Scott Downie's Interaction Design Portfolio

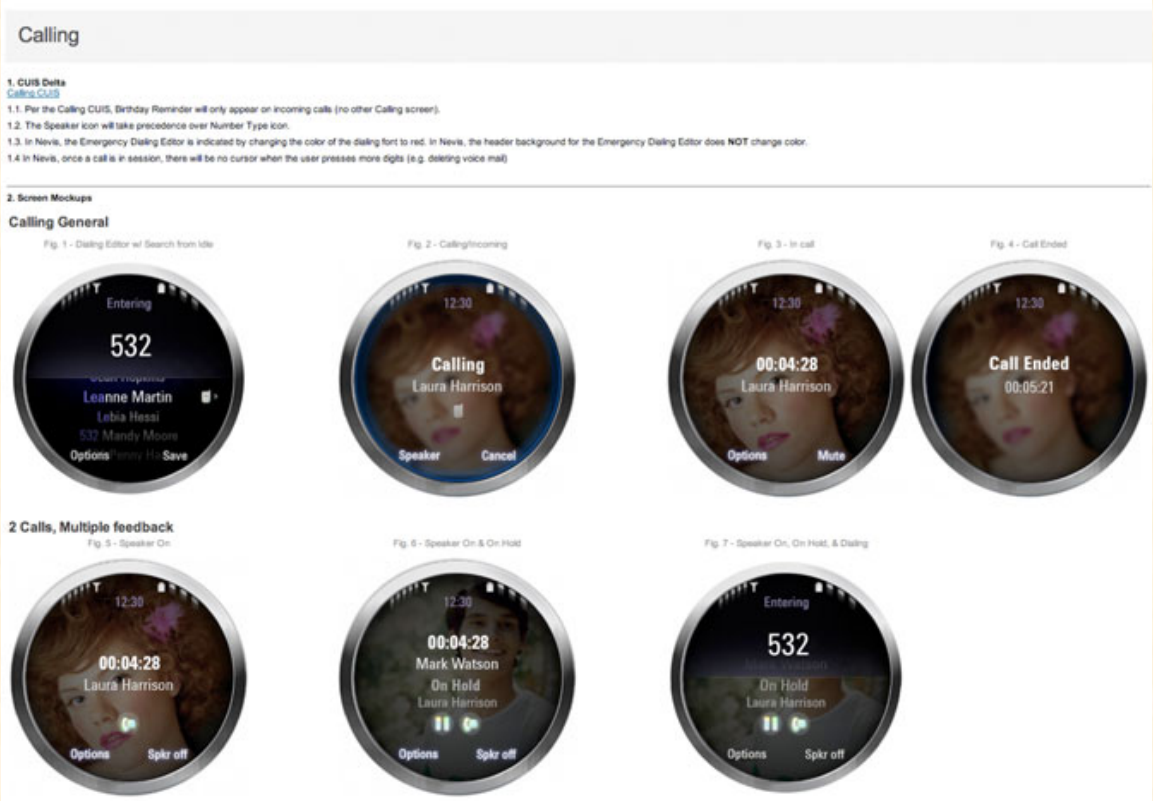
Motorola (2006 – 2009)

Collaborative Design/Wikis

I have advocated cross-disciplinary collaboration for a number of years. At Motorola, I was a successful wiki and CMS evangelist, getting managers, designers, and engineers up to speed on Compass (Motorola's company-wide CMS), Plone, TWiki, Subversion, and other valuable information-exchange tools.

For the AURA (a.k.a. "Nevis") project – Motorola's most expensive phone ever – I was the first in Motorola's Consumer Experience Design department to use web-based tools (wikis and CMS) to put all the design information in one place and allow any qualified team member to edit the documentation on the fly. As a result, the AURA went from concept to market in nine months instead of the standard two-year time frame.

This and the next two pages contain sections of the wiki specification page for the AURA "Calling" screen. AURA's lead visual designer (Lena Kim) created the photorealistic screen renderings and mockup animations. I created the schematic diagrams as well as a short QuickTime movie showing how the graphical layers fit together.



3. Transition Mockups and Specs

Fig. 17 - Calling transitions

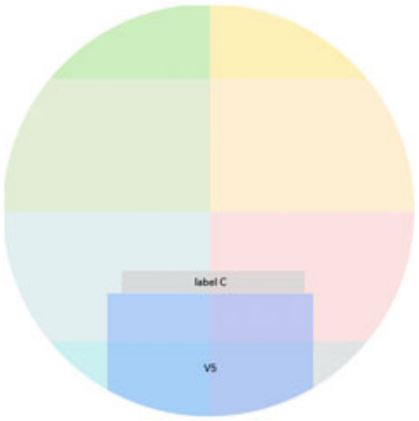
4. LRS Delta

Calling LRS for LJ 6.3.

The Qt LRS for Calling differs considerably from the TAT layout we are proposing for Nevis. Therefore please consider any Nevis padding and spacing names and values as unique to Nevis. In other words, a "V1" value in the Qt LJ 6.3 Calling app is not the same as a "V1" value in the TAT Nevis Calling app.

Related page: Search from idle

Fig. 17 - See figs 8 & 9 above - Configuration 1

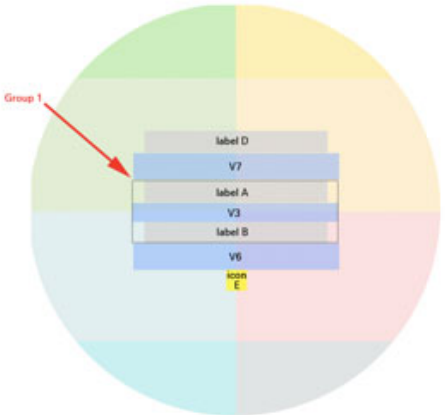


1. The meter images (e.g. the concentric volume circles) are centered in the screen (i.e. the intersection of the green, yellow, red, and blue regions).
2. Vertical spacer V5 separates the bottom of the screen from the bottom of label C.

Fig. 18 - See fig 5 and 6 above - Configuration 2

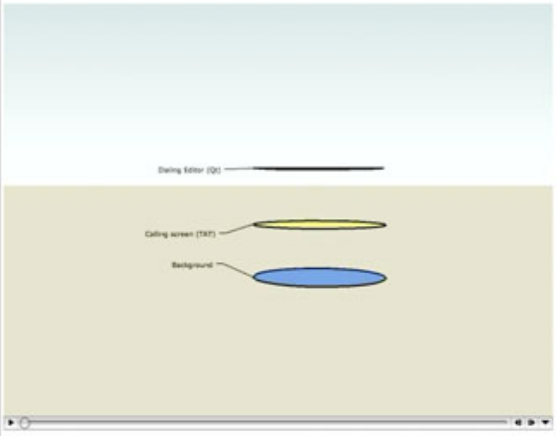
1. In this configuration, there are no center-aligned elements. All of the positioning is based on stacking elements from the bottom of the screen.
2. Vertical spacer V1 separates the bottom of the screen from the bottoms of icons A, B, & C.
3. The bottom of vertical spacer V2 is aligned flush to the tops of icons A, B, and/or C. If none of these icons are visible, the vertical space remains reserved.
4. Group 1 consists of the collection of label A, label B, and vertical spacer V3. Group 2 consists of the same collection of elements with the positions of labels A & B swapped.
5. Labels A and B are aligned flush to opposite sides of vertical spacer V3.
6. If a group is scaled in size, all of the elements within the group are scaled proportionately.
7. Groups 1 and 2 are aligned flush to opposite sides of vertical spacer V4. If Group 2 is not used, vertical spacer V4 is not used either.

Fig. 19 - See figs 2, 3, 4, 10, and 11 above - Configuration 3



1. In this configuration, the elements use center alignment as their starting point (unlike in Configurations 1 and 2).
2. Group 1 (labels A and B plus vertical spacer V3) is center aligned as a group.
3. If icon E is displayed, it is offset from the bottom of Group 1 by vertical spacer V6. If icon E is not used, vertical spacer V6 is not used either.
4. If label D is displayed, it is offset from the top of Group 1 by vertical spacer V7. If label D is not used, vertical spacer V7 is not used either.

Fig. 20 - Dialing Editor layers



Icon Animation

1. Calling in Nevis uses three animated icons: Speaker On (#1), Hold (#2), and Mute (#3).
2. These icons are located in the body area.
3. Each icon always occupies a definite place in line.
    1. When visible, the Speaker On icon is always the leftmost icon.
    2. When visible, the Mute icon is always the rightmost icon.
    3. If the Speaker On icon is not visible and the Hold icon is visible, the Hold icon is the leftmost icon.
    4. If the Mute icon is not visible and the Hold icon is visible, the Hold icon is the rightmost icon.
4. If only one icon is visible, it always appears in the horizontal center.
5. If a group of two or more icons is visible, the icon group is horizontally centered.

Fig. 21 - A Single Icon

See Fig. 5 (Speaker On) for an illustration.

Fig. 22 - Two Icons

See Figs. 6 and 7 for illustrations.

Fig. 23 - Three Icons

5. Placement Values

Standard LJ 6.3 Placement Values

State	Description	Property	Value	
			Skin 1	Skin 2
		Calling Dialog Header		
ALL	Padding	Top/Bottom	not applicable	not applicable
ALL	Padding	Left/Right	62 px	62 px
		Calling Dialog		
ALL	Padding	Left/Right	62 px	62 px
		Dialing Editor		
ALL	Padding	Top/Bottom	12 px	12 px
ALL	Padding	Left/Right	59 px	59 px
ALL	Background	calldialog_bg	Fixed	Fixed
		Emergency Dialing		
ALL	Emergency Background	calldialog_bg	Fixed	Fixed

Font Values

State	Description	Property ID	Font Size	Font Style	Alignment	Color		Outlines (px value/color)	
						Skin 1	Skin 2	Skin 1	Skin 2
ALL	Label A	call_status_std	44 pt	Bold	Center	# ffffff	# 000000	0	0
ALL	Label B	call_name_std	36 pt	Regular	Center	# ffffff	# 000000	0	0
ALL	Dialing Label: Small	dialog_small_std	44 pt	Regular	Center	# ffffff	# 000000	0	0
ALL	Dialing Label: Medium	dialog_medium_std	54 pt	Bold	Center	# ffffff	# 000000	0	0
ALl	Dialing Label: Large	dialog_large_std	70 pt	Bold	Center	# ffffff	# 000000	0	0

Nevis Specific Properties/Values

State	Description	Property	Value	
			Skin 1	Skin 2
ALL	Spacing	V1	105 px	105 px
ALL	Spacing	V2	7 px	7 px
AlL	Spacing	V3	8 px	8 px
ALL	Spacing b/w icon to icon	H1	10 px	10 px
ALL	2 calls: Spacing: label Group 1 to label Group 2	V4	13 px	13 px
ALL	Spacing	V5	149 px	149 px
ALL	Spacing	V6	16 px	16 px
ALL	Spacing	V7	22 px	22 px
ALL	Secondary Header (Group 2) Font size (%)	n/a	80 %	80 %
ALL	Blending Color/Opacity: 2 calls (b/w label A & label B)	blend_2calls	#000000/ 40%	#000000/ 10%
ALL	Image Blend: Caller id - REMOVED	overlay_caller_id	Fixed	Fixed
ALL	Blend: Caller ID - ADDED	blend_caller_id	#000000/ 75%	
ALL	Image: Birthday Reminder background	reminder_bg	Mid-Stretch	Mid-Stretch
ALL	Padding: Birthday Reminder background	Top	5	5
ALL	Padding: Birthday Reminder background	Bottom	4	4
ALL	Padding: Birthday Reminder background	Left	13	13
ALL	Padding: Birthday Reminder background	Right	13	13

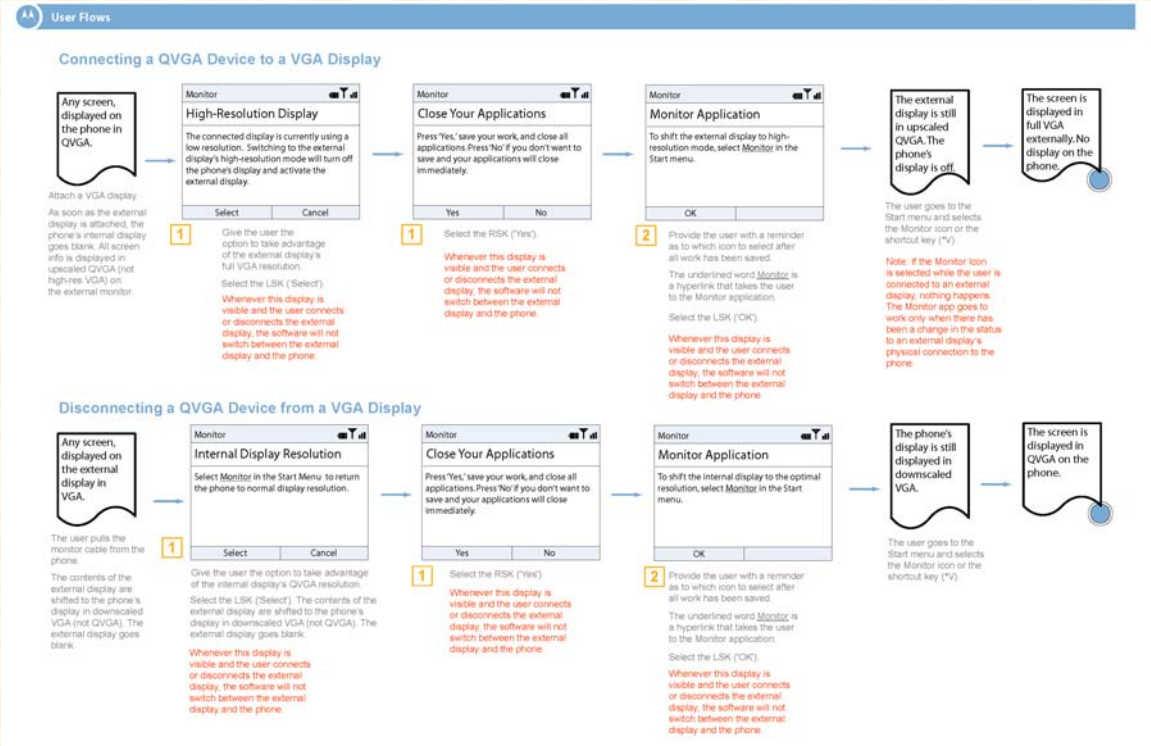
Nevis Specific Font Values

State	Description	Property ID	Font Size	Font Style	Alignment	Color		Outlines (px value/color)	
						Skin 1	Skin 2	Skin 1	Skin 2
ALL	Birthday Reminder (label D)	call_bday_reminder	24 pt	Regular	Center	# 7e6dbb	# cea000	0	0
				Emergency Dialing					
ALL	Dialing Label: Large	emergdialog_large_std	70 pt	Bold	Center	# f50101	# d10000	0	0

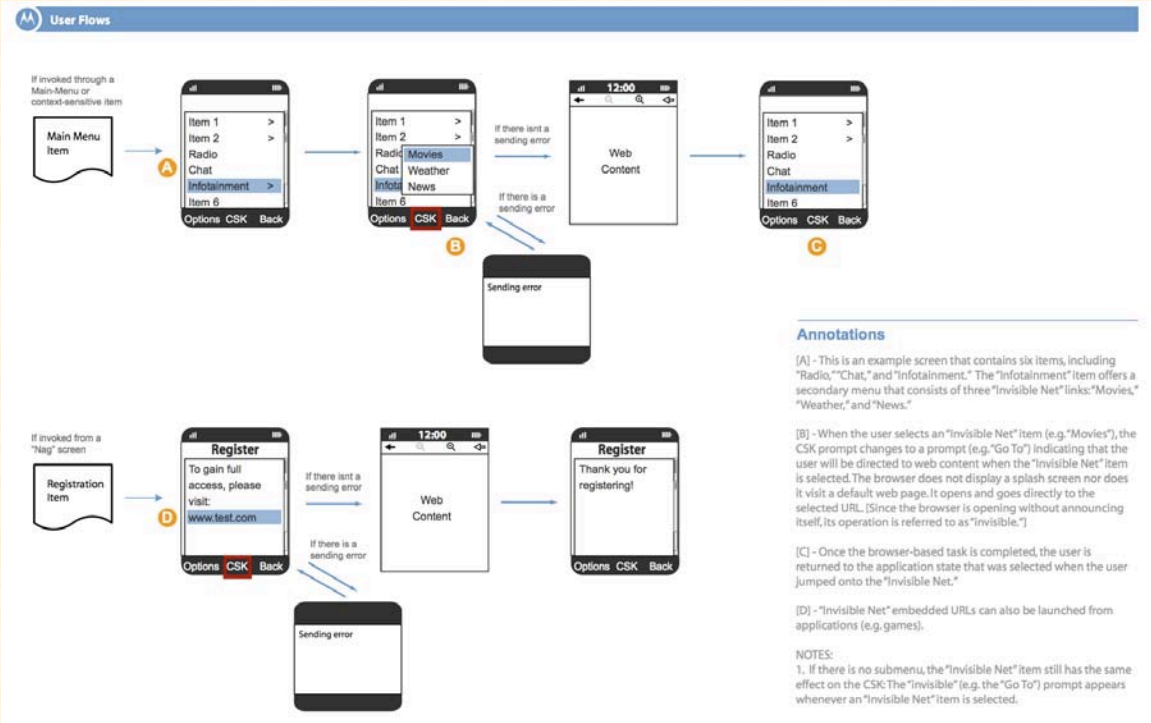
see Effects and Definition page for Blur Effect Values.

Interaction Flows

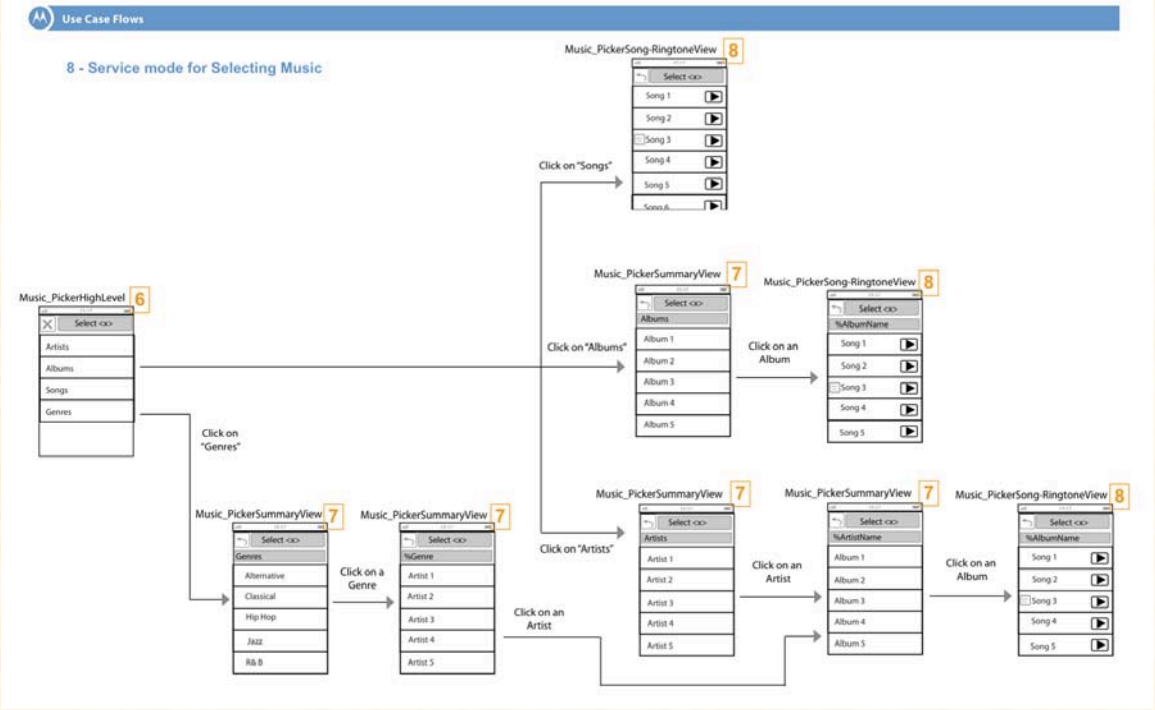
I contributed interaction designs to the Motorola team – developing flows, wireframes, mockups, and simulations – for a variety of mobile applications on a variety of software platforms.



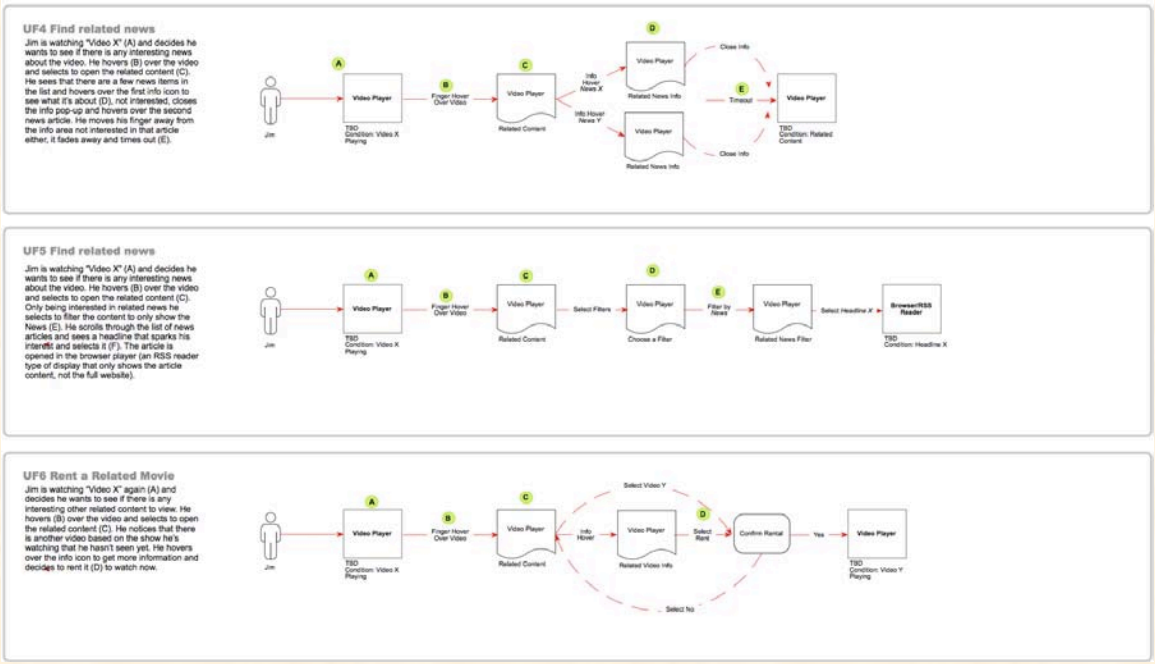
A page from a short Windows Mobile feature flow



A page from a short UIQ feature flow



A page from a music player flow



A page from a set of multimedia viewer maps

Screen Layouts/Architectures

I was the Consumer Experience Design department's Layout Requirements Architect from August 2006 to February 2008. In that capacity, I created "blueprints" and coordinated the Chicago headquarters' graphic and interaction design work with software development teams located in China, India, Russia, Brazil, France, the UK, and the US.

Chinese Text-Input Widget

Description

The Chinese Text-Input Widget concurrently displays all possible Chinese, Numeric, and Roman disambiguations for a given virtual key-press sequence. The Chinese Text-Input Widget is separated into two sections: the Disambiguation Bar and the Conversion Bar.

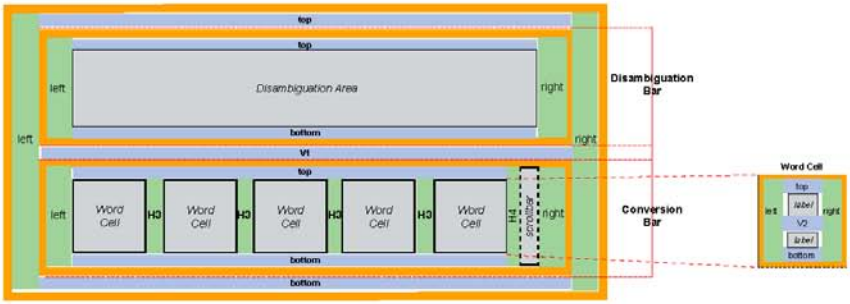
- The Disambiguation Bar, containing the "Mixed-Mode UI" (MMUI), provides three alternative input methods: Chinese (PinYin, Stroke, or ZhuYin), Numeric, and Roman.
- The Conversion Bar presents the user with the results in Chinese or Roman characters, depending upon which category in the Disambiguation Bar owns the current active or inactive focus highlight.

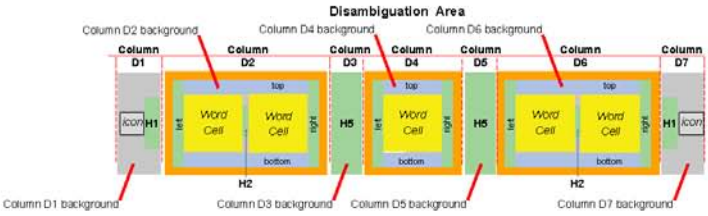
The Chinese Text-Input Widget is only shown at the request of the user for text entry, and must be positioned dynamically so that it will never obscure the word being built at the cursor.

Related Widgets

- None

Blueprint

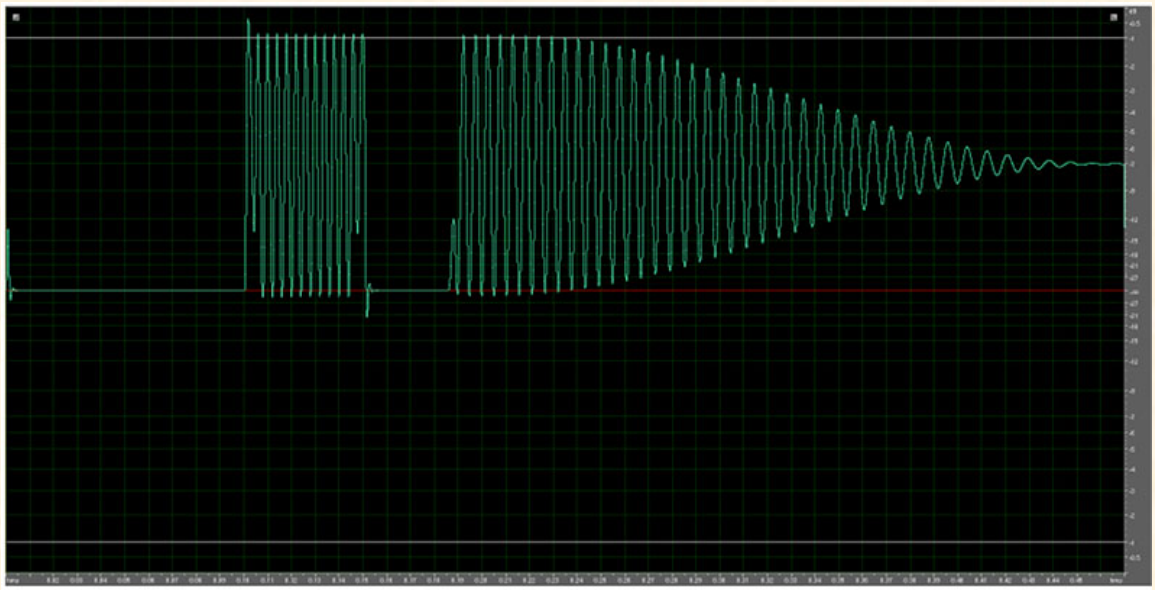




The first page from a Linux-Java platform layout requirement

Haptics

I researched and designed haptic (touch) experiences to use on mobile devices of the future. This involved helping design the hardware, developing drivers for the software, and determining what sorts of touch “melodies” our users would find most informative and pleasing.



Screenshot of a low-frequency waveform fed to a low-frequency haptic transducer (vibrator) in the mobile device.

A “gesture matrix,” one attempt to organize our haptic experiences.

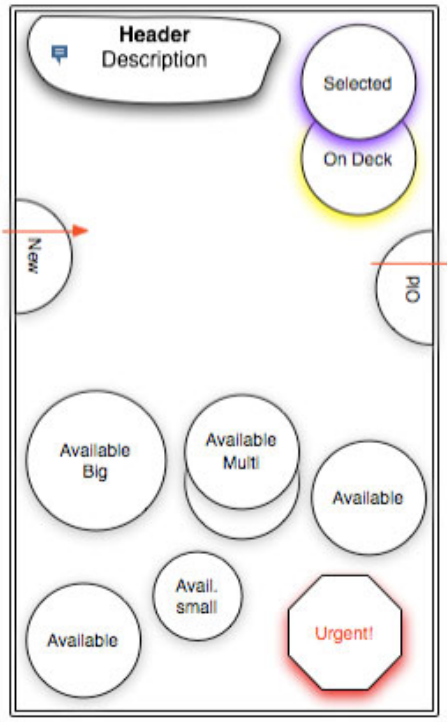
Design Brainstorming

A considerable amount of energy was devoted to working as part of the team responsible for coming up with the next generation of mobile social connectivity. Our group's primary emphasis was on multimedia sharing and presentation … from scratch. This was exciting and lots of fun because we were asked to throw ideas into our big brainstorm pot every day.

The next two pages illustrate my contribution to one afternoon's brainstorm session. (We typically held at least one session per day.) In this case, I decided to steal the notion of "monads" from Gottfried Wilhelm Leibnitz (1646-1716) and turn them into a UI philosophy of which Leibnitz may or may not have approved.

The third page contains one of the flows that emerged from our brainstorming sessions.

Monad Semi-Specifics



One of many possible page architectures could be as follows.

The shape-shifting header is in the top-left.

Selected media and their on-deck relations are in the top-right.

The currently chosen set of monads accumulate in the bottom of the "marble bag."

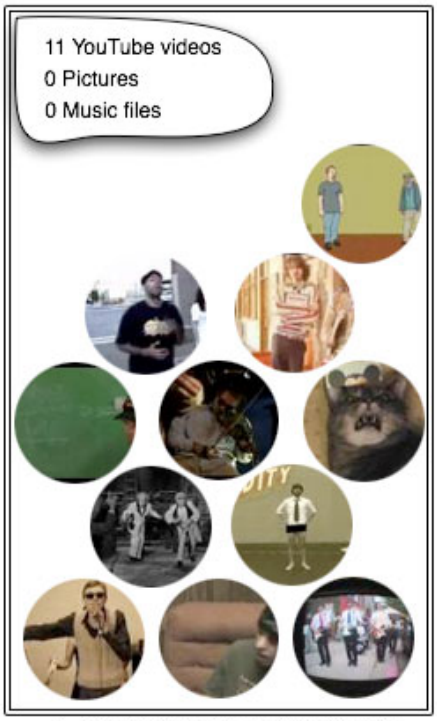
New monads enter on the left side, just below the header. Old monads shuffle off on the right side, just below the selected media area. (Think of a bathroom sink with its emergency drain.)

OR

Clusters of monads may be flicked on or off the screen to the left, right, top, or bottom, only to be replaced by another wave of monads.

Since the whole layout is fluid, the user could, for example, choose larger text sizes. As the text grew, some monads may grow in size and some may be pushed off the screen.

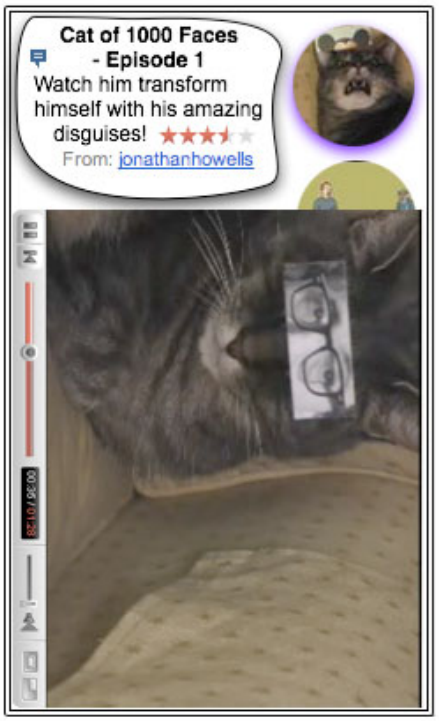
Although the header area is consistently in the top-left corner in this exploration, the header could be as free to roam as any monad. As long as we can consistently associate a specific meaning with a specific behavior, anything goes.



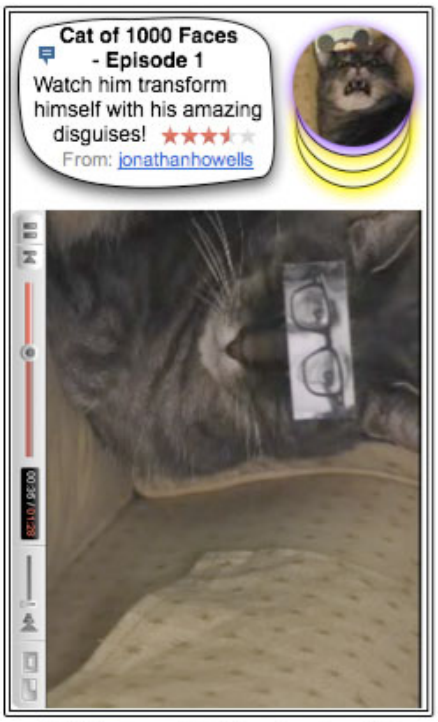
1. Jim starts with the default YouTube Featured Videos selection. The ordering comes from YouTube. These video "monads" slowly float around the screen. A monad's relatively vigorous wiggle could mean something (e.g. download in progress).

2. Jim taps a monad that looks like it has a cat in it. The selected monad -- indicated by a purple shadow -- starts to float to the selected media region, with other monads getting out of the way and generally wiggling around.

3. Like a famished amoeba, the header/ description area engulfs the chosen media monad as the information arrives from the server. Other monads continue to wiggle a bit. Any monads that might have some sort of relation to the chosen video could wiggle toward the upper-right corner.

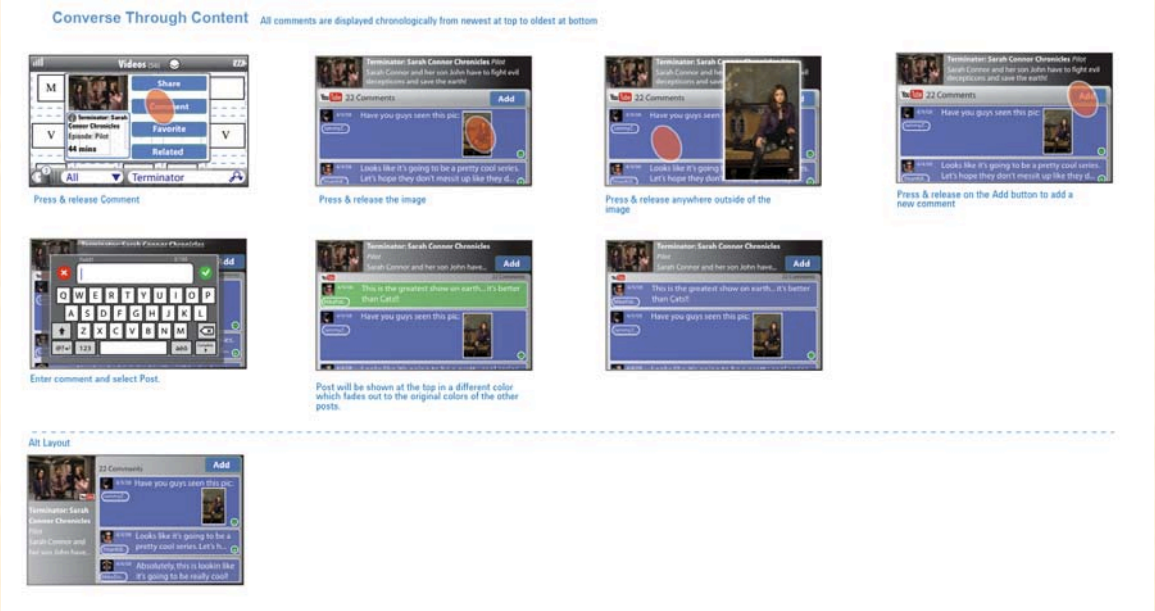


4. Tapping the media monad starts the YouTube player. While the player is getting ready to roll (or is paused), another monad peeks up in the upper-right corner, inviting the user to click it if Cat of 1000 Faces falls flat. When the video is playing, however, Jim should not be distracted by monad movement.



5. If Jim tapped the "jonathanhowells" link, other video monads from Mr. Howells might line up behind the currently selected monad. Jim could cycle through this set of on-deck videos by tapping ... somewhere.

6. As Jim clicks items that require screen real estate, the new material appears in layers on top of the previous layers. The video may or may not pause automatically when a new layer is requested.



Our group came up with a series of flows like this after we'd refined our raw brainstorming material.

Products

Here are three products I worked on in the between August 2006 and November 2008: the AURA (Motorola's most expensive phone ever), the MOTO U9 and the ROKR E8. All run Motorola's Linux-Java operating system.

AURA

MOTO U9

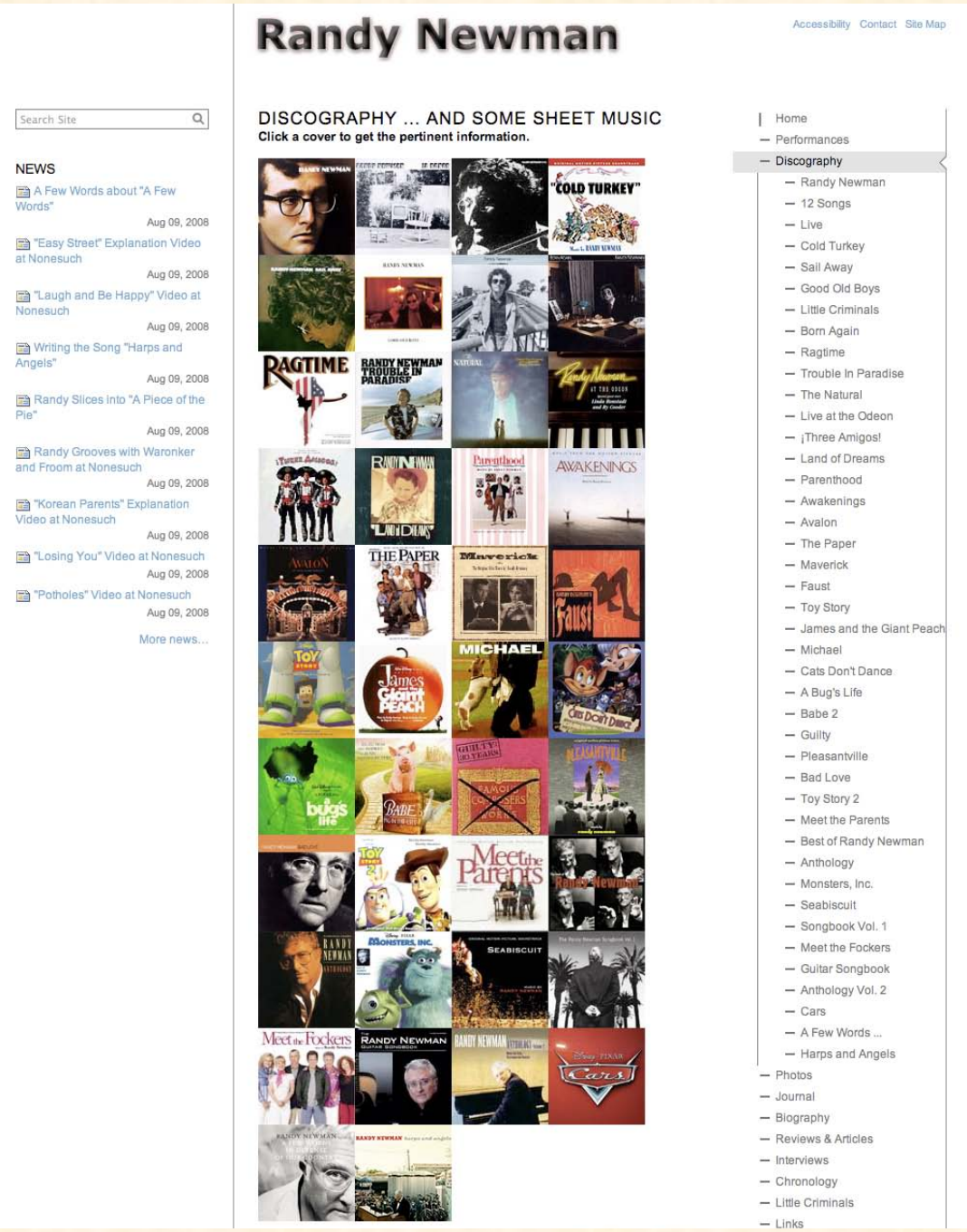
ROKR E8

Freelance Work (1998 – 2009)

I've designed and engineered websites, written articles on digital multimedia and authored online and printed user documentation.

Web Sites

I designed and constructed the web site for Oscar- and Grammy-winning musical artist Randy Newman.



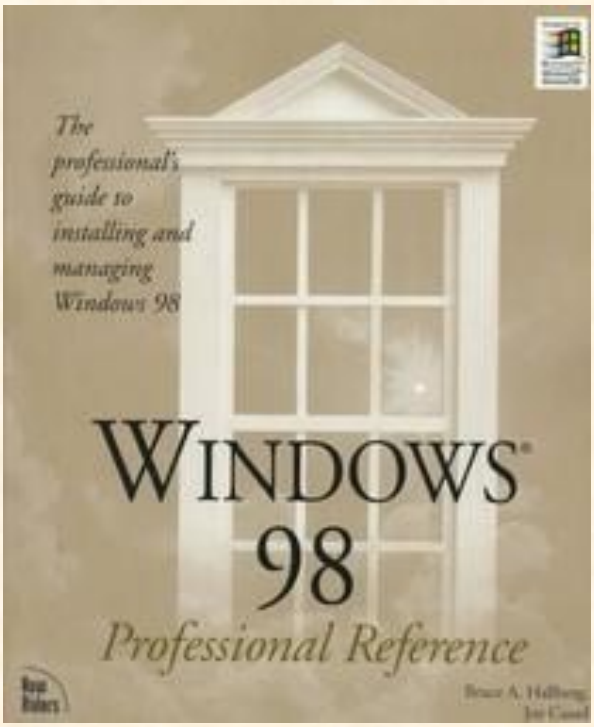
I have also contributed work to academic and non-profit organizations.

The Willow Pond Films front page

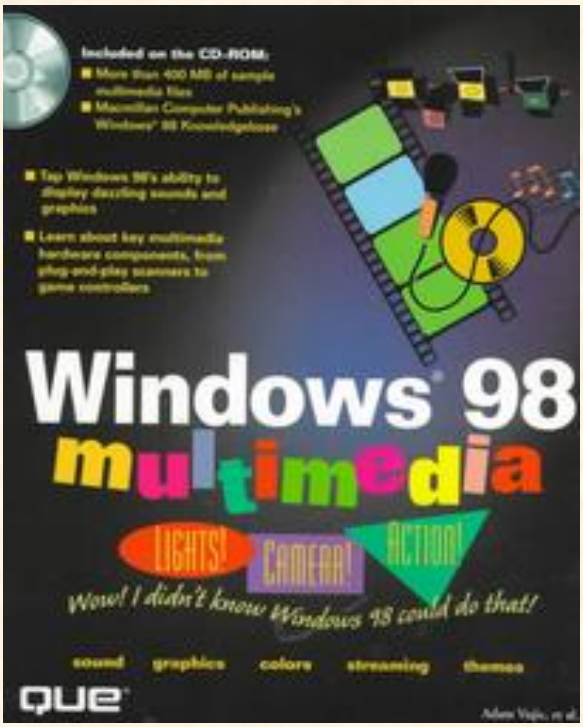
The front page for a 2006 academic conference

Writing

I wrote the "Multimedia" chapter (#14) in New Riders' *Windows 98 Professional Reference*.

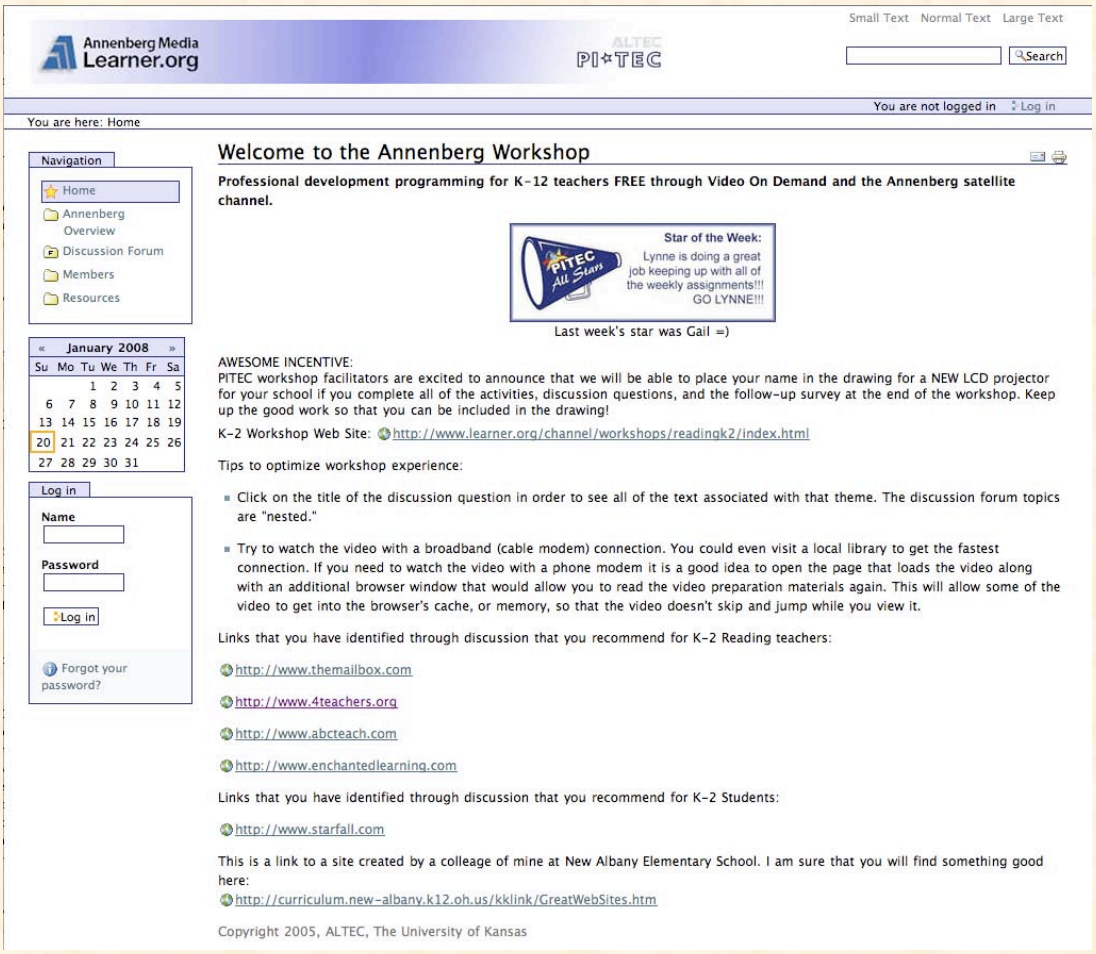


I also wrote the "Working with MIDI" chapter in Que's *Windows 98 Multimedia*.



ALTEC (2005)

ALTEC is a grant-funded extension of the University of Kansas' Education Department. I worked there part time while I was finishing my Master's in Interaction Design. My main duty was to quickly create web portals that educators could use for a variety of purposes. My platform of choice was Zope/Plone, although I also worked on .Net-based sites too.



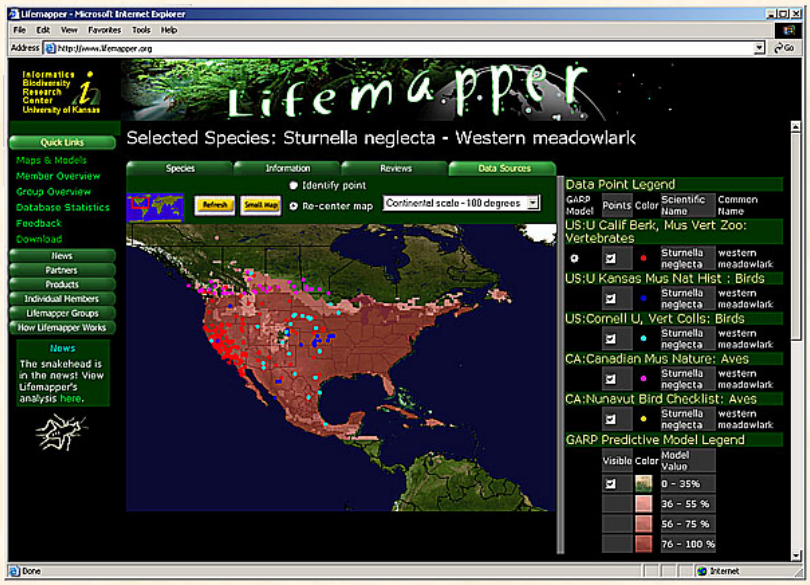
Screenshot of a typical Altec Plone site

Biodiversity Research Center (2002 – 2004)

At the University of Kansas' Biodiversity Research Center (BRC), I was hired as an Application Interface Developer, but I served in a number of other capacities as well: Documentation supervisor, software engineer, and even second-string network guru.

The BRC's two main applications were Specify – a museum-collections database – and Lifemapper – an ambitious attempt to map the world's species to their native environments.

The Specify software site's home page

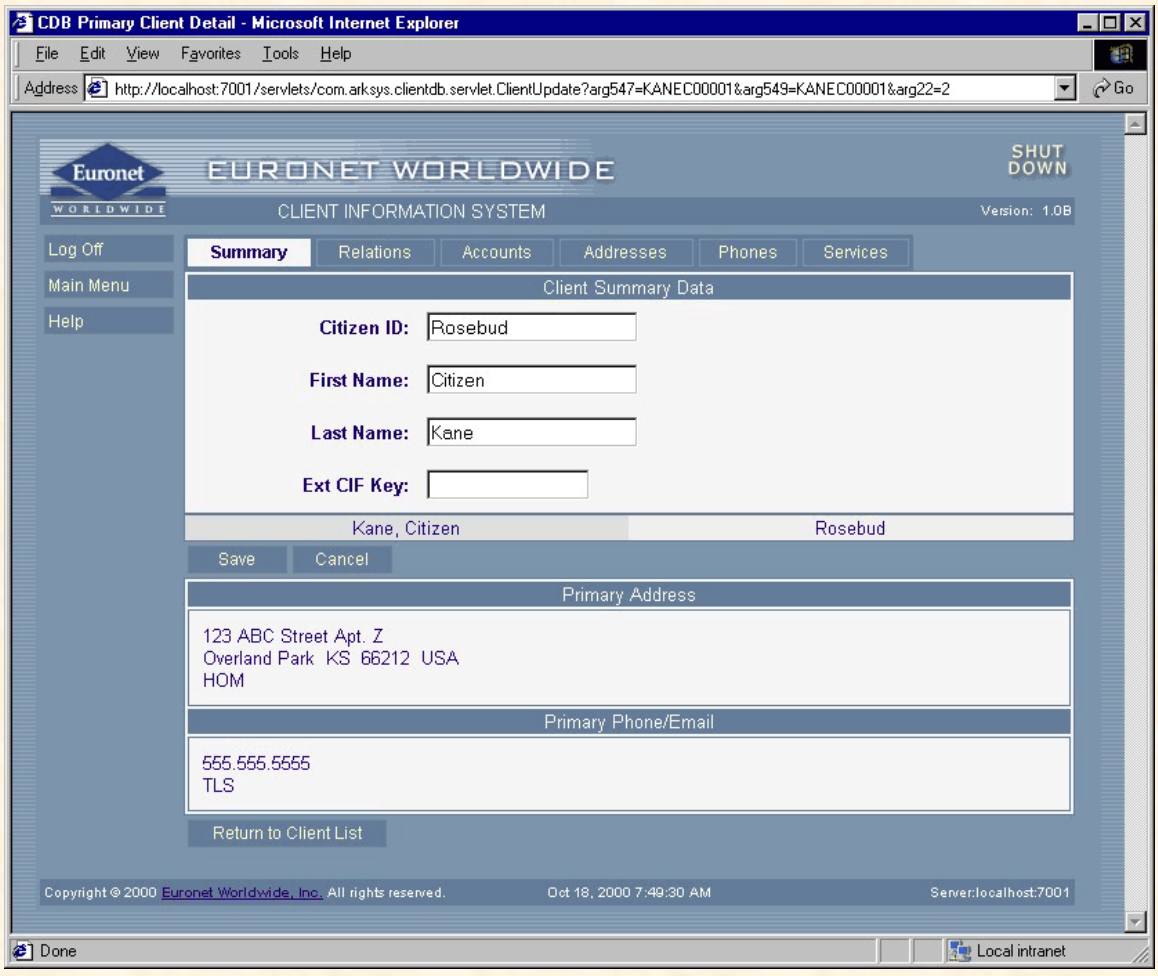


The Lifemapper home page, circa 2004

Euronet (2000)

Euronet was another place in which I did more than just the user interface work. At various times I also managed the Documentation and Quality Assurance departments.

Our charter was to create web-based financial systems that would run on Windows NT. This dictated a very conservative approach to user interface design. The example below is one screen from a suite of administration pages that were to be used by a bank executive.



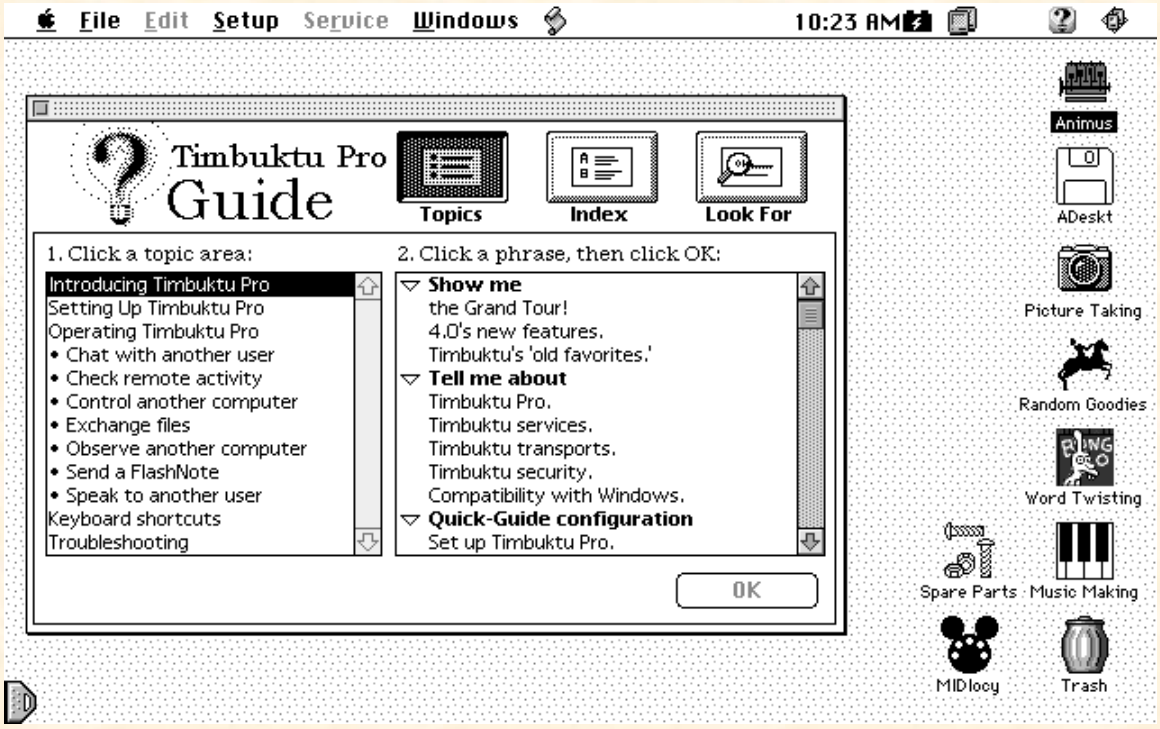
One of the pages from Euronet's Client Administration module

Netopia (1996 – 1999)

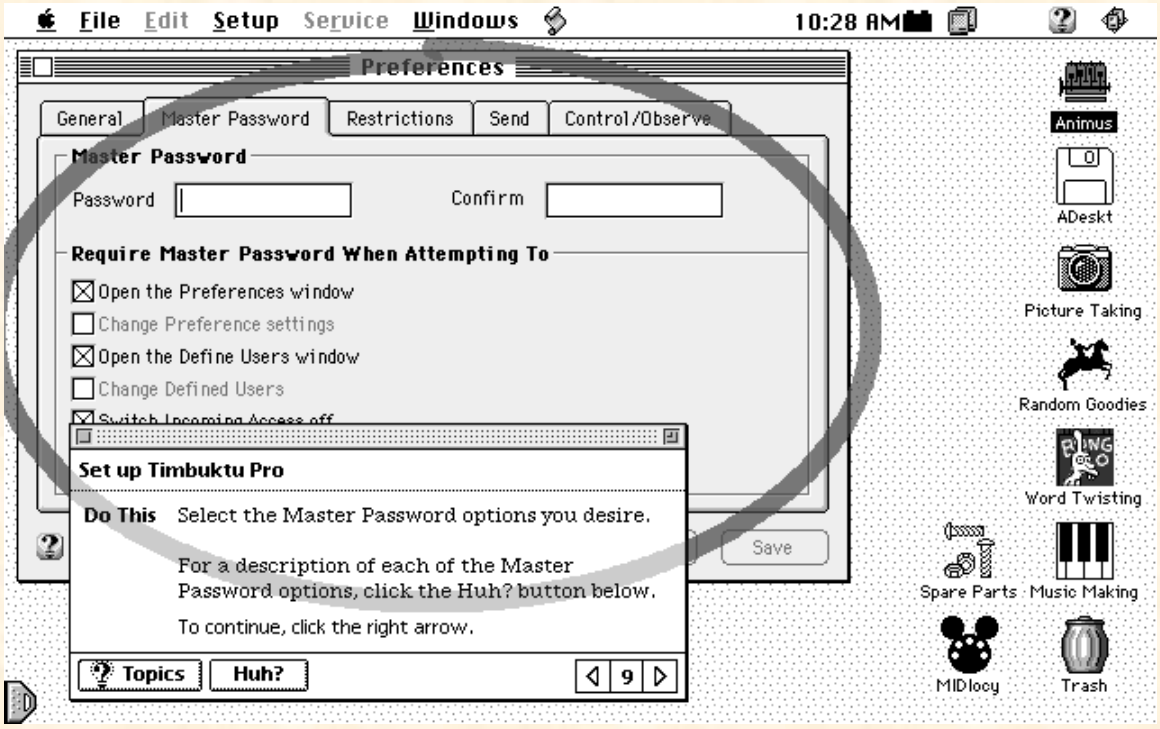
I began working with modern software UI at Netopia (which is now part of Motorola), primarily through the online help systems I wrote.

The help system for which I won a number of Society for Technical Communication awards was the Apple Guide help for Timbuktu Pro. This was more than just a contextually aware help engine. Apple Guide could actually draw marks on the application itself, eliminating the need for screen shots. It also offered up only the chunked pieces of textual help that were immediately relevant to solving the problem at hand. The system was very much ahead of its time. Unfortunately, it was also difficult to master, leading most Apple developers to simply ignore it. Therefore Apple Guide did not survive the Apple's transition to OS X.

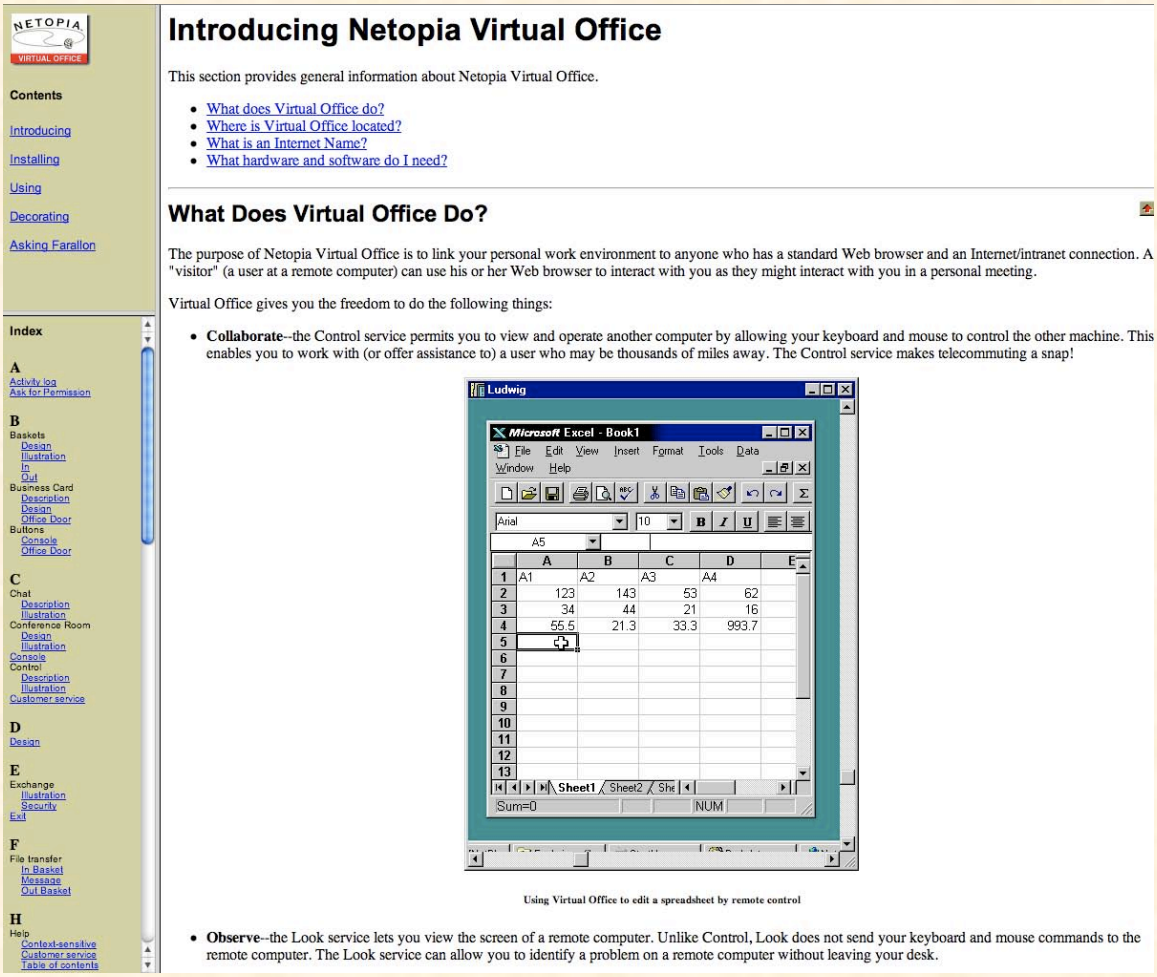
The screenshots below are from a black-and-white, 1993-vintage PowerBook 160. The first image depicts Timbuktu Pro's main Guide page.



This is a screen shot of Apple Guide telling the Timbuktu Pro application to circle the Master Passwords options in the Preferences panel. If the user had not opened the Preferences panel beforehand, the Apple Guide would have opened it for him! I planned out the Apple Guide flow, coordinated work with the software developers to make sure that the proper hooks were in place, and wrote all of the Apple Guide code and content.



I also created web-based help system when the Web was in its infancy (circa 1995).



The home help page for Netopia's Virtual Office from 1996

PianoDisc (1992 – 1993)

Before the World Wide Web, there were player pianos. Or so it was in my case.

As PianoDisc's technical director and senior music editor, I was in charge of creating the PDS-128, a player-piano control system that was in production from 1993 to 2006 – an eternity in the consumer electronics space. I was given the freedom to create any hardware and software layout I wanted … as long as my designs didn't cost any money. (Hence the control buttons aligned in a single row: It was cheaper to have them soldered directly to the printed-circuit board.)

Although I had designed software interfaces before (e.g. the 1988 with the Commbase academic database), this was the first interface – albeit a crude one by contemporary standards – to be marketed to an upscale clientele. It sold well and, judging by the lifespan (13 years) and legacy of the PDS-128, there were (and still are) many happy customers!