.2.0 Conclusions

5.2.1 Preventing Disease

The single most costly disease in America is heart disease. In 1997, $36 billion was spent on hospital and nursing home expenses to cover heart disease patients. Of this, $6 billion was spent on emergency room patients. Interestingly, 70% of the patients admitted to the emergency room were subsequently proven not to have heart disease [27]. Health monitoring can lead the way towards better predictability and assessment of health conditions. Furthermore, health monitoring enables patients to monitor their lifestyles— their weight, their consumption, their exercise, etc.— which,
according to the National Institute for Health is the single most important factor in reducing risk towards heart disease. [28]

5.2.2 Monitoring the Elderly

Following heart disease, one of the most important problems bearing down on the American health care system is how to care for the increasing percentage of the population over the age of 65. The Robert Wood Johnson foundation has been applying its philanthropy towards improving care and support for people with chronic health conditions [29]. Their 1999 statistics include the following chart showing a dramatic increase in the number of adult day care centers required to support for aging Americans with medical conditions. Health monitoring will serve as an important supplementary resource when the supply of adult day care centers can no longer keep up with the number of elderly requiring care.

5.2.3 Staying Out of the Hospital

The average hospital stay has been steadily decreasing. Some figures place hospital stays at 20% shorter than they were 10 years ago [30]. Many experts claim this is because U.S. health care policy has made shorter hospital stays a goal, and that as a result hospitals have generally become more efficient [31]. Regardless of these facts, patients are spending less time in the hospital and more time at home. Post recovery periods are clearly a place in which health monitoring will play a big role.

5.2.4 Health at Home

All of these issues point towards how critical home health monitoring will become in the near future. Doctor/Patient relationships will see a remarkable transformation as more information is collected while they are away from each other than during routine visits, and patients will continue to
turn to the Internet as a source of health information. Devices will enter our homes that will integrate information about our health with our lifestyles. Most importantly, as the prototypes in this thesis have shown, future health monitoring systems must have the ability not just to inform our doctors, but to empower us to improve our own health.
6.0.0 References

[1] Zeeshan Husain, DPM


[26] The National Institute of Diabetes and Digestive and Kidney Diseases
http://www.niddk.nih.gov/

http://www.compbase.com/press.htm


http://www.ncpa.org/~ncpa/ba/ba141.html
Class BioPanel

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Image;
import java.awt.Graphics;

public class BioPanel extends java.awt.Panel implements Runnable{
    ImageLayer[] layer;
    Thread paintthread;
    Image offimage;
    Graphics offgraphics;
    Dimension offdimension;
    DPoint[] on, off;

    // ------------ Constructos ------------
    public BioPanel( int layers ){
        layer = new ImageLayer[layers];
        on = new DPoint[6];
        off = new DPoint[5];
        for (int foo = 0; foo < 5; foo++){
            on[foo] = new DPoint(.2 * foo, .2 * foo);
            off[4 - foo] = new DPoint(.2 * foo, .2 * foo);
        }
        on[5] = new DPoint(1, 1);
        start();
    }

    // ------------ Methods --------------
    public void paint( Graphics g ){
        Dimension d = getSize();
        if ( ( offgraphics == null ) ||
            ( d.width != offdimension.width ) ||
            ( d.height != offdimension.height )){
            p("create graphics context");
            offimage = createImage( d.width, d.height );
            offgraphics = offimage.getGraphics();
            offdimension = d;
        }

        // ------------ erase --------------
        offgraphics.setColor(this.getBackground());
        offgraphics.fillRect(0, 0, d.width, d.height);

        // ------------ images scaled and panned ----
        for (int foo = 0; foo < layer.length; foo++){
            try {
                int w = (int)( layer[ foo ].image.getWidth(this) *
                    layer[ foo ].scale.x );
                int h = (int)( layer[ foo ].image.getHeight(this) *
                    layer[ foo ].scale.y );
                int x = (int)( layer[ foo ].shift.x - (w / 2) );
                int y = (int)( layer[ foo ].shift.y - (h / 2) );
                offgraphics.drawImage( layer[ foo ].image, x, y, w, h, this );
                layer[foo].advance();
            } catch (Exception e){
                //p("paint: * foo + * + e);
g.drawImage( offimage, 0, 0, this);
}

public Graphics getLayerGC( int which ){
    if ( isLayer( which ) )
        return layer[ which ].image.getGraphics();
    else
        return null;
}

public void initLayer(int which, int w, int h){
    if ( isLayer( which ) )
        layer[ which ] = new ImageLayer();
        layer[ which ].image = createImage( w, h );
        layer[ which ].shift = new DPoint( 0, 0 );
        layer[ which ].scale = new DPoint( 1, 1 );
}

public void hiddenInit( int which, int w, int h ){
    if ( isLayer( which ) ){
        initLayer( which, w, h );
        layer[ which ].scale = new DPoint( 0, 0 );
    }
}

public void scaleLayer( int which, DPoint percent ){
    if ( isLayer( which ) )
        layer[ which ].scale = percent;
}

public void scaleLayer( int which, DPoint[] percents ){
    if ( isLayer( which ) )
        layer[ which ].scaleq = layer[ which ].fillQ( percents );
}

public void shiftLayer( int which, DPoint increment ){
    if ( isLayer( which ) )
        layer[ which ].shift = increment;
}

public void shiftLayer( int which, DPoint[] increments ){
    if ( isLayer( which ) )
        layer[ which ].shiftq = layer[ which ].fillQ( increments );
}

public void showLayer( int which ){ if ( isLayer( which ) )
    scaleLayer( which, on );
}

public void hideLayer( int which ){ if ( isLayer( which ) )
    scaleLayer( which, off );

private boolean isLayer( int which ){
    if ((( which > -1 ) && ( which < layer.length ) )
        return true;
    else
        p("layer [* + which = *] does not exist.");
        return false;
    }

public void update(Graphics g){
    paint(g);
}

public void run(){
    while ( paintthread != null ){
        repaint();
        try {
            Thread.sleep( 10 );
        }
    }
}
catch ( Exception e ){
  p( "run: " + e );
}

public void start(){
  paintthread = new Thread( this, "paint" );
paintthread.start();
}

public void stop(){
  paintthread = null;
}

public void p( String in ){
  System.out.println( "BioPanel: " + in );
}


Class ImageLayer

import java.awt.Image;
public class ImageLayer extends Object{
  Image image;
  DPoint shift;
  DPoint scale;
  Queue scaleq;
  Queue shiftq;

  public ImageLayer(){
  }

  public boolean advance(){
    boolean changed = false;
    if ( scaleq.hasElements() ){
      scale = ( DPoint )( scaleq.getLast() );
      changed = true;
    }
    if ( shiftq.hasElements() ){
      shift = ( DPoint )( shiftq.getLast() );
      changed = true;
    }
    return changed;
  }

  public Queue fillQ( DPoint[] points ){
    Queue q = new Queue();
    for ( int foo = points.length - 1; foo > -1; foo-- )
      q.addElement( points[ foo ] );
    return q;
  }

}

Class InnerView

import java.awt.Frame;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.MediaTracker;
import java.awt.Toolkit;
import java.awt.Image;

public class InnerView {
  public static void main(String args[]){
  

if (args.length < 1) {
    System.out.println("Usage: java InnerView [dev] [dev]...");
    System.exit(0);
}

SerialSource[] ss = new SerialSource[args.length];
for (int foo = 0; foo < args.length; foo++)
    ss[foo] = new SerialSource(args[foo], SerialSource.READ_LINES);

BioPanel bp = new BioPanel(3);
bp.setBackground(Color.black);

// 0 = heart
// 1 = netweight long term
// 2 = netweight now
NetWeightAgent nwa = new NetWeightAgent(bp, ss);
PolarHRMAgent phrma = new PolarHRMAgent(bp, ss);

Frame iv = new Frame("InnerView");
iv.setSize(800, 600);
iv.setLayout(new GridLayout(1, 1));
iv.add(bp);
iv.setBackground(Color.black);
iv.show();
}

Class NetWeightAgent
import java.awt.Frame;
import java.awt.Choice;
import java.awt.Label;
import java.awt.GridLayout;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import java.awt.Image;
import java.awt.Color;
import java.awt.Point;
import java.awt.Toolkit;
import java.awt.MediaTracker;
import java.awt.Graphics;
import java.util.GregorianCalendar;
import java.util.Calendar;
import java.util.Date;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import Serialio.SerialConfig;
import Serialio.SerInputStream;
import Serialio.SerialPortLocal;

public class NetWeightAgent implements Runnable, ItemListener {
    final long YEAR = 31536000000;
    final int CALIBRATION = 15;

    Frame f;
    Choice c;
    BioPanel bp;
    SerialSource[] ss;
    Thread readthread;
    int watch;
    int weight = 0;
    int current = 0;

    protected static PrintWriter pw;
    int daily_weight[];
public NetWeightAgent (BioPanel bp, SerialSource[] ss){
    this.ss = ss;
    this.bp = bp;
    watch = 0;
    c = new Choice();
    c.add("no device selected");
    for (int foo = 0; foo < ss.length; foo++)
        c.add(ss[foo].device);
    f = new Frame("NetWeight Agent");
    f.setSize(200, 75);
    f.setLayout(new GridLayout(1, 1));
    f.add(c);
    f.show();
    c.addItemListener(this);
    daily_weight = new int[365];
    for (int foo = 0; foo < daily_weight.length; foo++)
        daily_weight[foo] = 0;
}

public void itemStateChanged(ItemEvent evt){
    Object target = (Object)evt.getItem();
    p(target.toString());
    stop();
    Toolkit tk = Toolkit.getDefaultToolkit();
    units = tk.getImage("units.jpg");
    MediaTracker mt = new MediaTracker(bp);
    mt.addImage(units, 0);
    try {
        mt.waitForID(0);
    } catch (Exception e){
        System.out.println(e);
    }
    int center = bp.getSize().width / 2;
    int left = center - (units.getWidth(bp) / 2);
    bp.initLayer(1, 365 * 2, units.getHeight(bp));
    bp.shiftLayer(1, new DPoint( center, 400 ));
    oldgraphics = bp.getLayerGC(1);
    bp.hiddenInit(2, units.getWidth(bp), units.getHeight(bp));
    bp.shiftLayer(2, new DPoint( bp.getSize().width - (int)units.getWidth(bp) / 2, 400 ));
    newgraphics = bp.getLayerGC(2);
    newgraphics.drawImage( units, 0, 0, bp );
    for (int foo = 0; foo < ss.length; foo++){
        if (target.toString().compareTo(ss[foo].device) == 0)
            watch = foo;
        p("start watching: " + target.toString());
        start();
    }
}
public void run(){
    int delay;
    while(readthread != null){
        String raw = ss[watch].getRead();
        if (weight <= 0 && state == true) {
            state = false;
            bp.hideLayer(1);
            bp.hideLayer(2);
        } else {
            state = true;
            bp.showLayer(2);
        }
        newgraphics.drawImage( units, 0, 0, bp);
        newgraphics.setColor( Color.red );
        int adj_weight = (int)weight + CALIBRATION;
        newgraphics.fillRect( 0, graph(adj_weight), 3, adj_weight - 1 );
        //p("" + weight);
        delay = 10;
        if ( (int)parse(raw) == weight ) && ( weight > 0 ) {
            current++;
        } else {
            weight = (int)parse(raw);
            current = 0;
        }
        if (current > 60){
            Weight w = new Weight((double)weight, new Date().getTime());
            int x = 730;
            Calendar c = new GregorianCalendar();
            c.setTime( new Date() );
            int day = c.get( Calendar.DAY_OF_YEAR ) - 1;
            dataOut(w);
            while (x >= 0){
                oldgraphics.drawLine( x, graph( daily_weight[ day ] ), x - 2,
                graph( daily_weight[ dayBefore( day ) ] ));
                oldgraphics.setColor(Color.white);
                oldgraphics.drawLine( x, graph( 0 ), x - 2, graph( 0 ));
                x -= 2;
                day--;
                if (day < 0)
                    day = 364;
            }
            bp.showLayer(1);
            current = 0;
            delay = 15000;
        }
        try{
            Thread.sleep(delay);
        } catch (Exception e){
            p("" + e);
        }
    }
}
private int dayBefore(int day){
    if (day != 0)
        return day - 1;
    else
        return 364;
}
private void openFile(String filename){
Weight[] buffer = new Weight[10000];

Calendar c = new GregorianCalendar();

try {
    p("appending to " + filename);
    BufferedReader bri = new BufferedReader(new FileReader(filename));
    buffer = getContents(bri);
} catch (Exception e){
    p("new file");
} try{
    pw = new PrintWriter(new FileOutputStream(new File(filename)), true);
    int foo = 0;
    while (buffer[foo] != null){
        c.setTime(new Date(buffer[foo].time));
        daily_weight[c.get(Calendar.DAY_OF_YEAR) - 1] = (int)buffer[foo].weight;
        dataOut(buffer[foo]);
        foo++;
    }
} catch (Exception e) {
    p("" + e);
}

private Weight[] getContents(BufferedReader br){
    Weight[] buffer = new Weight[1000];
    String in = "start";
    int total = 0;
    try {
        while (in != null){
            in = br.readLine();
            if (in != null){
                long time = Long.valueOf(in.substring(0, 12)).longValue();
                double weight = Double.valueOf(in.substring(29, in.length() - 4)).doubleValue();
                buffer[total] = new Weight(weight, time);
                total++;
            }
        }
        br.close();
    } catch (Exception e){
        p("" + e);
    }
    return buffer;
}

private void dataOut(Weight in){
    if (pw.checkError() != true){
        pw.println(in.toString());
        p(in.toString());
    }
}

private int graph( double w ){
    return graph((int) w);
}

private int graph( int w ){
    return 300 - w;
}

private double parse(String in){
    double value = 0.0;
    if (in.length() > 0){
        try {
            value = (double)Integer.parseInt(in, 16);
        }
catch (Exception e) {
}
value = interpolate(value, new DPoint(6441, 0), new DPoint(65440, 220.2));
return value;
}

private double interpolate(double point, DPoint lo, DPoint hi) {
double m = (hi.y - lo.y) / (hi.x - lo.x);
return (m * (point - lo.x) + lo.y);
}

public void start() {
    readthread = new Thread(this, "read");
    readthread.start();
}

public void stop() {
    readthread = null;
}

private void p(String in) {
    System.out.println("NetWeightAgent: ' + in);
}

Class PolarHRMAgent
import java.awt.Frame;
import java.awt.Image;
import java.awt.Choice;
import java.awt.Label;
import java.awt.GridLayout;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.MediaTracker;
import java.awt.Toolkit;
public class PolarHRMAgent implements Runnable, ItemListener {
    Frame f;
    Choice c;
    BioPanel bp;
    SerialSource[] ss;
    Thread readthread;
    int watch;
    DPoint[] pul;

    public PolarHRMAgent (BioPanel bp, SerialSource[] ss) {
        this.ss = ss;
        this.bp = bp;
        c = new Choice();
        c.add("no device selected");
        for (int foo = 0; foo < ss.length; foo++)
            c.add(ss[foo].device);
        f = new Frame("PHRM Agent");
        f.setSize(200, 75);
        f.setLayout(new GridLayout(1, 1));
        f.add(c);
        f.show();
        c.addItemListener(this);
        pul = new DPoint[11];
        for (int foo = 0; foo < 5; foo++) {
            pul[foo] = new DPoint(.2 * foo, .2 * foo);
            pul[10 - foo] = new DPoint(.2 * foo, .2 * foo);
        }
    }
}
public void pulse()
{
    bp.scaleLayer(0, pul);
}

public void itemStateChanged(ItemEvent evt)
{
    Object target = (Object)evt.getItem();
    stop();
    Toolkit tk = Toolkit.getDefaultToolkit();
    Image heart = tk.getImage("heart.jpg");
    MediaTracker mt = new MediaTracker(bp);
    mt.addImage(heart, 0);
    try {
        mt.waitForID(0);
    } catch (Exception e){
        System.out.println(e);
    }
    bp.hiddenInit(0, heart.getWidth(bp), heart.getHeight(bp));
    bp.shiftLayer(0, new DPoint(400, 175));
    Graphics polargraphics = bp.getLayerGC(0);
    polargraphics.drawImage(heart, 0, 0, bp);
    for (int foo = 0; foo < ss.length; foo++){
        if (target.toString().compareTo(ss[foo].device) == 0){
            watch = foo;
            p("reading from: " + target.toString());
            start();
        }
    }
}

public void run()
{
    int delay = 20;
    while(readthread != null){
        if (ss[watch].getReadCount() > 0){
            pulse();
        }
        ss[watch].resetReadCounter();
        try{
            Thread.sleep(delay);
        } catch (Exception e){
            p("** + e");
        }
    }
}

public void start()
{
    readthread = new Thread(this, "read");
    readthread.start();
}

public void stop()
{
    readthread = null;
}

private void p(String in)
{
    System.out.println("PolarHRMAgent.java: " + in);
}
Class PortRequester

/ *---------------------------------------------------------------
 * File: PortRequester.java
 * Function: This interface is used for call backs from the port receive
task
 * when an object wishes to become the receiver for port data.
 *---------------------------------------------------------------*

import Serialio.*;

interface PortRequester {
    public void usePort(SerialPortLocal sp);
}

Class Queue

public class Queue extends java.util.Vector {
    public Queue() {}
    
    public boolean hasElementso() {
        if (super.size() < 1)
            return false;
        else
            return true;
    }
    
    public Object getLast() {
        int end = super.size() - 1;
        if (end >= 0) {
            Object o = elementAt(end);
            super.removeElementAt(end);
            return o;
        }
        else
            return null;
    }
}

Class ReadThread

import java.io.*;
import java.awt.*;
import Serialio.*;

// borrowed from JTermRcvTask, part of JavaTerm

class ReadThread extends Thread {
    int breakCnt = 0;
    FileOutputStream capFile;
    int rcvCnt = 0;
    boolean capture = false;
    boolean quiet = false;
    boolean echo = false;
    boolean showHex = false;
    boolean portRequested = false;
    SerialPortLocal p;
    SerialSource rcvBox;
    PortRequester caller;
public ReadThread (SerialPortLocal sp, SerialSource rcvBox) 
throws IOException
{
  if (sp.getPortNum() == -1) 
    throw new IOException("TermSndTask: serial port not initialized");
  p = sp;
  this.rcvBox = rcvBox;
  sp.setTimeoutRx(0);
  this.setName("ReadThread");
}

public void run() {
  int b, rdCnt;
  byte[] inBuf = new byte[128]; //115 bytes in 10ms at 115.2kbps
  rcvCnt = 0;
  boolean chkBreak = p.isSupported("sigBreak");
  for (;;) {
    try {
      if (chkBreak) {
        if (p.sigBreak()) {
          ++breakCnt;
          System.out.println("sigbreak");
          // this isn't important i don't think
          //rcvBox.writeChars("\nBreak "+breakCnt+");
        }
      }
      rdCnt = p.getData(inBuf);
      if (rdCnt > 0) {
        for (int i = 0; i < rdCnt; i++) {
          b = inBuf[i];
          ++rcvCnt;
          String s = GetDisplayable(b, rcvCnt);
          if (!quiet) rcvBox.addRead(s);
          if (echo) p.putByte((byte)b);
          if (capture) {
            try {
              capFile.write(b);
            } catch (IOException e) {
              System.out.println("capFile.write: "+e);
            }
          }
        }
        if (portRequested) {
          caller.usePort(p);
          portRequested = false;
        }
      }
    } catch (Exception e) {
      System.out.println("Error in TermRcvTask "+e);
      e.printStackTrace();
    }
  }
}

private char[] hexSym = {
  '0', '1', '2', '3', '4', '5', '6', '7',
  '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};

private String GetDisplayable(int b, int rcvCnt) {
  Character c = new Character((char)b);
  StringBuffer s = new StringBuffer();
  s.append(c.toString());
  return new String(s);
}

public void RequestPort(PortRequester caller) {
  this.caller = caller;
  portRequested = true;
}
public class SerialSource {
    final static boolean READ_LINES = true;
    final static boolean READ_CHARS = false;

    String device;
    String buffer;
    Frame f;
    TextField current;
    ReadThread rt;
    boolean lines;
    int reads;
    public SerialSource(String device, boolean lines){
        this.lines = lines;
        this.device = device;
        buffer = "";
        current = new TextField("");
        current.setEditable(true);
        f = new Frame(device);
        f.setLayout(new GridLayout(1, 1));
        f.add(current);
        f.setSize(200, 75);
        f.show();
        try {
            rt = new ReadThread(setUpSerial(device), this);
        } catch (Exception e) {
            p("can't get serial port");
        }
        p("start reading " + device);
        reads = 0;
        rt.start();
    }
    public String getRead(){
        return current.getText();
    }
    public int getReadCount(){
        return reads;
    }
    public void resetReadCounter(){
        reads = 0;
    }
    public void addRead(String in){
        if (READ_LINES)
            int i = (int)in.charAt(0);
            if ((i > 47 && i < 58) ||
                (i > 64 && i < 71) ||
                (i > 96 && i < 103)){
                buffer += in;
            }
            else{
                if (buffer.length() > 1){
                    current.setText(buffer);
                    reads++;
                }
                buffer = "";
            }
        }
    }
}
else{
    try {
        current.setText(in);
        reads++;
    } catch (Exception e) {
        p("" + e);
    }
}

private SerialPortLocal setUpSerial(String device) {
    // 9600, no handshake, no parity, 8bit, one stop
    SerialConfig serCfg;
    SerialPortLocal sp;
    try {
        serCfg = new SerialConfig(device);
        p("device: " + device);
        serCfg.setBitRate(SerialConfig.BR_9600);
        serCfg.setDataBits(SerialConfig.LN_8BITS);
        serCfg.setStopBits(SerialConfig.ST_1BITS);
        serCfg.setParity(SerialConfig.PYNONE);
        serCfg.setHandshake(SerialConfig.HSCTSRTS);
        sp = new SerialPortLocal(serCfg);
        sp.open();
        return sp;
    } catch (Exception e) {
        p(e + " at setUpSerial");
        System.exit(1);
    }
    return null;
}

private void p(String in) {
    System.out.println("SerialSource: " + in);
}

Class Weight
import java.util.Date;
public class Weight {
    double weight;
    long time;
    public Weight(double weight, long time) {
        this.weight = weight;
        this.time = time;
    }
    public Weight(double weight) {
        this.weight = weight;
        time = new Date().getTime();
    }
    public Weight() {
        weight = 0.0;
        time = new Date().getTime();
    }
    public String toString() {
        return (time + "(" + (new Date(time).toString().substring(4, 19)) + "") * weight + " lbs");
    }
}