

UNIVERSITY OF CALGARY

A Visual Programming Language for Live Video Sonification

by

Roberto Arturo Diaz-Marino

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

MARCH, 2008

© Roberto Arturo Diaz-Marino 2008

THE UNIVERSITY OF CALGARY
FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled “A Visual Programming Language for Live Video Sonification” submitted by Roberto Arturo Diaz-Marino in partial fulfillment of the requirements for the degree Master of Science.

Supervisor, Saul Greenberg
Department of Computer Science

Rob Kremer
Department of Computer Science

Ehud Sharlin
Department of Computer Science

External Examiner, David Eagle
Faculty of Music

Date

Abstract

The problem explored in this thesis is how to allow a person to map features from multiple video streams, or parts of those video streams, into an audio space, and how they can then monitor the resulting soundscape. I consider the requirements, user interface and architecture of a system that lets an end-user configure a video-only stream so that it generates an audio soundscape. In particular, I present a programming by demonstration environment called Cambience. Using this system, end users can map visual characteristics collected from video regions onto audio samples, and assemble the audio stream in a way that creates a meaningful soundscape. Cambience is used to mitigate a simple scenario between two distance separated collaborators and a shared resource. The wide range of visual programming elements allow for complex sonification behavior. Informal evaluations of the system show that Cambience has real usefulness with mostly minor considerations.

Publications

Diaz-Marino, R. and Greenberg, S. (2006) Cambience: A Video-Driven Sonic Ecology for Media Spaces. Video Proceedings of ACM CSCW'06 Conference on Computer Supported Cooperative Work, November, ACM Press. Video and two-page summary. Duration 3:52

Diaz-Marino, R. and Greenberg, S. (2006) Demonstrating How to Construct a Sonic Ecology for Media Spaces through Cambience. Demo, Adjunct Proc ACM CSCW 2006.

Acknowledgements

First, I would like to give a huge thank-you to my supervisor Saul Greenberg who noticed my talents and encouraged me to pursue this Master's degree. Without his continued encouragement I might not have finished it with all the influences in my life that were pulling me away. Even though our areas of interest were very different and at first we had a tough time finding something that satisfied us both, I am proud of the very unique project we went ahead with. Furthermore, I am proud to have seen this project through until the end, as I am someone who typically has too many ideas to stick with one thing for very long.

I would also like to thank my fellow ILab members for their help and interest in this project. My regret is that I couldn't get more involved with group activities and get to know everyone even better. Nevertheless, it didn't stop anyone from welcoming me in, on the rare occasion I was able to pull myself away from my other responsibilities.

I would also like to thank:

- Leila Sujir for facilitating my session at the Banff centre for my dance case study.
- Jim Jewitt of the former QUAB Gallery for his patience, cooperation, and helpful advice in the art gallery case study.
- Alan Dunning for giving me a crash course in Isadora and Eyesweb.

Dedication

I dedicate this thesis to my dad, who probably had no idea what he was starting when he gave me my first BASIC programming book. When I got the wild idea to create my own video game, he taught me from the engineering calculation programs he created on our Tandy 1000, and within a month he couldn't help me anymore because I had gone beyond what he knew. Still he found ways to nab programming reference manuals from work to keep me going. Until I got my own computer we would often fight for time on his. When I had my own computer, he would fight for my time off it! Nevertheless, he put me on my direction in life and I felt very fortunate to have that kind of certainty as I went on to University.

Table of Contents

Approval Page.....	ii
Abstract.....	iii
Publications.....	iv
Acknowledgements.....	v
Dedication	vi
Table of Contents	vii
List of Tables	xi
List of Figures	xiii
Chapter 1. Introduction	1
1.1 Motivations.....	2
1.1.1 Video and Privacy.....	2
1.1.2 Video as a Foreground Display.....	3
1.1.3 Video as an Interactive Installation.....	5
1.2 Goals.....	6
1.3 Organizational Outline	6
Chapter 2. Background and Related Work	8
2.1 Media Spaces.....	8
2.1.1 Casual Interaction and Informal Awareness	9
2.1.2 What is a Media Space?.....	10
2.1.3 Sound in Media Spaces.....	12
2.1.4 Abstract Media Spaces.....	14
2.1.5 Abstract Mappings	16
2.2 Sound.....	17
2.2.1 Acoustics.....	17
2.2.2 Psychoacoustics	21

2.2.3	Changing Sound.....	22
2.2.4	Sound as an interface	23
2.3	Visual Programming	24
2.3.1	VPLs for Mapping Video to Audio	26
2.4	Summary	28
Chapter 3.	Introducing Cambience through Scenarios of Use.....	29
3.1	Scenario 1	30
3.1.1	Step 1 – Physical Setup.....	30
3.1.2	Step 2 – Configuring Regions of Interest	31
3.1.3	Step 3 – Monitoring Activities in Regions	34
3.1.4	Step 4 – Mapping Activities to Sounds.....	36
3.1.5	Step 5 – Sharing Mappings with Others	41
3.2	Scenario 2.....	41
3.2.1	Step 1 – Physical Setup.....	42
3.2.2	Step 2 – Connecting to the Server.....	43
3.2.3	Step 3 – Loading an Ecology Share	43
3.3	Scenario 3.....	44
3.3.1	Step 1 – Configuring Regions in Private Space.....	46
3.3.2	Step 2 – Monitoring and Mapping Private Activities	47
3.4	Scenario 4.....	50
3.4.1	Configuring Regions in Public Space	50
3.4.2	Step 2 – Monitoring and Mapping Public Activities	50
3.5	Scenario Summary	51
3.6	Conclusion.....	52
Chapter 4.	Shared Media and Live Visual Programming	54
4.1	Sharing Media	54
4.1.1	Shared Data.....	55
4.1.2	Cambience as a Distributed System.....	56
4.2	Visual Programming	59
4.2.1	Items, Attributes, and Functions	62

4.2.2	Sources, Sinks, and Patches	63
4.2.3	Signals.....	64
4.2.4	Live Programming	67
4.3	Summary	69
Chapter 5.	Mapping Video to Audio.....	70
5.1	Visual Items.....	70
5.1.1	Analyzing Regions.....	70
5.1.2	Change Detection through Frame Differencing.....	71
5.1.3	Other Change Metrics	74
5.1.4	Ambiguity	75
5.1.5	Signaling Change with Region Items	76
5.1.6	Camera Items	77
5.2	Sound Items.....	77
5.2.1	Sonic Ecologies.....	78
5.2.2	Selecting Sounds.....	80
5.2.3	The Wave3D Item.....	82
5.2.4	The Wave Item.....	83
5.2.5	The Audio Item	84
5.2.6	The Playlist Item	84
5.2.7	MIDI Instrument and MIDI Track Items	85
5.2.8	The Cambience SFX Library	86
5.2.9	Audio Configuration	87
5.3	Signal Items.....	87
5.3.1	Monitoring and Testing.....	89
5.3.2	Signal Transformation	90
5.3.3	Logical Operations.....	95
5.3.4	Mathematical Operations	97
5.3.5	History and Memory	99
5.3.6	Time and Date.....	102
5.3.7	Composite Items	102

5.4	Extensibility	104
5.5	Summary	107
Chapter 6. Applications and Case Studies		109
6.1	Case Studies	109
6.1.1	Workgroup Space.....	110
6.1.2	Dance Performance	113
6.1.3	Art Gallery	117
6.2	Applications to Prior Research.....	120
6.2.1	Very Nervous System	120
6.2.2	Iamascope	121
6.3	Summary	122
Chapter 7. Conclusion and Future Work		123
7.1	Contributions.....	123
7.2	Future Work	124
7.2.1	Revision and Evaluation of Cambience.....	124
7.2.2	Sound as an Interface	125
7.2.3	Social Protocols of Public Audio.....	126
7.3	Possible Future Applications.....	126
7.3.1	Store Security.....	126
7.3.2	Interactive Music in Malls	127
7.3.3	Interactive Museum Exhibits	127
7.3.4	Aging in Place.....	128
7.3.5	Home Security and Notification	128
7.3.6	Abstract Art Piece	129
7.4	Conclusion.....	129

List of Tables

Table 2.1: Summary table of desired Visual Programming Language features.	27
Table 4.1: Key-value structures in Cambience’s shared data.	59
Table 4.2: Signal Components.....	65
Table 5.1: Contributions of Cambience’s standard modules.....	106

List of Figures

Figure 1.1: The Community bar from McEwan and Greenberg (2005).	5
Figure 2.1: Xerox PARC media space. Image taken from Bly et al, 1993.	11
Figure 2.2: A screenshot of Portholes, taken from www.billbuxton.com	12
Figure 2.3: Blur (top) and pixelize (bottom) filters, taken from Boyle, 2005.	15
Figure 2.4: Combining two waveforms and corresponding spectral plots.	19
Figure 2.5: An example of an ADSR amplitude envelope.	20
Figure 2.6: A time varying spectral plot.	20
Figure 3.1: Floor layout for Kate and Josh. Cameras are marked as red dots.	30
Figure 3.2: Three types of activity that Kate and Josh wish to monitor.	32
Figure 3.3: The Cambience main window.	32
Figure 3.4: (a) Creating the front region, (b) Naming the front region.	33
Figure 3.5:(a) Naming the Region, (b) Camera Preview.	34
Figure 3.6: Creating a Region Item. A person enters the front region.	35
Figure 3.7: Adding and activating (a) displaywall, and (b) behind regions.	36
Figure 3.8: Audio Browser	38
Figure 3.9: (a) A newly created Wave3D Item, (b) Adjusting the item.	39
Figure 3.10: (a) The Toolbox panel, (b) Connecting the Threshold Item, (c) Adjusting the Threshold Item.	39
Figure 3.11: Mappings from the front (a) and displayWall (b) regions.	40
Figure 3.12: Sharing a workspace configuration (Ecology)	42
Figure 3.13: Connecting to the Cambience Server	44
Figure 3.14: Josh sees regions from the server when he connects.	44
Figure 3.15: The User Tab can be used to view and load Ecology Shares.	45
Figure 3.16: The <i>displaywall</i> share is loaded in a new "untitled" tab.	45
Figure 3.17: Kate disables camera frames from being transmitted.	47

Figure 3.18: Kate's region covers her entire webcam view.	48
Figure 3.19: Creating a new ecology workspace.	49
Figure 3.20: Two mapping schemes run in parallel.	49
Figure 3.21: Josh's webcam view partitioned into areas of possible interest.	50
Figure 3.22: A sonic ecology to monitor Josh.	51
Figure 4.1: Cambience Main Program Window	61
Figure 4.2: Examples of Items, attributes, and functions	63
Figure 4.3: Examples of sources, sinks, and connections.	64
Figure 4.4: Attributes and Signal Components.	66
Figure 4.5: Signal visualizations - level and activation.	67
Figure 4.6: Infinite signal loop (left) broken on arrival of next signal (right).	68
Figure 4.7: The Local Camera View window can be expanded to any size.	69
Figure 5.1: Cambience Input Flowchart	72
Figure 5.2: Cambience Webcam Capture Settings	74
Figure 5.3: Region Item	76
Figure 5.4: The Camera Item (middle) and corresponding video window (right).	77
Figure 5.5: Cambience Output Flowchart.	79
Figure 5.6: The Audio Browser (a) and the three sound items (b-d).	81
Figure 5.7: Common sound item components, seen on an Audio Item.	82
Figure 5.8: Synchronizing a chain of sounds using their Playback Sources/Sinks.	82
Figure 5.9: Wave3D Item Sounds in Virtual 3D Space.	83
Figure 5.10: (a) Playlist Item, (b) Floating Playlist Window	85
Figure 5.11: (a) MIDI Module, (b) Instrument Item, (c) Track Item.	86
Figure 5.12: The Toolbox Panel contains a list of Signal Items.	88
Figure 5.13: (a) & (b) Meter Item, (c) History Item	89
Figure 5.14: (a) Threshold Item, (b-e) Transformation Graph Components	90
Figure 5.15: A demonstration of the Threshold Item.	92
Figure 5.16: A sequence of plateau adjustments (left, right, left).	92
Figure 5.17: A sequence of extent adjustments (left, right, left).	92
Figure 5.18: The family of vertical cubic functions.	93

Figure 5.19: The family of horizontal cubic functions.	93
Figure 5.20: Special case graphs. (a) Identity, (b) Inverse, (c) Constant.....	94
Figure 5.21: (a) 1-Gate Item, (b) 2-Gate Item	96
Figure 5.22: A sonic ecology demonstrating 1-Gate and 2-Gate Items.....	97
Figure 5.23: The Arithmetic Item with (a) two patches, (b) one patch and a constant.....	98
Figure 5.24: (a) Trigonometry Item, (b) Statistic Item	98
Figure 5.25: Utilizing the Trigonometry Item to perform 360° sound rotation.	100
Figure 5.26: A demonstration of the Statistic Item's maximum function.	100
Figure 5.27: A demonstration of the Accumulator Item.....	101
Figure 5.28: (a) Buffer Item, (b) Statistic Accumulator Item	101
Figure 5.29: (a) Time Cycle Item, (b-g) Various Time-scales and Graph Functions.....	103
Figure 5.30: Cross-fading two sounds between day and night.	104
Figure 5.31: (a) Using a Booster Item, (b) A simulation of Booster Item functionality.	105
Figure 6.1: Floor plan of gallery and camera/speaker deployment.	118

Chapter 1. Introduction

In this thesis, I consider the requirements, user interface and architecture of a system that lets an end-user configure a video-only stream so that it generates an audio soundscape. In particular, I present a programming by demonstration environment called Cambience. Using this system, end users can indicate regions of interest in live video that are collected from distributed sites, map visual characteristics of these regions onto audio samples, and assemble the audio stream in a way that creates a meaningful soundscape.

While such a system may seem somewhat unusual, there are two good motivating reasons for it. The first reason is perceptual. Video can only be understood if it is observed; it is very much a foreground display. If video can be translated into a meaningful audio stream, then people can perhaps hear and perceive events within it at the periphery, i.e., it becomes an ambient display. If the video channel is muted, this approach could even safeguard privacy. The second reason is experiential. If people's actions as seen by a camera are played back to them in real time as sound, then this creates an interactive installation.

When I first began this work, I was primarily motivated by problems with existing video-based media spaces (described shortly). It was only later that I realized that my solution could be applied to other settings as well, e.g., security and surveillance, interactive museum installations, interactive dance, and others. Consequently, I will use video-based media spaces as the primarily motivator for my work, where I will explain why sonification could mitigate some problems within it. I will also ground most of the example scenarios that I use to explain Cambience within the media space context. Along the way, I will introduce several other settings that could benefit from such a system, although not in as much detail.

1.1 Motivations

1.1.1 Video and Privacy

Video is being used increasingly in a way that allows a distant person to monitor one or more other people at other sites. In particular, video media spaces (VMS) are always-on video connections that link distance-separated people (Bly et al, 1993). The idea is that they offer collaborators a window into each other's work areas; people use the video image to maintain awareness of the actions and availability of others, and ultimately to move into conversation and interaction. For example, the video stream informs an observer if the "observee" is in their office, on the phone, looking away from the computer, concentrating on work, or conversing with others. Based on this information, the observer can then decide upon opportune moments to initiate casual interaction (Kraut et al, 1990; Wittaker et al, 1994).

While this sort of awareness has proven beneficial in a variety of organizational settings (Dourish and Bly, 1992), privacy can become a large issue for obvious reasons (Neustaedter et al, 2006; Boyle and Greenberg, 2005). As a consequence, many techniques have been developed to alter the video image in a way that can mask sensitive information while still revealing sufficient awareness cues (Boyle and Greenberg, 2005; Neustaedter et al, 2006; Smith and Hudson, 1995). While some of these techniques are shown to work for benign situations, (Boyle and Greenberg, 2005), they are ineffective when 'risky' situations appear in the video (Neustaedter et al, 2006; Boyle and Greenberg, 2005).

Privacy risks derive from many things. Some are a direct result of how a person appears in the video: from low risk (unattractive postures, picking noses, etc.) to high risk (nudity, embracing others). Some result from contextual differences, such as connections between a home office (and home occupants) to a workplace. For example, a person in a home office may be dressed inappropriately for a work setting, or other home occupants may come in and use that room for non-work purposes. Camera positioning may also

unintentionally reveal other information. If it captures a doorway, for example, one can see individuals as they walk by or enter the room.

While we are primarily motivated in this thesis by video media spaces, we recognize that *video-based surveillance systems* share mainly similar properties. These systems are also used to allow one person to monitor others, except in this case it is for security or safety reasons rather than for casual interaction. For example, one person can monitor a video feed of a personal space (such as an office or workroom) to see the presence, identity, and actions of possibly unwanted intruders. Another example is a video system used as a ‘baby monitor’. Depending on the situation, there is a fine line between warranted surveillance and privacy intrusions.

The challenge is: how can we provide people with awareness information, whether used for computer-mediated communication or security, while still safeguarding some degree of privacy? One possibility that we and others (Hindus et al, 1996) advocate, is to interpret the video as audio, such that the original video is not shown. People would ‘hear’ events in the video rather than see them. Examples include the arrival and departure of a person into the scene as ascending or descending footsteps, motions over time as a shuffling sound whose volume corresponds to the amount of motion, actions on a particular part of the scene as a distinctive sound, and so on. Depending on how this is done, the sounds can be fairly literal (i.e., the information details what is going on) or quite abstract (i.e., it hints at what may be happening); the choice of literal to abstract representation can, in turn, also regulate privacy.

1.1.2 Video as a Foreground Display

The premise behind both VMS and surveillance systems is that people are somehow monitoring the video contents of the distant site. This only works, of course, if the observers are actually attending the video. That is, video is a foreground display requiring people to focus on it. This can be problematic for several reasons.

First, watching video can become somewhat complex when, for example, a single person wants to monitor several video streams simultaneously. Second, even if they are watching, it may be easy to miss events of potential interest, either because they occur too quickly, or because they are visually small, or because the person's attention drifts momentarily. Third, it may be unreasonable to expect a person to watch the video channel because, within a media space for example, awareness of what is going on within the video is often a secondary task. Within security and surveillance (such as an in-store camera), people are often concerned with other primary tasks and objectives.

To make this more concrete, consider the *Community Bar* (CB), a groupware media space that allows a modest number of individuals to share and move into interaction over personal information (McEwan, 2006). It presents itself as a sidebar containing media items; each item can include things like public text chats, shared web pages, and photos. One very useful type of media item (the Presence Item, shown in numerous forms in Figure 1.1) lets people post their personal webcam video streams. While using their own computer, an observer may occasionally glance at other participants' video to see what they are doing, to assess if there has been any significant change since they last looked, and to decide what action to take, if any. The problem is that it is possible to miss events of interest if they occur between glances. Events can occur without that person's knowledge (Kate just visited Josh in his office), or when that person finally realizes that a significant event has occurred, it is too late to do anything about it (i.e., this could have been a good time to talk to both Josh and Kate, but Kate has just left). The fact that Community Bar can display quite a few video media items makes this even more problematic, as it supports multiple video feeds.

Instead of asking people to watch video, we can perhaps have them monitor changes in audio that are in turn driven by changes in video. A person can pursue their primary task, and changes to the ambient soundscape can perhaps attract attention or trigger interest. They may be able to interpret the event by knowing how it is associated with the sound, or the sound may provide sufficient cues to cause the person to glance at the video to see details of what is actually going on. Multiple video feeds could be handled as well; the

member could map each person's video feed into one or more personally meaningful soundscapes representing group activity. Within the domain of surveillance, such audio cues can serve as an attractor for security people, where it could help them detect events that occur when that person is either not looking at a particular feed, or if attention wanes.

1.1.3 Video as an Interactive Installation

Computers are being used increasingly within the arts to create interactive installations. Within such installations, participants – be it professional dancers, actors, or just members of the audience – become performers in the art piece, where their conscious and unconscious actions affect how it behaves. One challenge in such installations is how to do both input and output. Vision as captured in video streams is clearly a powerful input mechanism, for it can capture a performer's motion. The problem is that this visual input must be somehow translated into one or more other mediums – such as music, sound, light, etc. – that form the output of the installation.

One way that this has been done is to use the movements of dancers, actors, or even members of the audience to affect an audio “performance” in a meaningful or completely abstract way (Rokeby, 1986-1990). That is,

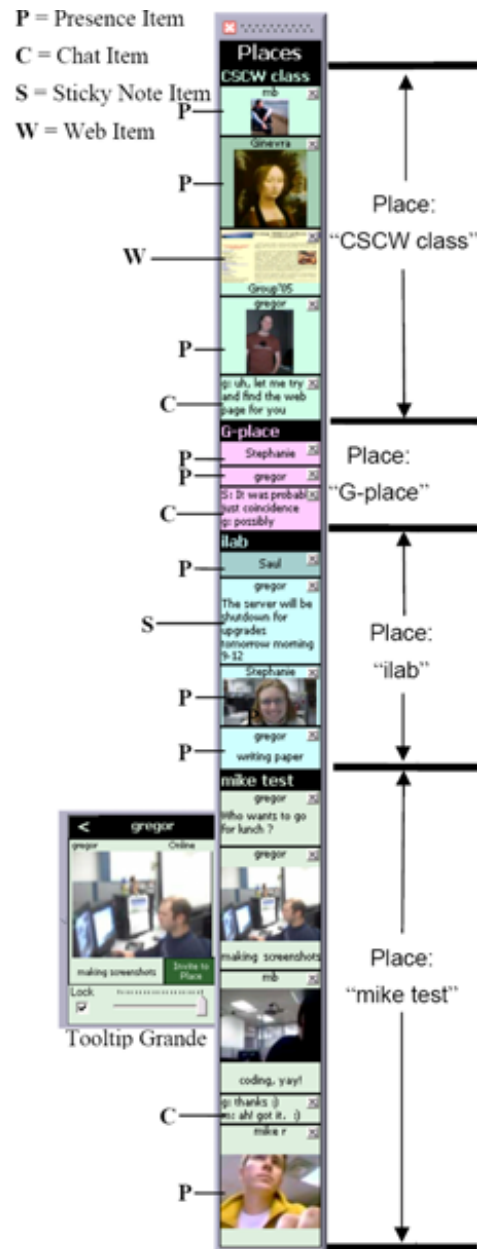


Figure 1.1: The Community bar from McEwan and Greenberg (2005).

sound is generated in real time from the actor’s video as “motion feedback”, and the actor can then modulate their acts to change the sound accordingly. For instance, a dancer can have their motions produce music, and in turn they can respond to that music to drive the dance (Fels, 1997).

There are a few ways this can be realized. Installations may rely on a single camera to produce an audio response that the actor can hear. Multiple cameras can be augmented in the same performance space, seeing different angles and areas, to respond to one or more co-located actors. Furthermore, multiple cameras can be used by multiple actors in distance separated locales to conduct a unified audio performance.

1.2 Goals

The problem that I explore in this thesis is how to allow a person to map features from multiple video streams, or parts of those video streams, into an audio space, and how they can then monitor the resulting soundscape. My primary goal is to create a visual programming environment that lets people:

- Indicate regions of interest in live video collected from distributed sites,
- Map characteristics of these video regions onto audio samples,
- Assemble these audio streams in a way that creates a meaningful soundscape,
- Monitor the immediate results in real time.

I will achieve this goal by designing, constructing and implementing the Cambience visual programming environment, and showing by example how it meets the above requirements.

1.3 Organizational Outline

Chapter 2 provides background in areas related to the thesis. First, I summarize previous work in media spaces, the primary motivator that drove my work. I introduce the idea of

abstract media spaces and abstract mappings, especially in regards to mapping video to audio. Next, I introduce sound: how we as humans perceive it, and how it can (and has) been used as a computer interface. Finally, I review visual programming, where I use prior work to develop a basic set of criteria for Cambience.

Chapter 3 introduces Cambience and its features as seen by the end user. I walk through a number of usage scenarios to demonstrate Cambience in action, where I show the steps necessary to set up cameras, to monitoring regions, and to construct and map a basic sonic ecology driven by activity in the monitored video regions.

Chapters 4 and 5 dive into greater detail. I provide a technical review of the Cambience framework, and discuss how it works as a distributed system. I revisit the idea of Visual Programming and explain how the basic structures in Cambience facilitate this. I also review Cambience's various Visual Programming building blocks, and how they can be attached to one another to create a video to audio mapping.

Chapter 6 briefly describes a handful of case studies of Cambience in actual use. While not a formal evaluation, it reveals some problems and prospects of the system. To further validate the power of Cambience, I show how Cambience can both replicate and improve upon the capabilities of a number of other proprietary video to audio mapping systems.

The thesis concludes with a review of goals, contributions and future work.

Chapter 2. Background and Related Work

As a video to audio tool, Cambience relies on many technical foundations: computer vision, sonification, sound engineering, and visual programming. As an application, Cambience has potential within several different niches: social awareness, media spaces, groupware, peripheral and ambient displays, abstract displays, and even augmented reality. Background to this range of knowledge is clearly too broad for a single review chapter. Consequently, I constrain my discussion to only those areas that directly motivated the creation of Cambience.

To set the scene, I first provide a brief background on media spaces, and why a sound-based media space has relevance. Next, I detail basic characteristics of sound. Finally I briefly review the concepts of Visual Programming, focusing on those that come closest to mapping video to audio.

2.1 Media Spaces

My conception of Cambience had its roots in social awareness systems such as Community Bar (McEwan, 2006) and the Notification Collage (Rounding, 2004). I originally developed it as an add-on or replacement of a video-based media space. Instead of having people monitor the visually demanding and potentially distracting video channel, I wanted to see how sound could help them maintain timely awareness of each other. Media spaces were previously introduced in §1.1, and this section provides additional detail.

2.1.1 Casual Interaction and Informal Awareness

Individuals with a common goal need to communicate with one another in order to complete their individual work, and integrate it with that of others. These people are *intimate collaborators*, defined by Greenberg and Kuzuoka (1999) and described by Rounding (2004) as “groups of people that have a real need for close coordination and communication.” Some of these interactions are planned ahead of time, with expectations about social conventions and desired outcomes. Meetings, focus sessions and classes are examples of *formal interactions* among these individuals. Far more prevalent, however, are *informal* or *casual interactions*. These are unplanned encounters that may be equally effective in accomplishing goals. They can involve serendipitous meetings of individuals in or near common areas, which present unique opportunities for collaboration.

As we go about our daily tasks we subconsciously keep a mental model of important features in the environment around us, using whatever details are available. When our lives and work are intertwined with other people, an awareness of their presence and actions can be vital; this information changes quite frequently and often has an impact on our own actions. This mental model is *informal awareness* (Gutwin, Greenberg and Roseman, 1996), and it is the major facilitating factor of casual interaction.

One obstacle to informal interaction that work groups must deal with quite often in today’s world is that of *distance separation*. Individuals who want or need to be intimate collaborators are often separated by geographic distances. In this case there is no possibility of awareness or chance encounters, making the prospect of casual interaction completely disappear. Even having workspaces on different floors of the same building can impede casual interaction because it can force collaborators to effortfully seek one another out when serendipitous interactions are infrequent. Furthermore, without an awareness of the availability of others, those efforts may be fruitless. When it becomes taxing to communicate, individuals will often defer matters until the next accidental meeting, or until things become critical (Dabbish and Kraut, 2004).

For example, let's talk about Kate and Josh, two researchers who often collaborate because they share an interest in a common piece of technology – a display wall. The display wall is just outside of Kate's office, and Josh's workspace is one floor down. For Kate, her proximity to the display wall gives her an informal awareness of who is using it, but she has no awareness of Josh because of their distance separation. Meanwhile, Josh neither has an awareness of Kate's availability, nor that of the display wall. Josh comes to visit Kate with the hope of talking to her regarding a paper they are co-authoring, but finds that Kate has left for lunch. Josh may periodically check back until he finds Kate available to speak with him. Josh also needs to use the display wall on occasion for his research. Inconveniently, he will sometimes find that other students are using it when he wants to. This forces him to book the time he needs with the wall so that he does not risk others displacing him. On the other hand, Kate can easily step out of her office and begin using it as soon as those other students are finished.

Many researchers have sought to break down the barrier of distance separation amongst intimate collaborators by electronically delivering awareness information, and offering a lightweight communication channel. The result is a Media Space, which I will formally introduce in the next section.

2.1.2 What is a Media Space?

At its inception, a Media Space was defined by Stults (1986) as, "An electronic setting in which groups of people can work together, even when they are not resident in the same place or present at the same time. ...people can create real-time visual and acoustic environments that span physically separate areas. They can also control the recording, accessing and replaying of images and sounds from those environments." Concisely, Media spaces are always-on electronic settings that connect groups of intimate collaborators who would otherwise be isolated by distance. Depending on the design of the media space it may facilitate not only group work, but casual or playful interaction, informal awareness, or any combination thereof.

Media spaces can deliver content visually, aurally, digitally, and in rare cases physically. The most basic form is a live audio/video link that allows individuals to see and hear one another, almost as if they were in the same room together. Xerox PARC (Figure 2.1) was responsible for a great deal of the early research surrounding the implementation and use of such a media space to connect the private offices of several employees at different job sites (Goodman and Abel, 1986; Abel, 1990; Stults, 1986/88; Bly and Minneman, 1990; Tang and Minneman, 1990; Buxton and Moran, 1990). Later research brought about the creation of CAVECAT (Mantei et al, 1991), an audio/video media space geared toward group meetings and casual encounters, which had built in limitations to address issues of privacy and surveillance.



Figure 2.1: Xerox PARC media space. Image taken from Bly et al, 1993.

Researchers have also experimented with the implications and affordances of multiple video feeds in a media space; most present a small number of video windows following a picture in picture display approach (Figure 2.2). Dourish and Bly's "Portholes" application (1992) shared intermittently updated still-images from the web cameras of a

large number of remote sites, with the goal of promoting general awareness of a community. This system uses the *overview model*, where, “in a manner somewhat similar to an overview of surveillance cameras, one can quickly and continuously get a sense of what is happening at a variety of different locations.” (Tang and Rua, 1994)

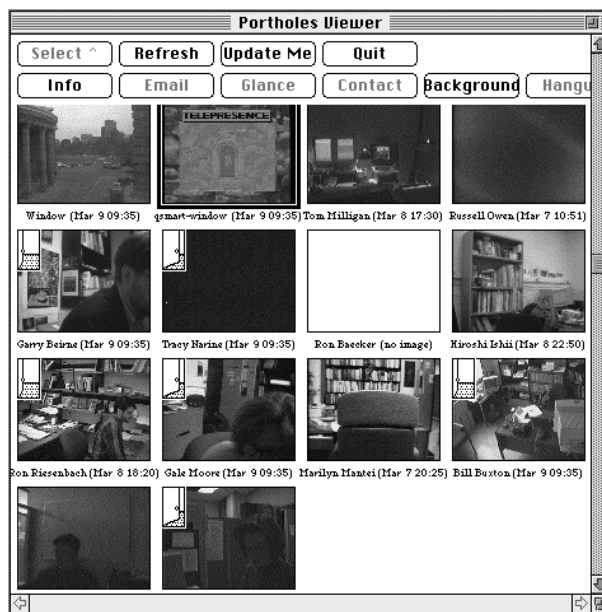


Figure 2.2: A screenshot of Portholes, taken from www.billbuxton.com.

More recent research has looked at uniting a diverse range of media forms within the same media space. Rounding’s *Notification Collage* (2004) allows for sharing of video, images, textual notes, desktop screenshots, image annotations, and much more. MacEwan’s *Community Bar* (2006), seen in Figure 1.1, allowed for sharing of similar media (with the inclusion of a basic audio channel) within user-designated locales.

2.1.3 Sound in Media Spaces

There are two different ways that sound is typically used in media spaces. The first is **always-on audio**. This approach captures and transmits actual sound from each remote site, and is left running throughout a person’s daily activities. It can be used across several sites to multiplex speech as in a conference call (Watts et al, 1996), but it can also

pick up incidental sound cues that reveal the actions and status of an individual even in the absence of video (e.g. shuffling papers, talking on the phone, sounding distant from the microphone when speaking etc.)

One experiment in a media space would be to remove the video channel and evaluate the effectiveness of the audio on its own. This was done with Thunderwire (Hindus et al, 1996), an always-on audio-only social media space. They report that it brought its users many advantages, including instant access to the expertise of co-workers, social planning, and informal awareness of others. However, it also had disadvantages regarding privacy, as an individual's personal telephone conversations became public to the group. Their findings were that: 1) audio can be sufficient for a usable media space, 2) audio spaces can lead to social spaces, and 3) the nature of these social spaces is affected by audio's affordances.

The second method is the use of **audio alarms and notifications** to represent the actions and activities of others. Tang and Rua's Montage system (1994) initially used a telephone metaphor that would sound an alarm when another person wanted to glance into one's office. This cue served as an explicit request that the host accept a connection. However this approach was deemed too jarring and the metaphor too limited in the way of awareness cues. Their next attempt involved video conferencing, which participants found heavyweight to initiate. Finally they used a hallway metaphor, where participants could leave their "office doors" open to allow others to "glance" into their workspaces. The person receiving the glance would get an advance audio notification that mimicked hearing someone physically approach your office from down the hallway.

The Arkola Simulation (Gaver, Smith, and O'Shea, 1991) was a shared artifact space that used sound to monitor joint activities, and broached a concept easily applicable to media spaces. This project electronically simulated a beverage bottling plant where two co-located individuals were expected to collaborate to ensure the plant ran smoothly. The simulation used **continuous audio feedback** to communicate the status of the plant. Sounds were present when the plant was running properly, but as problems began to

occur (e.g. due to incorrect actions by one person), the sounds changed and became more jarring. When the participants solved the problems, the sounds returned to normal.

2.1.4 Abstract Media Spaces

So far I have talked mainly about *literal* media spaces, where the video/audio captured at one location is seen or heard in a (mostly) unaltered form at any other location. *Abstraction* is the concept of reducing or altering detail in information to emphasize particular – usually important – aspects of it. The technology we use to capture, transmit, and display information imposes limitations on some aspects of literal information. For instance, extreme compression may be necessary to meet bandwidth limitations, but may cause digital artifacts to appear in video and audio. Although this is technically an abstraction from the original data, we do not consider it as such because it is our best attempt to represent the information accurately.

Abstraction can take many forms in a media space. Boyle (2005) used video distortion filters such as blur and pixilation to abstract video frames, with the goal of protecting privacy (Figure 2.3). Used correctly, these filters eliminated finer-grained details meant to be kept private while still giving an overall impression of what was happening in the setting. The problem was that there was no single filter setting that would be successful in all risky situations – sometimes sensitive details were still visible, while other times the meaningful information within a scene was completely eliminated. Users were not able to manage their privacy settings in anticipation of all events (e.g. in a home setting, the user’s partner unexpectedly shows up to give them a kiss), and privacy violations still occurred.

Abstraction can also be applied to sound. Smith and Hudson (1995) distort speech to mask the exact words spoken in a media space, while still giving the sense of the speaker and timing in the conversation. They termed this “Low Disturbance Audio” on the premise that overhearing such a conversation, when not directly involved with it, would not be as distracting or as privacy-invasive if one is unable to follow the meaning.

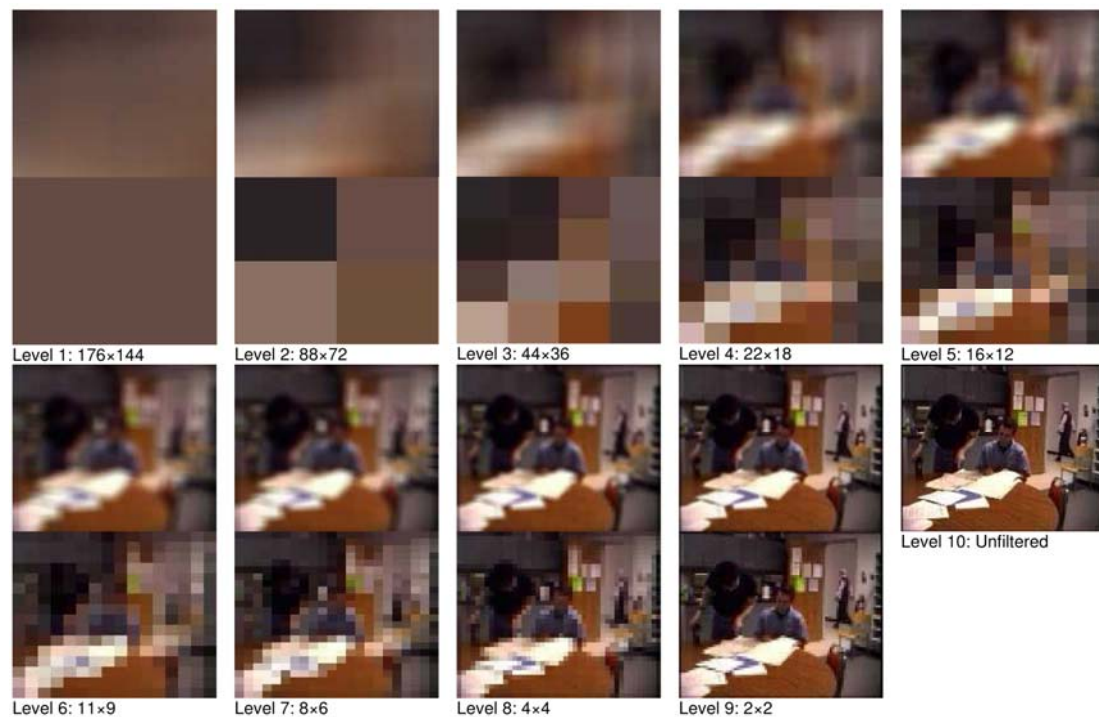


Figure 2.3: Blur (top) and pixelize (bottom) filters, taken from Boyle, 2005.

Pedersen and Sokoler (1997) took a much bigger step into media space abstraction in their system entitled AROMA (Abstract Representation Of presence supporting Mutual Awareness). Many of the program outputs did not resemble their original inputs at all, although a predictable correspondence was still present. For instance, they used input from a video camera and microphone to synthesize ocean wave sounds, to cause a Peltier element to produce heat, to spin a toy merry-go-round, and to render a cloud animation all at once.

Many more examples of abstract media spaces exist, but they tend to be grounded in the arts world rather than the scientific community. One prominent example that I have encountered is the Very Nervous System, which uses gestural body movements to guide music synthesis. Examples of his work include dancers creating their own soundscape in response to their motion, and disabled people creating music as a band, by the interplay of their limited gestures. This interactive art installation was developed by David Rokeby between 1986 and 1990, and evolved through many different forms.

2.1.5 Abstract Mappings

Media forms lend themselves well to transformations that produce a result of the same media form (e.g. video \rightarrow video, audio \rightarrow audio, etc.), however there are no natural correspondences between different media forms (e.g. video \rightarrow audio, audio \rightarrow video, etc.) In order to achieve a cause and effect relationship across media forms, it is necessary for the programmer to devise an algorithm to abstract the input media to something that can equate to the output media. I refer to such a correspondence as a *mapping*, and a collection of correspondences as a *mapping scheme*. More generally, I use the term *sonic ecology* to describe the audio output of either, however this term will be better explained later on.

Returning to the AROMA system, some examples of its mappings might be as follows (although these are not necessarily the way the system actually worked):

- The loudness of sound picked up by the microphone causes the merry-go-round to spin faster.
- More activity picked up by the camera causes the waves to grow louder.
- The overall brightness of the captured video frames control the heat produced by the Peltier element.
- Higher-pitched sounds captured by the microphone result in smaller cloud formations, while lower pitch sounds cause larger ones.

In each of these examples, a particular feature is extracted from the input medium and fed to a control parameter of the output medium. This is usually in the form of a numeric value that has a definite possible range of values.

For example, a video frame's maximum brightness reading would occur when all pixels are white (R=255, G=255, B=255), and reach its minimum when all are black (R=0, G=0, B=0). We may (naively) choose the brightness value to be the average of the RGB values, in which case the range of the brightness would be between 0 and 255. Thus we

have what I refer to as a *span*, where $\text{min}=0$, $\text{max}=255$. We can now map this value to (say) heat produced by Aroma's Peltier element. An algebraic transformation converts the range of 0-255 to (say) the digital minimum and maximum values of the Peltier element; the maximum value would cause it to be on at full current, while its minimum would have it off completely. The actual transformation need not be 1:1; it could follow any function as long as the output values map into the legal range of input values.

Who should assign these mappings? Currently in abstract media spaces and interactive installations discussed previously, mappings are determined by the programmer. Yet we believe the users of the system may want to try different schemes to determine the mappings that are meaningful and comfortable for them personally. Thus there needs to be some mechanism to juggle mappings without requiring programming expertise.

As we will see later, one of my goals is to facilitate customization of cross-media mappings (particularly between video and audio) in a way that is more accessible to a non-programmer.

2.2 Sound

In this thesis, I limit the notion of sound to people's perception of vibrations that travel to their ears through a medium such as air, water, etc. This leads to two important properties of sound: its acoustics (i.e., the basic physics of sound), and psychoacoustic (i.e., how people perceive sound). Buxton, Gaver, and Bly (1989) provide an excellent introduction to both of these properties, including implications for interface design. The remainder of this section summarizes the main concepts that they present.

2.2.1 Acoustics

Sound is our perception of pressure variations that propagate in an elastic medium – typically the air. A sound's *waveform* is a graph of the amplitude of pressure variation over time, typically a sinusoidal shape; this is how most sound editors visually display a

sound on the screen. Yet visually inspecting a waveform does not readily reveal its properties, e.g., two sounds with similar waveforms may be perceived as sounding very different, while two sounds with different waveforms may be perceived as sounding very similar. The reason is that any complex sustained sound is actually a combination of a basic building block that can vary by the following properties:

- *Frequency* refers to the relative number of oscillations in a waveform, and is usually equated to pitch. An *octave* is a grouping of the standard 8 musical notes, which are closest to one another in frequency. However, corresponding notes in different octaves tend to sound more similar than the notes within a single octave.
- *Amplitude* is the strength of an oscillation, and can be read as the peak distance from the origin line. Amplitude is always mirrored across the origin, and is usually equated to loudness.
- *Phase* is the temporal positioning or offset of a waveform. Humans do not perceive differences in phase. The appearance of a waveform can vary a great deal because of this property, but to us the difference is imperceptible.

Using Fourier analysis, a sound can be portrayed as a *spectral plot*. As seen in Figure 2.4, the spectral plots show amplitude along the vertical axis vs. frequency on the horizontal axis. Each band in this graph, also known as a *partial*, can be thought of as a sine wave at the particular frequency with the depicted amplitude. This provides a better means to compare sounds but is limited in that it can only represent an instant, whereas spectral properties in a sound usually vary over time.

Another important concept is that of a sound *envelope*: the variation of a single property over time. For example, the change in amplitude of a naturally occurring sound – its amplitude envelope – typically has four stages that can be seen when inspecting the silhouette of its waveform (Figure 2.5).

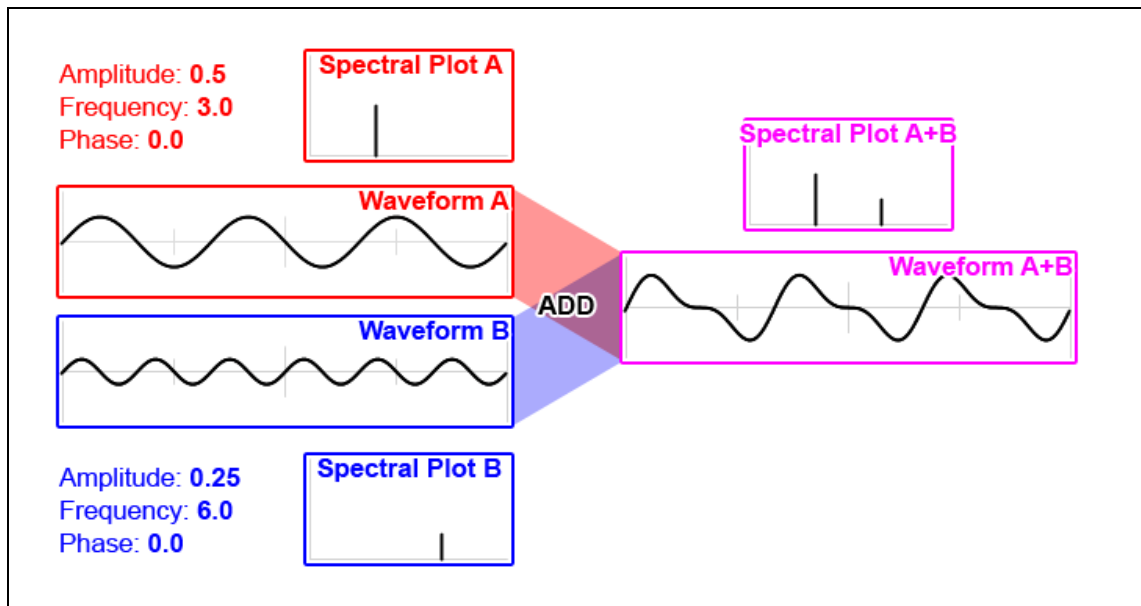


Figure 2.4: Combining two waveforms and corresponding spectral plots.

- *Attack* refers to the very beginning of a sound, where a waveform transitions from silence to an initial peak in amplitude.
- *Decay* is the slight diminishing of the initial amplitude peak.
- *Sustain* is a relatively level continuation of the sound, typically exhibiting lower amplitude than the initial attack. As the name suggests, it is this part of the waveform that is extended when a sound is held for longer durations.
- *Release* is the tail end of the waveform that transitions back to silence.

The spectral plot of a sound also changes over time, and it is possible to create a time varying spectral plot to show envelopes for each partial in the frequency spectrum (Figure 2.6). The relative change of partials over time is known as *timbre*, and is the basic essence of how humans identify sounds. Time variation is more of a distinguishing feature than the actual frequencies of the partials, as demonstrated by the ability to recognize and group similar sounds (produced by a musical instrument, for instance) at different pitches.

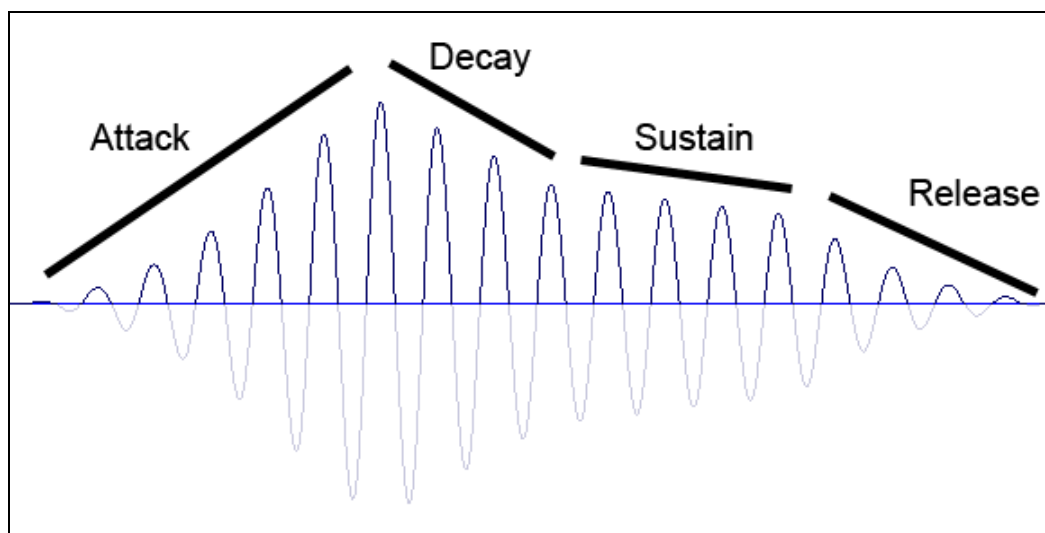


Figure 2.5: An example of an ADSR amplitude envelope.

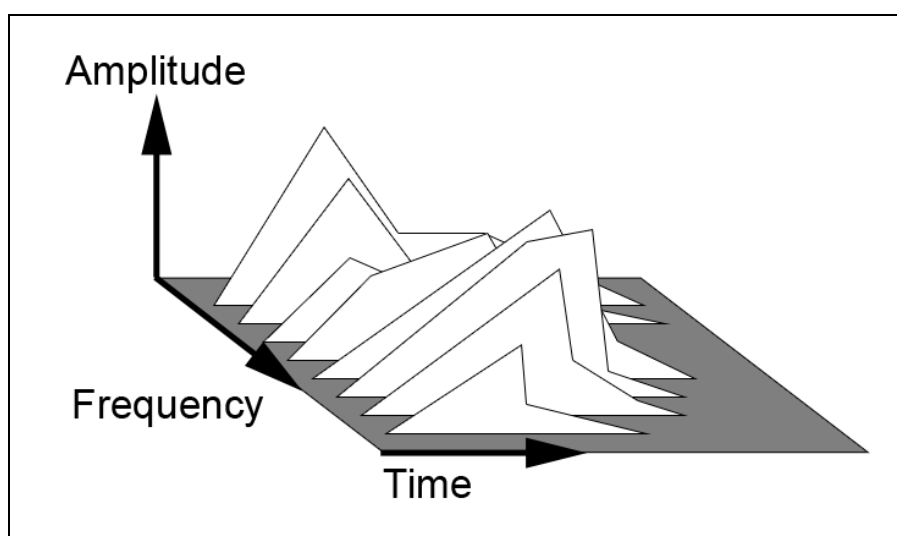


Figure 2.6: A time varying spectral plot.

2.2.2 Psychoacoustics

While people's perception of sound is obviously related to its acoustic properties, there are several properties that affect how that sound is heard.

People generally hear *pitch* as a logarithmic function of frequency. However, pitch can be affected by other factors such as loudness and timbre. *Bright* sounds have relatively greater amplitudes in their higher frequencies, while *dull* sounds are more prominent in their lower frequencies.

Loudness generally corresponds to amplitude. However, people's perception of loudness also depends on frequency, duration, and bandwidth. *Bandwidth* is the broadness of the range of frequency bands in a sound, and a larger bandwidth can result in a louder sound. *Critical Bands* are regions within the frequency spectrum, approximately a third of an octave wide. This property interacts with loudness, and is a major factor in sound masking – where one sound cannot be heard due to the presence of another more prominent sound. For instance, the roar of a waterfall would mask the trickle of a stream because the two sounds occupy the same critical bands, but the waterfall is much louder.

Duration corresponds to physical duration, yet people's perception of it can be affected by the onset of a sound's attack. This is because a sound is not perceived to start until a certain level of energy is reached.

As mentioned earlier, *Timbre* (also known as *tone color*) is the characteristic of a sound that makes it unique from other sounds. There are many factors that contribute to the overall sense of timbre, however only a few of these are understood.

Of course, sounds do not occur in isolation, and people's perception of multiple sounds depends on how they mix together. *Streaming* is our ability to separate (or fuse) sounds from multiple sources into perceptual streams. For example, when listening to music we can distinguish percussion instruments from melodic instruments even though they arrive at our ears as a single sound. Similarly, we hear a trumpet section or string section as a

single source rather than a multitude of separate instruments. Because the individual sounds are very similar and occur at the same time, they are mentally fused as a single stream.

2.2.3 Changing Sound

People's perception of sound also depends on how it fits within their surroundings. *Changing sound* conveys information about activity in the surrounding environment. People are 'wired' to seek out the meaning of this stimulus to decide if any action is necessary. For instance, the rustling of leaves, creaking of wood and snapping of branches might be indicative of a falling tree.

If these sounds quickly get louder it may signify that the tree is falling toward the listener, in which case they would divert their visual attention to assess whether they need to move in order to avoid being injured. The rate of change is a deciding factor of how much attention a sound demands: slowly changing sounds are more likely to fall to the background while quickly changing sounds grab our attention immediately. If a background sound changes quickly, it moves into our foreground attention; likewise, if a foreground sound slows or stops changing, it returns to the background.

Individual sounds can change in several different ways: volume, frequency, relative position, environmental effects, and timbre. Volume and position can interact because we judge relative position (a combination of distance and direction) by the volume of the sound in each ear. Frequency refers to the rate of a sound's playback, and affects pitch and speed. Environmental effects refer to the addition of various degrees of echo, reverberation, muffling etc. that affect the perception of the environment surrounding the sound source. Timbre is much more complex; when repeating a pre-recorded sound we have little control over timbre. However, if a sound is being generated or synthesized, it can be broken down into a number of parameters that allow for timbre variation. For example, Conversy (1998) talks about synthesizing wind and wave sounds with parameters such as strength, wave size, wave shape, and beach material (sand vs. rock).

In addition, Gaver (1993) talks about the synthesis of impacting, breaking, bouncing, spilling, scraping and machine sounds, all with unique parameters that affect timbre.

2.2.4 Sound as an interface

Buxton, Gaver and Bly (1989) identify sounds that help us as information, while those that impede us are considered noise. Unfortunately one person's information may be someone else's noise. This is especially true in shared office environments, where many people eschew the potential benefits of sound as feedback, out of politeness: they turn down or turn off their speakers, or apologize when an unexpected sound is heard by others. People can often make do without an audio feedback channel because sound is usually designed as a redundant add-on to the interface. In safety-critical systems, however, sound is used as an alarm that indicates a note-worthy event and we direct our vision to the screen to discover its cause.

To illustrate how this applies to digital environments, Gaver, Smith, and O'Shea (1991) created the ARKola Bottling Plant simulation that utilized various sounds as feedback about its inner workings. Although users were given a visual representation of the plant, changes in the audio feedback were more effective in alerting them to undesirable situations that needed their attention to correct.

In spite of Gaver's findings, most software designers have not exploited the better feedback possibilities of audio (excepting games). Perhaps this was because historically sound was hard to produce, or because computers were typically used in shared office space. Yet the world is changing, and many people now use computers in personal environments or with headphones. It is now quite common to listen to music while working. In these settings, detailed audio cues can become welcome, helpful, engaging and even pleasurable when we do not have to worry about disturbing others. Indeed, people seem to have little trouble listening to 'properly presented' sounds – such as music or symbiotic sounds – in parallel to other tasks. They seem to be able to shift between these two parallel channels with little effort.

Some researchers have made use of this affordance in recent times. Mynatt et al (1998) created *Audio Aura*, a system that presents serendipitous information through background auditory cues. It leverages people's physical actions - such as walking down a hallway or stopping in front of someone's office – to decide when to present useful but non-critical information on the periphery. Physical actions are detected by the use of active badges and networked sensors, while audio is delivered to individuals via portable wireless headsets. Ishii et al's *ambientROOM* (1998) provided a “subtle but audible soundtrack of birds and rainfall,” whose volume and density could be used to represent unread E-mail messages, the value of a stock portfolio, etc. Alexanderson (2004) identifies three types of continuous processes that lend themselves well to monitoring via natural sounds: “to give feedback about users' actions when interacting with a computer, auditory notification of system events, and awareness of other people's activities and work.”

My particular interest is to offer sound as a parallel channel that portrays information that is typically unrelated to what one is doing on a screen. The sound is generally in the background of people's attention; while they can attend to it if they want, they can easily ‘tune out’ its peripheral information when concentrating on a primary task. Yet, when interesting events happen and the sound changes, people perceive these changes and thus attend to it in the foreground of their attention.

2.3 Visual Programming

Visual programming refers to the act of defining program behavior through the arrangement of visual elements in 2 or more dimensions (Myers, 1986). This differs from conventional programming which is mostly textual, and Visual Development Environments which are concerned with GUI layout. I refer to the application that facilitates visual programming as a Visual Programming Environment (VPE), whereas the realm of possible behaviors that can be defined in that environment is the Visual Programming Language (VPL).

Visual programming typically looks like a set of boxes that represent instances of objects or functions, connected by arrows that represent control or data flow, laid out in a 2D workspace. The user chooses from a list of available objects or functions, and creates instances of the ones they need in the workspace. Then they define the interaction of these objects (more generally the behavior of the program) by connecting them together.

In a VPL that uses control flow, the connections between objects define the order in which each object (or function) is “executed”. In contrast, data flow connections define how data is passed between properties of objects (or functions). The data that an object produces can be of various data types, and thus connections can only be made between properties with the same data type.

Execution in a VPL is either interactive or batch (Myers, 1986). A batch system compiles or locks the underlying program during execution – changes are ignored or disallowed until execution ceases. In contrast, an interactive system allows a running program to be changed and its execution is dynamically affected.

A VPL can never be as flexible, efficient, or as general purpose as a conventional programming language such as C++, Visual Basic, C# etc. However, only a small percentage of the general population has the knowledge necessary to use conventional programming languages (Gould and Finzer, 1984). One advantage of VP is the potential for abstraction, in order to hide lower-level details that might be confusing to a novice (Myers, 1986). Research has shown that users without programming experience can still understand and manipulate VPLs to create complex programs (Halbert, 1984; Smith, 1977; Choi and Kimura, 1986). For instance, *Show and Tell* was developed for use by school children (Kimura et al, 1986, 1990). Incidentally, Kimura also states that “novice users find dataflow easier to understand than control flow.” (1993)

Visual Programming is a good fit for Video to Audio Mapping. The objects are building blocks that, like Lego, can be combined according to certain rules, and usually in too many configurations to name. Thus, a mapping could be created within the VPL that

even the creator of the VPE may not have anticipated. The factors that I examine to distinguish one Visual Programming Environment from another are:

- **Interface** - the design of the framework used to create and connect objects in the workspace, and execute the resulting behavior.
- **Language Elements** - the repertoire of available objects and the individual functionalities that they provide.
- **Level of Detail** - the details that the interface and language elements may hide, sacrificing fine-grained control for ease of use.

I define a VPE as being more *powerful* than another if it has a larger VPL (ie. a larger range of possible behaviors) than another. However there is often a tradeoff between power and level of detail: more powerful VPEs often have higher levels of detail, making them more complex. Higher complexity can also result from the design of the interface, the quantity and design of language elements, and the number of distinct data types that can be passed between objects (in data flow VPLs). Having to deal with greater complexity places more demand on the user when learning the language and defining program behavior – thus it is something we wish to minimize.

In the next section, I expand my three distinguishing factors into a set of criteria that must be met to satisfy my vision of a video to audio mapping system for a more general audience.

2.3.1 VPLs for Mapping Video to Audio

This thesis is primarily concerned with creating a VPE that maps properties of a distributed live video to audio in order to create a soundscape. Within this context, there are a variety of systems that can do this at some level. None are dedicated primarily to video to audio mapping, but most have facilities that make at least a rudimentary mapping possible. Thus I defined the following design criteria that a VPE must satisfy in order to meet my particular goals (§1.2):

- It must handle **networking as part of the interface** rather than as a function of the language elements, to allow ad-hoc program changes without networking interruptions.
- It must **function as a media space**, where live-captured webcam frames are shared among multiple distance-separated collaborators, and updated regularly.
- It must **unify construction and execution modes (interactive not batch)**, so that users can receive immediate feedback as they compose their visual program.
- It must **hide unnecessary low-level details** by creating high-level designs for the language elements, specifically purposing them for live video input and dynamic audio output.

I examined a handful of prominent multimedia-oriented Visual Programming Environments (Table 2.1). While all three of them are very powerful industry-standard multimedia oriented VPLs, they fundamentally do not meet my criteria outlined above. **Isadora** did not have networking functionality, and thus could not function as a media space. **Eyesweb** and **MaxMSP + Jitter** handled networking functionality as part of the language elements. Two of the three had separate construction and execution modes, while all had levels of complexity that were daunting to learn even for me, as a programmer.

Thus, a custom-built VPE was necessary to meet the goals of my thesis.

Program	Interface	Language Elements	Complexity
Isadora	Construct/Execute Modes	Live Video Input Dynamic Audio Output	Medium
Eyesweb	Construct/Execute Modes	Networking Pre-Recorded Video Input Dynamic Audio Output	High
MaxMSP + Jitter	Interactive Execution	Networking Live Video Input Dynamic Audio Output	High

Table 2.1: Summary table of desired Visual Programming Language features.

2.4 Summary

In this chapter we examined three major topics:

- **Media Spaces** – casual interaction, informal awareness, abstraction, and existing applications of sound.
- **Sound** – acoustics (physical properties of sound), psychoacoustics (how we perceive sound), changing sound and how it impacts attention, and the current paradigm of sound at the interface.
- **Visual Programming** – environments versus languages, distinguishing features, my criteria for an effective video to audio mapping environment, and a brief review of some existing visual programming environments that did not satisfy these criteria.

The remainder of this thesis revolves around the custom Visual Programming Environment that I created to accomplish my goal. This application was designed to satisfy the criteria defined in the previous section.

In the next chapter, we will go through a usage scenario that will introduce the interface of Cambience and briefly touch on its capabilities. Afterward, Chapter 4 will overview Cambience's visual programming language, followed by Chapter 5 which will explore the finer details of how it can be used to create mappings from video to audio.

Chapter 3. Introducing Cambience through Scenarios of Use

This chapter introduces the basic features of Cambience as seen by endeavors through several scenarios. As we saw in Chapter 2, there is a great deal of research that has revolved around the value of interpersonal awareness in situations where individuals engage in casual interaction (Gutwin et al, 1996; Greenberg and Kuzuoka, 1999; Dabbish and Kraut, 2004; Rounding, 2004; McEwan, 2006). Our scenarios will center around this context.

Consider the case of two researchers, Josh and Kate, whose offices are on different floors of a building as shown in Figure 3.1. Recently, they began an intensive project where they work together developing software for a large piece of equipment – a high resolution display wall. Their work involves many joint tasks: writing proposals, papers and software documentation; coming to consensus about many issues; coordinating not only their activities but also those of others working on the project, and so on. Yet this is not such an easy relationship to maintain. Each researcher has other work obligations, as well as a need to manage their own family, friends, and social life. For that reason, these two individuals would like to remain in relatively close contact, while still maintaining a degree of independence from one another. As well, they need to keep track of the Display Wall equipment and how others are using it. If they were colocated by the wall, much of this would be achieved by both being aware of what the other was doing, whether they were available for spontaneous casual meetings, as well as who was using the wall.

Because of their distance separation, Josh and Kate decide to use Cambience to facilitate their awareness and interactions around the wall. Several scenarios illustrate how they set up and create a Cambience-based media space from the ground up. Required equipment

is one or more standard computers, a camera for each (i.e., a cheap web cam), the Cambience software, and speakers or head phones.

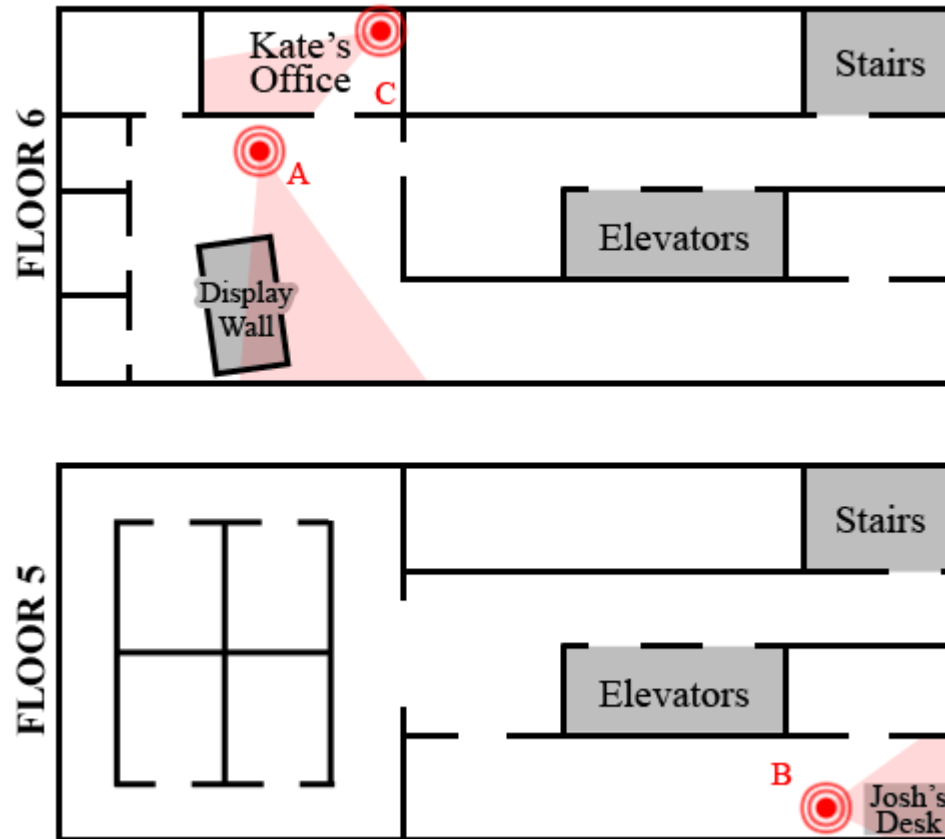


Figure 3.1: Floor layout for Kate and Josh. Cameras are marked as red dots.

3.1 Scenario 1

3.1.1 Step 1 – Physical Setup

The first task is to set up a Cambience server whose job it will be to monitor the display wall, and to act as a central point that Kate and Josh can connect to from other computers when they are interested in monitoring the activities around the wall.

Kate takes responsibility for this, as she is seated near the display wall. She positions the computer near the area of interest, and then downloads and installs the Cambience software and the SFX Library add-on (a set of useful sound clips). This takes only a few moments.

She designates this computer as the *Cambience Server*, a central point to which she and Josh can connect. She also decides to use this server as a client that captures the display wall. She attaches the webcam (“A” in Figure 3.1) to the computer and orients the camera so that the display wall and its surrounding area are in the camera’s field of view.

3.1.2 Step 2 – Configuring Regions of Interest

The next task is to partition the video image into several distinct visual regions of interest. These regions will later be configured to track particular activities occurring within them, and to convert them into sounds that distant people will monitor.

Currently, Camera A in Figure 3.1 is oriented to capture the video scene represented in Figure 3.2. Kate starts Cambience, and sees the display as illustrated in Figure 3.3. The live video image of the Local Camera View is displayed in the upper left corner. Kate examines the video, and decides to create three regions, each of which will be used to monitor particular activities. Figure 3.2 shows three prototypical uses that will be captured by these regions.

- The *front* region will capture people who are moving around the area in front of the wall (Figure 3.2a).
- The *displayWall* region will capture people close to or touching the wall’s surface (Figure 3.2b).
- The *behind* region will capture the area behind the wall; people usually enter this area to access the display wall for repair or projector calibration (Figure 3.2c).

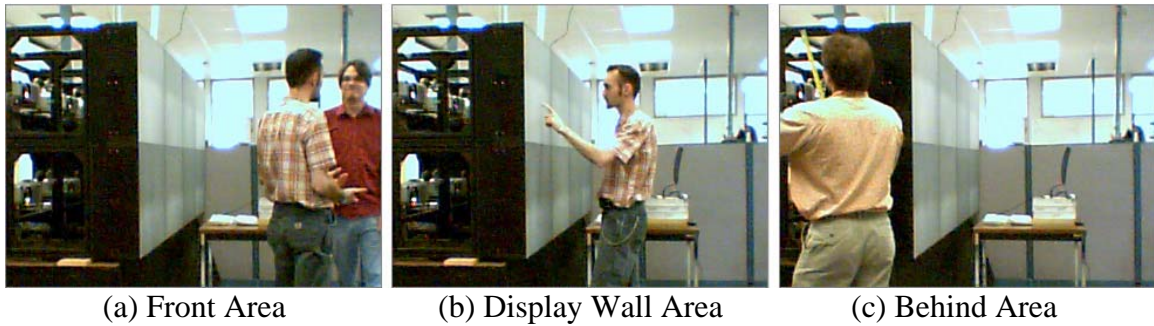


Figure 3.2: Three types of activity that Kate and Josh wish to monitor.

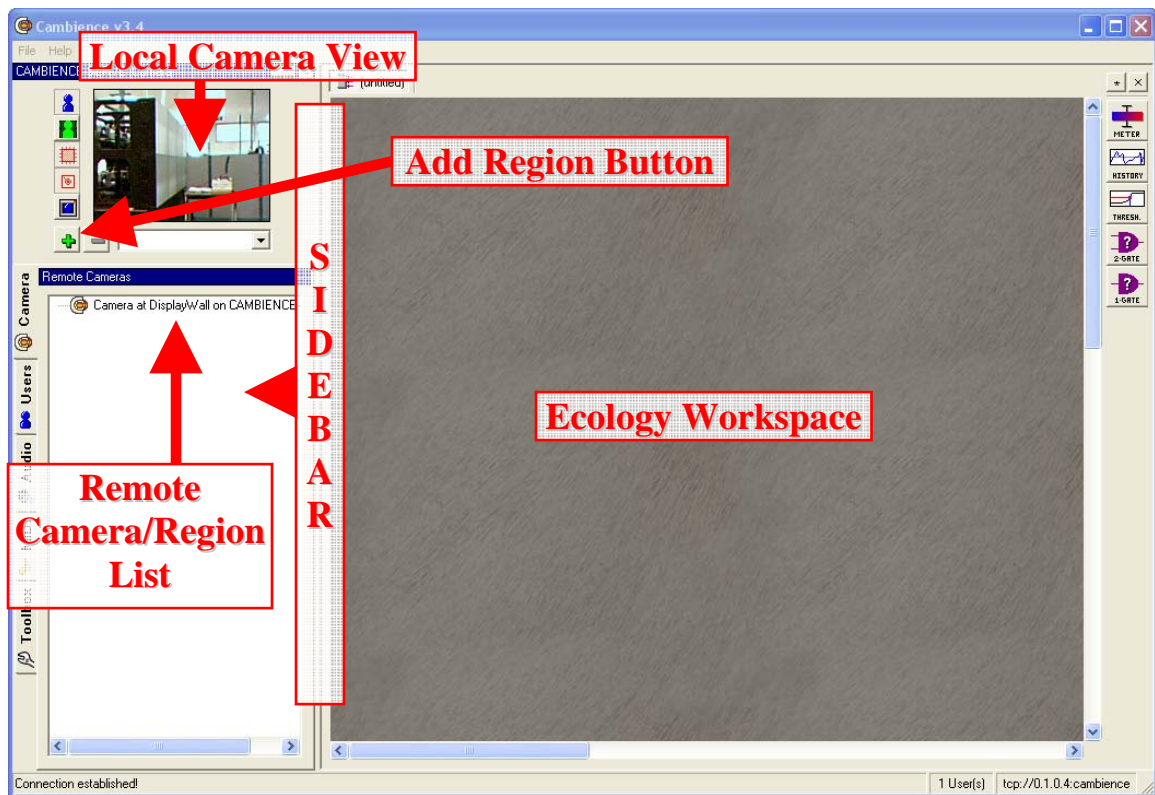


Figure 3.3: The Cambience main window.

To create these regions, Kate presses the ‘Add Region’ button (seen as a ‘+’ sign below the live video), and drags out a rectangle atop the video to surround the area of interest. Figure 3.4a shows a close-up of this. As seen in the Figure, the region appears atop the video frame as an untitled, relocatable and resizable box. At the same time, the ‘untitled’

entry is shown in the Region Properties Pane below. Kate can name and describe what this region is for; in this case, she calls it ‘front’ and describes it as ‘The area in front of the display wall’ (Figure 3.4b). Similarly, she creates names and annotates regions for surrounding the *displayWall* and *behind* areas on the video (Figure 3.5a and b).

As a side note, the Remote Cameras pane, underneath the Regions Property pane, displays both the camera and the regions that were just created. When Kate or any other person selects a camera from the list (including cameras connected to other computers), they get a view of that camera and the regions within it, in the Selected Remote Camera pane below (Figure 3.5b). They can also select a particular region to see its description (not shown). This is important for Kate and others – in particular Josh – when more than one camera is available containing multiple regions. It enables them to determine the presence and actions of individuals that are working within these regions around the video wall, as discussed shortly.

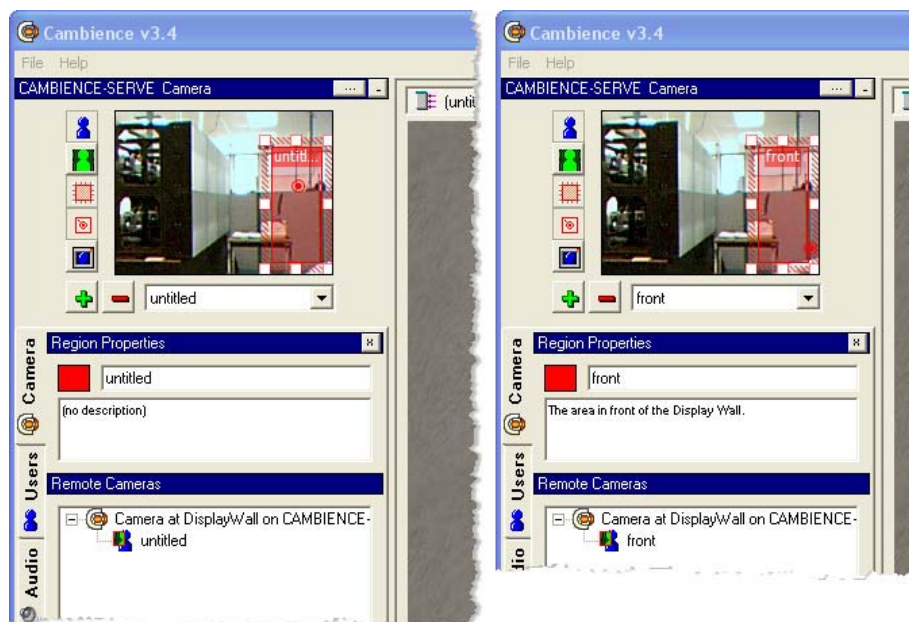


Figure 3.4: (a) Creating the front region, (b) Naming the front region.

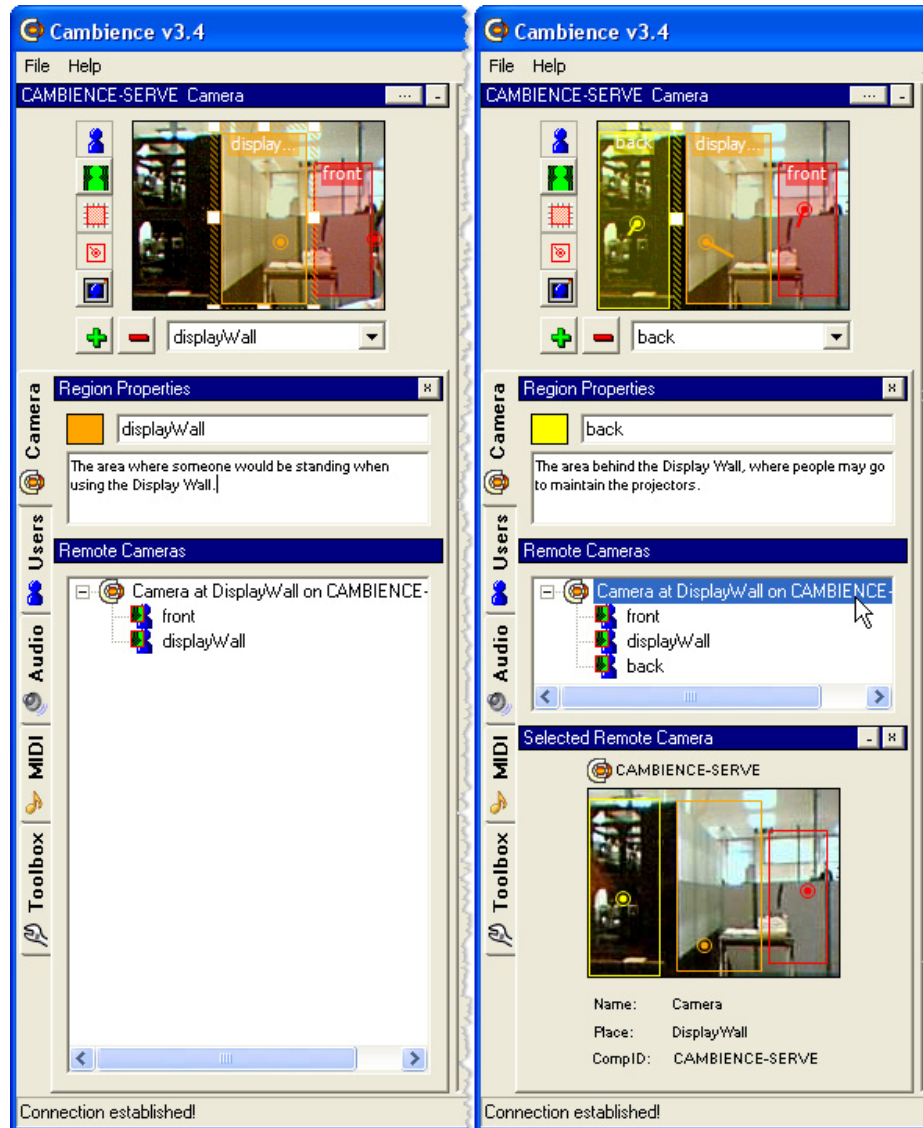


Figure 3.5:(a) Naming the Region, (b) Camera Preview.

3.1.3 Step 3 – Monitoring Activities in Regions

In order for Kate to start working with the data collected by a region, she must create a Region Item in the Ecology Workspace pane (the large textured area as labeled in Figure 3.3). She does this simply by clicking and dragging one of the regions from the Remote Cameras list to the workspace. Figure 3.6 shows a newly created Region Item for the *front* region.

Notice that in the Remote Cameras list, the text for the *front* region now appears in purple. This indicates that the *front* region already exists in the workspace, and a duplicate cannot be created if it is dragged to the workspace again.

When someone physically enters the *front* region, Kate can see the colored bars in the corresponding Region Item change. These indicate certain activities in the scene. For instance, Figure 3.6 shows the following:

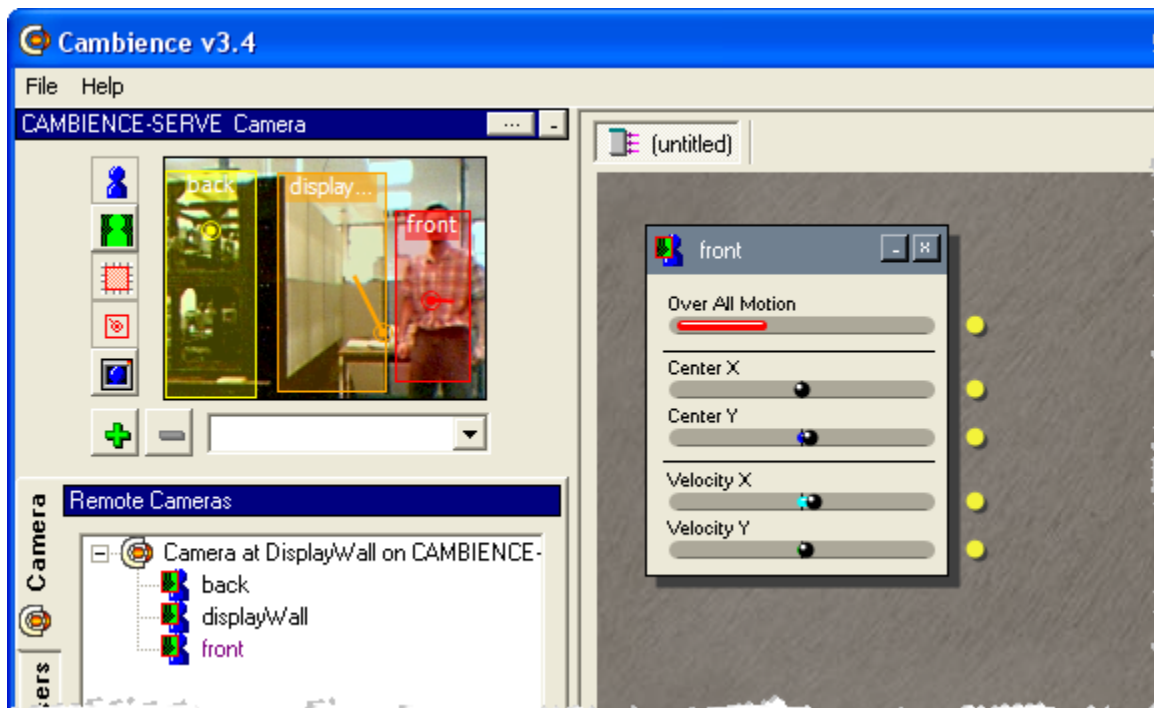


Figure 3.6: Creating a Region Item. A person enters the front region.

- The Over All Motion measurement jumps up as someone moves within the region bounds.
- The Center X and Center Y bars are almost perfectly centered because the person is in the middle of the region. Alternately, this is also indicated by the red dot in the middle of the *front* region in the Local Camera View.

- The Velocity X reading jumps slightly because the person has just stepped into the region from the right side. The velocity is also indicated by the red line from the center dot, as seen in the *front* region in Figure 3.6.

Similarly, Kate creates Region Items for the *displayWall* and *back* regions (Figure 3.7a and b). In the next step we will show how she can now use the Region Items to create mappings to particular sounds.

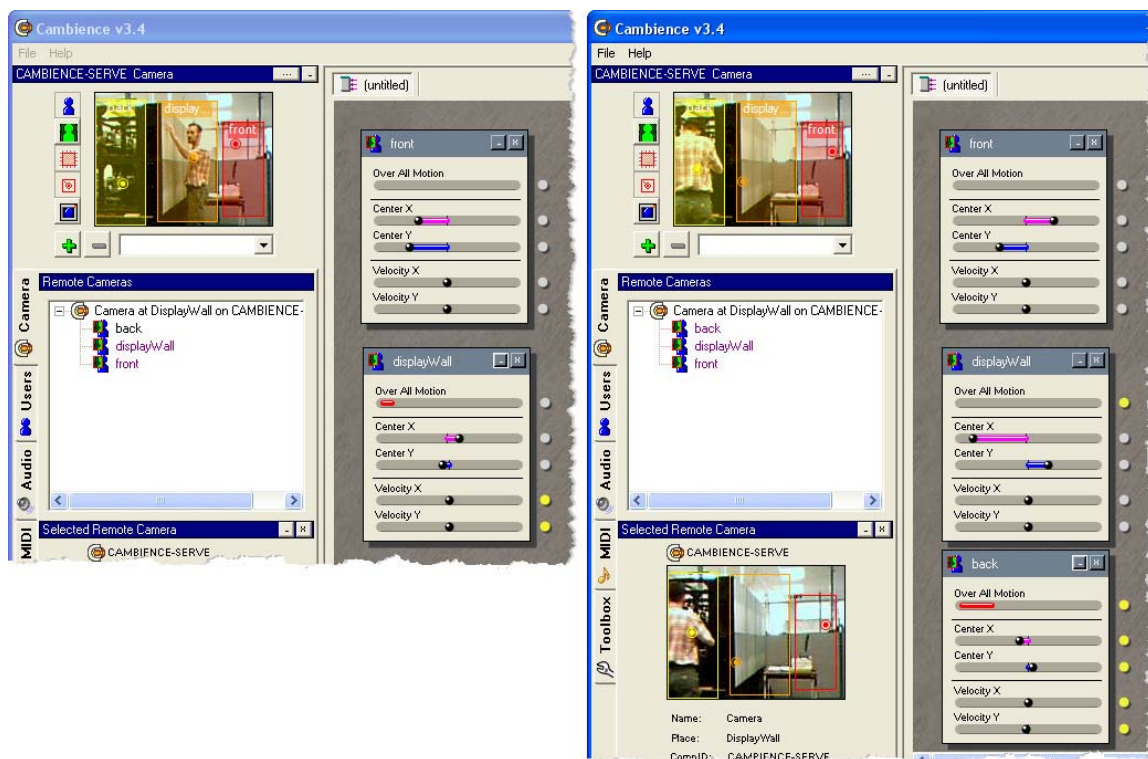


Figure 3.7: Adding and activating (a) displaywall, and (b) behind regions.

3.1.4 Step 4 – Mapping Activities to Sounds

Kate's next task is to create a "sonic ecology", where the properties of one or more playing sounds are mapped onto particular measurements provided by the Region Items. In this case, she will only use the Over All Motion and Center X values.

To do so, she could drag her own audio files from the desktop to the workspace (Cambience supports several audio file types, such as WAV, MP3, and M3U). However, she decides instead to use Cambience's built-in Audio Browser to select from sounds that came with the Cambience SFX Library package. She clicks the Audio Tab (as labeled in Figure 3.8) and the Audio Browser appears in the sidebar underneath the Local Camera View. She selects *lake_ebb.wav*, which appears under the Media Files classification in the *nature* subfolder, and drags it to the workspace.

As Figure 3.9a illustrates, a Wave3D Item representing that file appears; its default properties are displayed in its various controls (e.g. Volume is set to maximum). Kate uses this item to set the sound properties. She presses the play button to preview the sound, and then toggles the playback mode to looping (the circular arrow in Figure 3.9b). She also turns down the volume initially to give it a more ambient feel.

Next, she wants to map the Over All Motion property of the *front* region to this sound. If she wanted to, she could connect the Over All Motion of the *front* Region Item to affect the volume of the *lake_ebb* Wave3D Item. She could do this by directly connecting the two, by dragging a connection line between the *source* of the Over All Motion property (the solid circle) to the *sink* of the Volume property (the hollow circle). However, such a direct mapping won't work well, as small but important motions in the scene won't be able to produce and sustain clearly audible sounds. Instead Kate drags a Threshold Item from the ToolBox tab onto the workspace, as shown in Figure 3.10a, which she will use to boost low level motion signals.

She connects the Over All Motion signal property to the threshold input port (Figure 3.10a), and then connects the threshold output port to the Volume property (Figure 3.10b). She then adjusts the threshold graph to boost the signal when it is at low levels (Figure 3.10c). The height and shape of the Threshold Graph determine how a signal's value is adjusted.

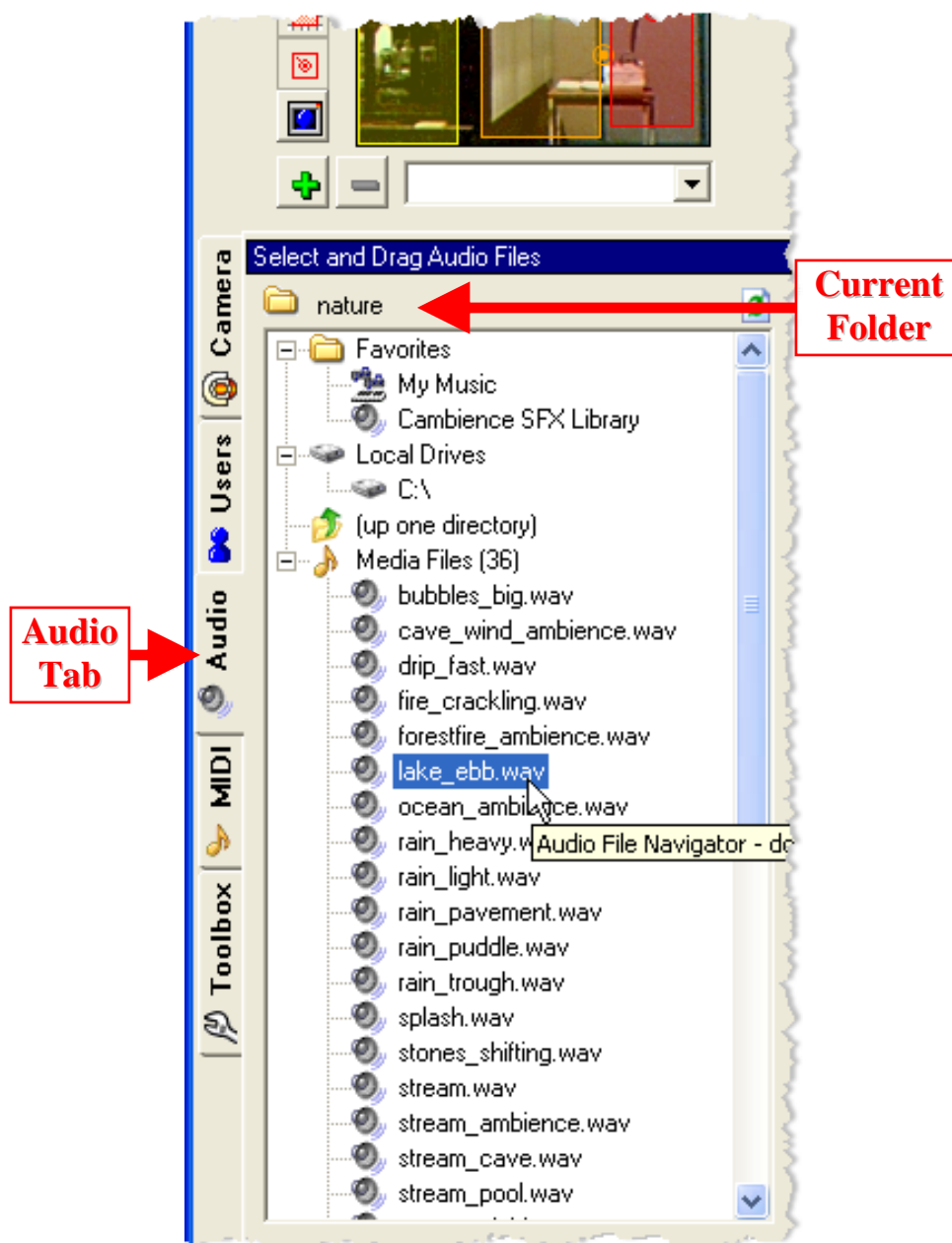


Figure 3.8: Audio Browser

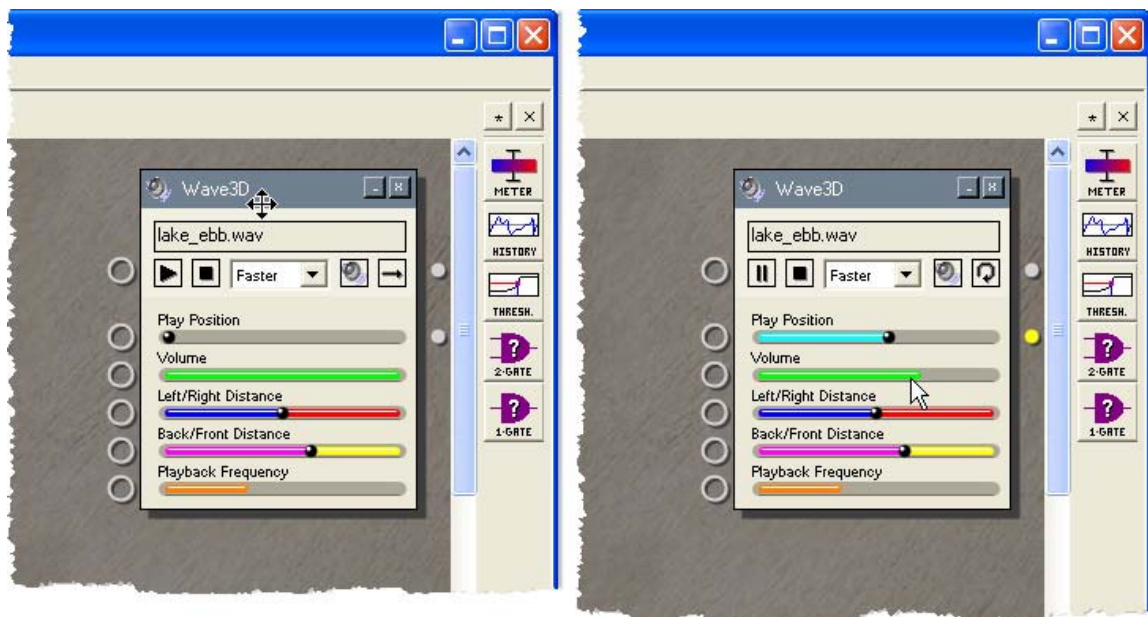


Figure 3.9: (a) A newly created Wave3D Item, (b) Adjusting the item.

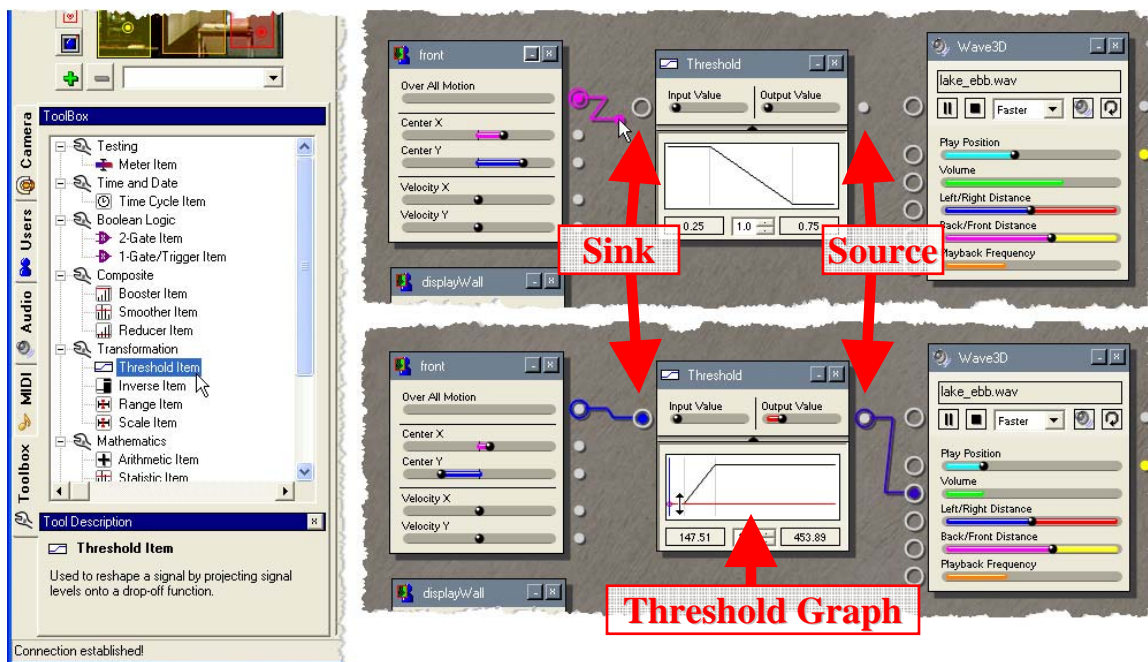


Figure 3.10: (a) The Toolbox panel, (b) Connecting the Threshold Item, (c) Adjusting the Threshold Item.

Using the same procedure, Kate can create a mapping to a second sound, this time using the *displayWall* region to drive it. Figure 3.11a and b show the mappings in action as someone enters the *front* and *displayWall* regions. Notice that she has used the Center X property of the *displayWall* region to drive the volume of *wind_gusting.wav*.

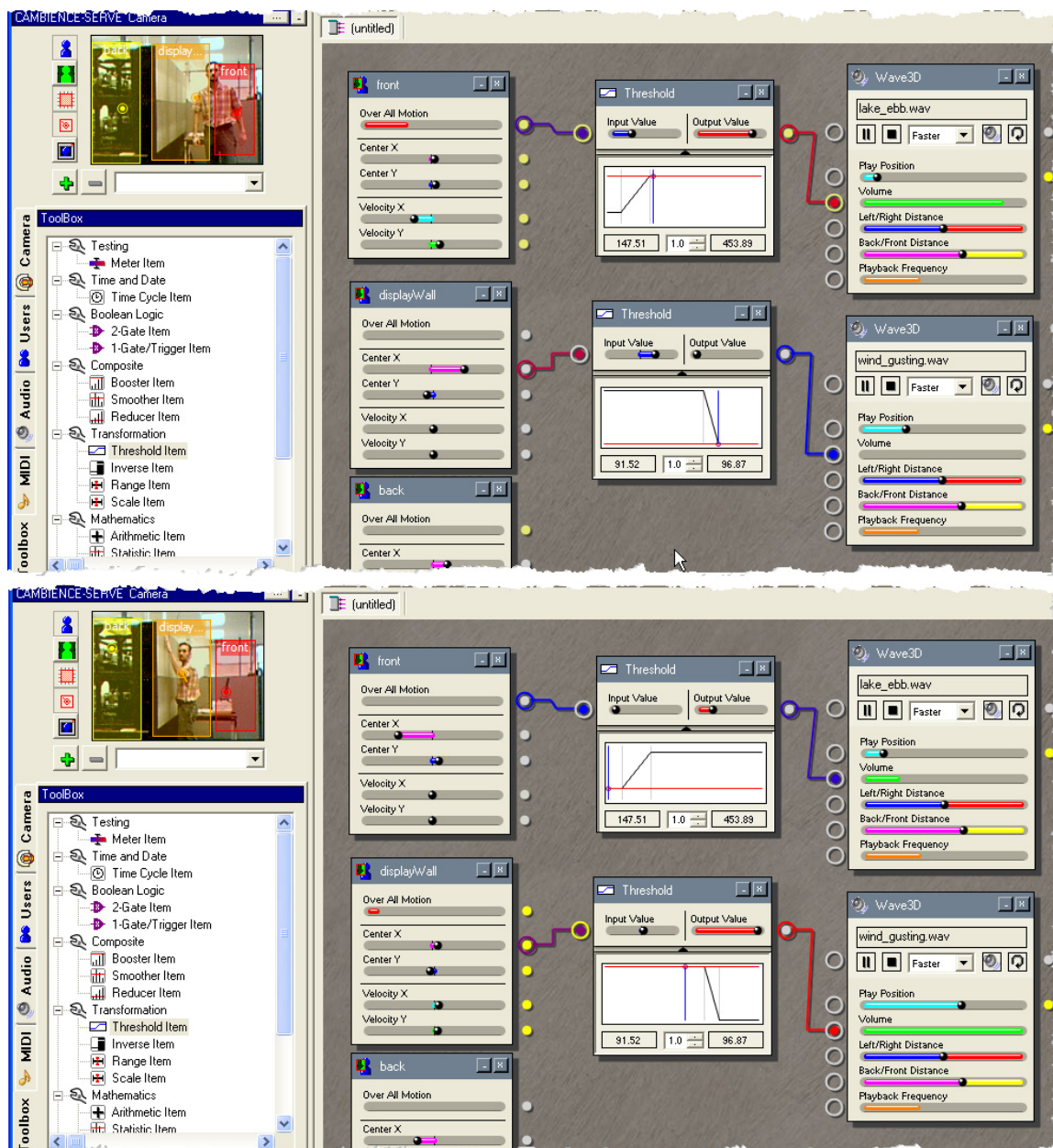


Figure 3.11: Mappings from the front (a) and displayWall (b) regions.

The sonic ecology is now complete. Movement in front of the display wall produces a gentle sound of water lapping against a shore; movements toward or away from the display produce wind gusting sounds; and motion behind the display wall affects yet another sound (not shown in Figure 3.11).

3.1.5 Step 5 – Sharing Mappings with Others

Kate’s final task is to make the sonic ecology that she just created available to anyone who connects to the server.

To do this, she first saves the workspace as “displaywall.eco”. Then she clicks the “Users” tab on the left edge of the sidebar. In the Ecology Share List near the bottom of the sidebar (labeled in Figure 3.12), she clicks the green plus sign to add a share. This brings up a dialog box (not shown), where she selects the filename of the workspace to share. The *displaywall* file now appears in the list, as shown in Figure 3.12.

Currently there is nobody connected, and only the local computer shows up in the User List (labeled in Figure 3.12). Later we will see what happens when others users connect to this server, and also look at how a connected Cambience user can retrieve this published sonic ecology.

The display wall machine need not generate the soundscape any longer. Kate closes the sonic ecology using the “x” button in the upper right corner of the Ecology Workspace panel (not shown). The display wall server will continue to silently transmit region information to anyone that connects to it.

3.2 Scenario 2

It’s now Josh’s turn to connect to the display wall server that Kate set up, and use the shared sonic ecology.

3.2.1 Step 1 – Physical Setup

Josh's first task is to set up Cambience on his own computer. As Kate did, he downloads and installs the software, and connects his webcam ("B" in Figure 3.1). He will configure his webcam regions later – for now he wants to monitor the display wall ecology that Kate set up.

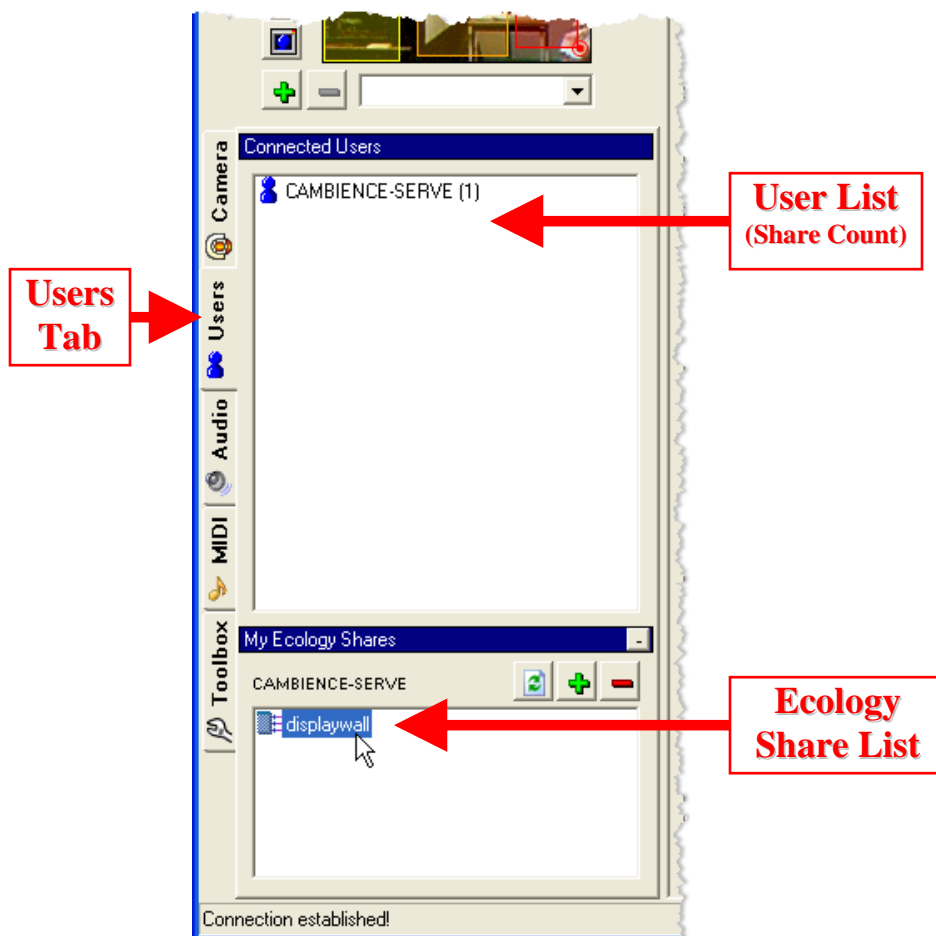


Figure 3.12: Sharing a workspace configuration (Ecology)

3.2.2 Step 2 – Connecting to the Server

When Josh first loads Cambience, using the Connection dialog box (Figure 3.13) he connects to the display wall server that Kate set up, by entering its IP address as supplied by Kate.

Upon connection, Josh's Remote Cameras/Regions List is updated to include those from the Cambience Server, as shown in Figure 3.14. Thus the display wall regions are showing, as well as Josh's Camera, which has a top-down view of his desk. Since Josh has not added any regions of his own, only his Camera node appears in the list. Although not shown here, the display wall server would have the exact same Remote Cameras/Regions list as Josh.

3.2.3 Step 3 – Loading an Ecology Share

In §3.1.5, Kate shared her *displaywall* ecology, and now Josh wishes to use it. Josh loads it by clicking on the Users tab and selecting CAMBIENCE-SERVE (Kate's Cambience Server) from the list (Figure 3.15). The number beside it is the total of shared ecologies available from this user. In the "Selected User" box below, the *displaywall* share is shown in the list of Ecology Shares.

Josh selects it and a copy of the display wall ecology appears in a new untitled tab in the Workspace, as seen in Figure 3.16. He could keep the display wall ecology as is, use the ecology as a starting point for his own customizations, or opt to create an ecology from scratch using the display wall regions.

Josh saves this ecology to his hard drive as a local copy. He can modify it without affecting the display wall machine's version (ie. he can personalize the use of the regions and sounds to his particular preferences). He can remove or reconnect the existing connections, or add new ones and adjust the graphs in the Threshold Items to meet his liking. If he drags a region from the Remote Region List, he can drop it on top of an existing Region Item to substitute the new region in its place. Similarly, he can substitute

sounds by dropping files from the Audio Browser on top of an existing Wave3D Item. However, the original version works just fine for him, so he leaves it as is.

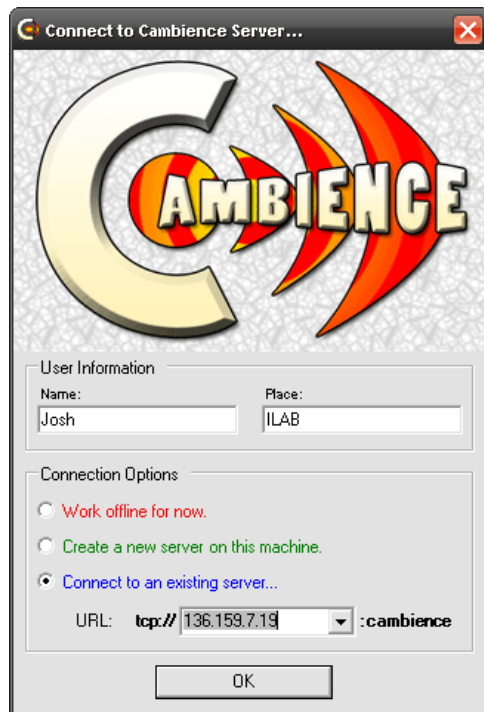


Figure 3.13: Connecting to the Cambience Server



Figure 3.14: Josh sees regions from the server when he connects.

At this point, Josh's machine is producing the same audio output that Kate designed on the display wall machine, described at the end of §3.1.4.

3.3 Scenario 3

Josh can now monitor the display wall as a soundscape. However to facilitate casual interaction between him and Kate, the two could also benefit by being aware of one another's presence and availability in their respective workplaces. As with the display wall, Cambience can be used in this situation to create such awareness.

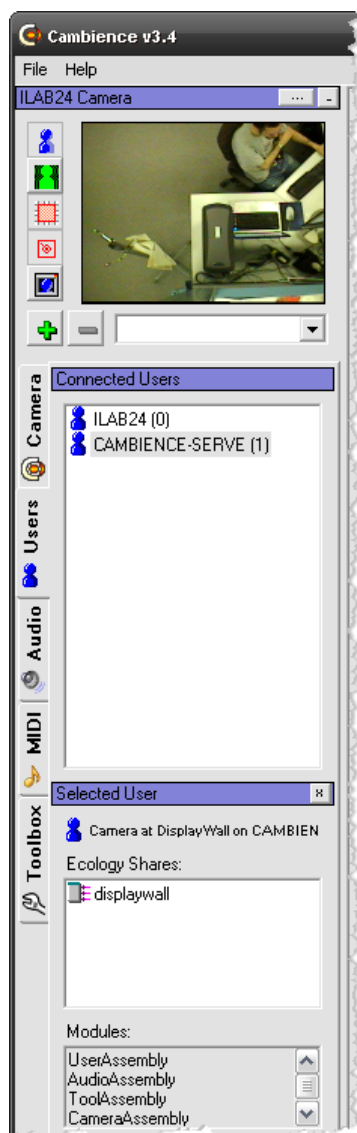


Figure 3.15: The User Tab can be used to view and load Ecology Shares.

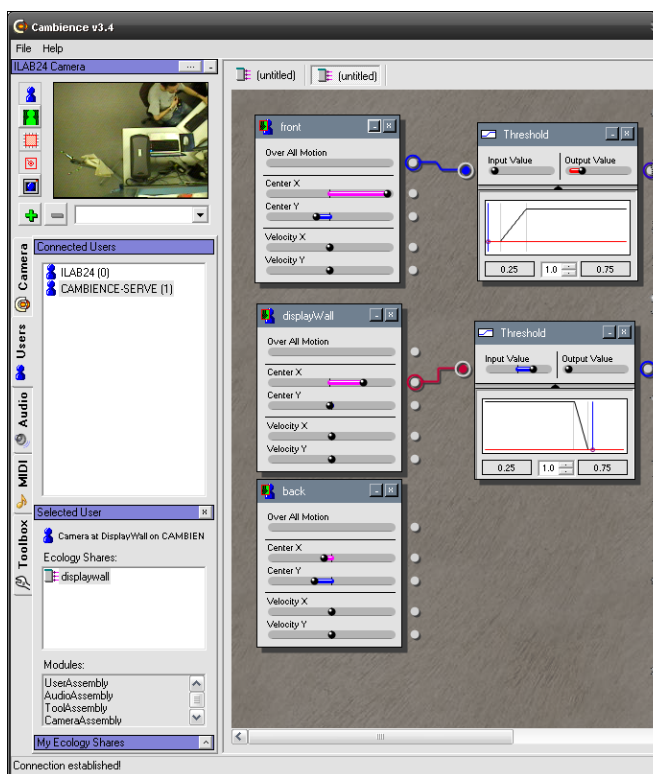


Figure 3.16: The *displaywall* share is loaded in a new "untitled" tab.

However, Kate is concerned with her privacy – she doesn’t mind people hearing sounds representing her activities, but doesn’t want others to see her video. In contrast, Josh doesn’t mind who sees him.

In this scenario, Kate will create a region for Josh to use that does not transmit video.

3.3.1 Step 1 – Configuring Regions in Private Space

Kate has also installed the Cambience software on the computer in her office, and has set up her webcam (“C” in Figure 3.1). To protect her privacy, she disables frame transmission before she connects to the Cambience Server. She clicks the “...” button above the Local Camera view, which brings up the Local Camera Settings window as shown in Figure 3.17.

She unchecks the “Transmit Camera Frames” option, which stops further live frames from being sent to other connected users, but Cambience still provides the last captured frame to others to reveal the relationships between her regions and the underlying scene.

Kate can verify that her live video is not being transmitted, by clicking on her camera in the Remote Cameras/Regions List and bringing up the Remote Camera Preview (not shown). The image is static, confirming that no frames are transmitted.

Kate can still make it possible for Josh to monitor her general office activity, even though her video frames are not transmitted. She decides to create one big region across her entire scene (Figure 3.18). By default, the newly created *kate* region, along with its measurement properties, become available to Josh. Kate doesn’t supply a mapping scheme for these measurements, leaving it up to Josh to create his own soundscape if he wishes to monitor her activities in the absence of video.



Figure 3.17: Kate disables camera frames from being transmitted.

3.3.2 Step 2 – Monitoring and Mapping Private Activities

Josh decides to create a simple sonic ecology around Kate's region. He is still monitoring the display wall sonic ecology that he loaded in Scenario 2. He clicks the '*' button at the top right corner of the workspace area to create a clean workspace (Figure 3.19). The display wall sonic ecology is no longer visible, but still produces sound until it is closed. Josh can easily switch back by clicking the "displaywall.eco" button above the workspace panel if he wants to.

Similar to what Kate did with the display wall, Josh creates a Region Item from the *kate* region, then brings in the *bird_piping* sound from the Audio Browser. He uses a threshold between the two items and creates connections such that Over All Motion affects Volume. Figure 3.20 shows the result of this mapping.

In addition to the sounds he hears from the display wall, he now hears the sound of a bird piping whenever anything moves in Kate's office. This change could be due to her shifting in her seat, turning the lights on or off, or even from a guest entering the office - but it is impossible for Josh to tell for sure. However, the rough time of the day can give him hints that Kate is arriving in the morning, popping out for lunch, or heading out at the end of the day. In this way, this movement information is still useful to Josh.



Figure 3.18: Kate's region covers her entire webcam view.

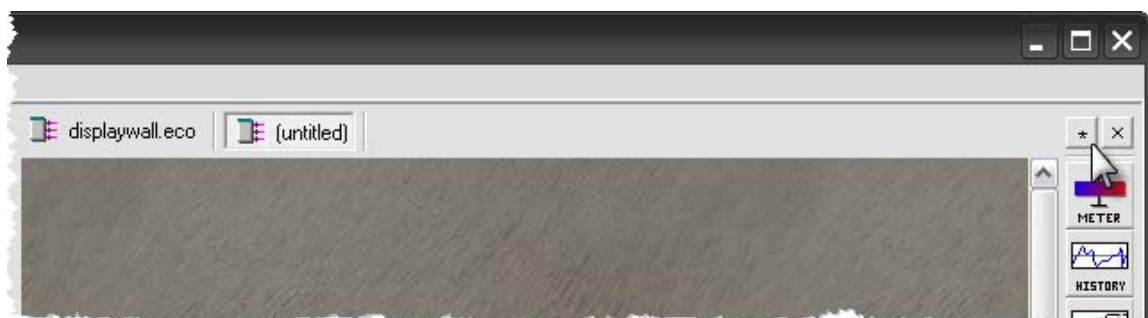


Figure 3.19: Creating a new ecology workspace.

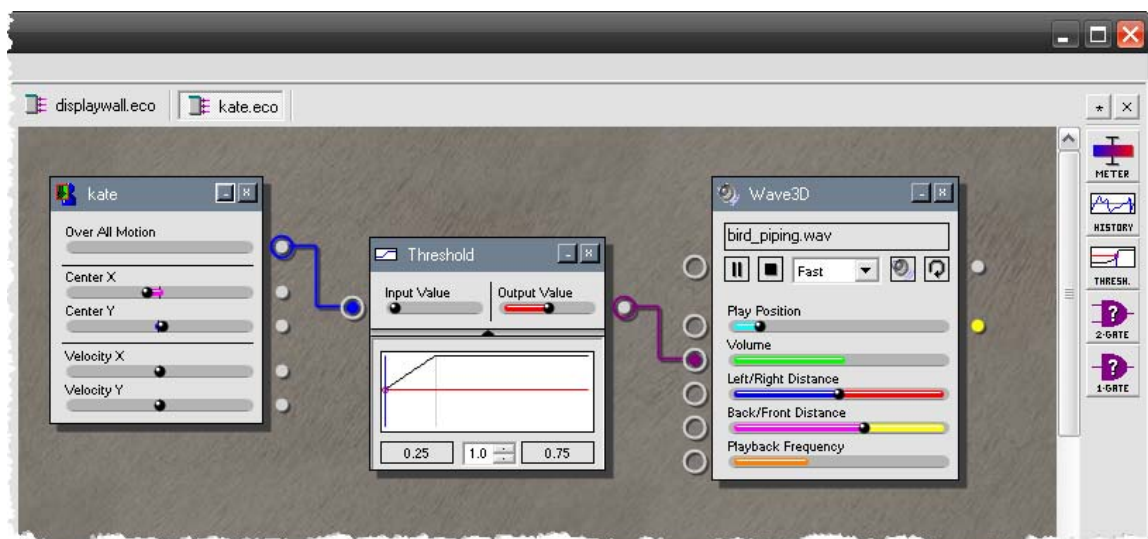


Figure 3.20: Two mapping schemes run in parallel.

3.4 Scenario 4

Finally, it is Kate's turn to monitor Josh.

3.4.1 Configuring Regions in Public Space

Josh wants Kate to be able to monitor his activities as well. In order to do this, Josh must create regions of interest on his own local webcam ("B" in Figure 3.1) which is mounted from the ceiling and pointing down at his desk area. Using a procedure similar to that described in Scenario 1, he ends up with the region configuration shown in Figure 3.21, but leaves it up to Kate to build an audio mapping.



Figure 3.21: Josh's webcam view partitioned into areas of possible interest.

3.4.2 Step 2 – Monitoring and Mapping Public Activities

Kate now designs the sonic ecology depicted in Figure 3.22 to monitor Josh's actions. The sonic ecology mapping is somewhat more sophisticated from prior examples:

- When Josh moves in his chair, the volume of *wind_gusting* will swell. Depending on the direction that Josh moves, the sound will pan left/right and front/back (assuming Kate has a 4-speaker sound set up).
- When Josh is typing on his keyboard or working at his desk, the *fishingrod_reeling* sound plays at full volume.

- When Josh goes near the coat rack to pick up or leave his coat, the *fishingrod_reeling* sound is stopped and the *fishingrod_running* sound plays at full volume. Once the motion calms down, *fishingrod_running* stops playing and *fishingrod_reeling* resumes.

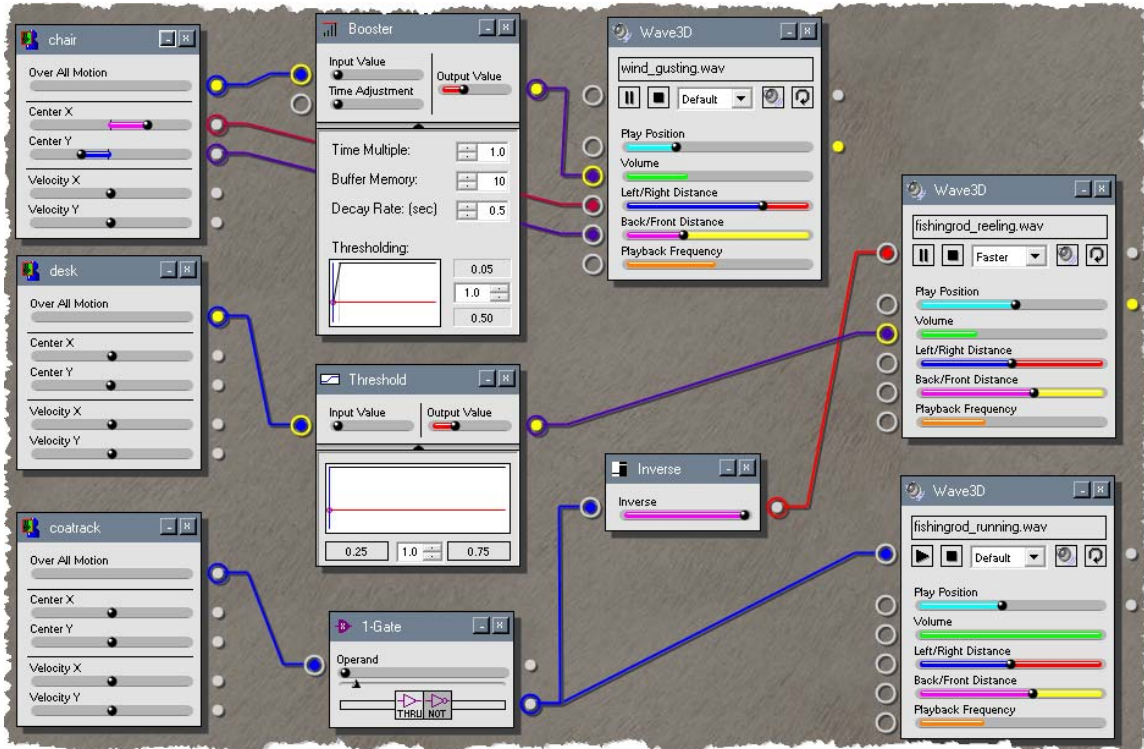


Figure 3.22: A sonic ecology to monitor Josh.

Various sound properties are manipulated: volume, left/right distance, and front/back distance. As well, various manipulation items permit this advanced use: a 1-gate, inverse, and a booster. These will be discussed in Chapter 4.

3.5 Scenario Summary

At this point, three separate computers are networked through Cambience, generating audio soundscapes based on the information gathered from one another's regions.

- The display wall computer is not producing a soundscape of its own – it’s main purpose is to collect region measurements for Josh and Kate to use, if they so choose.
- Kate’s computer is generating the soundscape we just described in the previous section, with the sound of wind and a fishing rod reeling and running to represent Josh’s actions within his workspace.
- Josh’s computer is generating a soundscape based on both the display wall and Kate. He hears the sound of water calmly lapping at a shore when someone approaches the display wall, and the sound of gusting wind when they are directly interacting with it. Additionally he hears the sound of birds piping when Kate moves in her office.

3.6 Conclusion

These scenarios illustrated the basic use of Cambience between two people, and how they crafted and exchanged sonic ecologies between the distance separated sites. In particular, we saw that Cambience does the following:

- **Cambience is a distributed video media space**, where each user specifies areas of their own personal video feed that others can monitor.
- It provides a Visual Programming Language, where **input from video can be mapped to sound or other output**. This enables people to manage their own notifications and desired level of awareness.
- It **monitors visual change in specific areas of personal video feeds**. The video regions each provide a number of different measurements that can produce different effects when connected to the playback properties of one or more sounds.
- It **reduces the need for a person to watch live video**, or to even have the video presented on their display (however these are both still possible).

- It uses input from **multiple video feeds** in the design of a **single local soundscape**.
- It **transitions awareness data** smoothly from background sound to foreground video.

The following chapters will explain some of these properties in greater detail, and will reveal other capabilities and mapping possibilities.

Chapter 4. Shared Media and Live Visual Programming

The scenario in the previous chapter illustrated various concepts of Cambience, as listed in §3.6. Of these, two are particularly important, and will become the focus of this chapter. First, Cambience is a distributed media space that allows great flexibility in how people can share their camera frames, their regions, and their sonic ecologies with others. Second, Cambience is a visual programming language where end-users can map video input to sound or other output.

This chapter revisits these two concepts from a more technical standpoint. We explain them in better detail (including related features not shown in the scenario), and describe the underlying implementation to demonstrate how these ideas were realized. This chapter will also replace the layman's terms employed in previous chapters with more precise terminology. Chapters later will dive into further details, e.g., how Cambience obtains its visual input, and what the possibilities are for audio output.

4.1 Sharing Media

In the last chapter we described a scenario where two distance-separated collaborators mutually used Cambience to construct a distributed media space and sonic ecology. This then allowed them to monitor a common resource (the display wall) and each other. As part of this process, each was able to share with one another things like their camera frames, their regions, and their sonic ecologies. This level of sharing sets Cambience apart from other systems of its ilk. Although Cambience, like other systems, can be used to generate a stand-alone sonic ecology built around only the local scene, Cambience's ability to share different levels of information raises many interesting and rich design possibilities in the creation and deployment of distributed media spaces.

4.1.1 Shared Data

The following information can be easily shared between Cambience users. This information can be leveraged by people when building a new media space, or can become a regular part of the system during day-to-day use.

Basic Information of Participants includes the person's name, location, and computer name. This helps other people identify who has generated particular data, where cameras are located, and so on. Of course, this helps not only technically but also socially, as participants can identify one another and talk about how they plan to use the information. The user name and place can be set when first creating or connecting to a Cambience Server (Figure 3.13), and are used to label information in the Remote Cameras/Regions list and the Camera Preview panel (Figure 3.5b).

Captured Video Frames from the camera connected to one particular computer are available to all. However, the camera's "owner" can decide on the frame transmission rate relative to their camera's actual frame capture rate. For example, they can transmit full video at (say) 10fps, or "snapshot video" at 1 frame every minute similar to Portholes (Dourish and Bly, 1992). The region measurements are transmitted every time a frame is captured, independent of the frame transmission rate, so region measurement fidelity is assured. The owner can also disable video transmission completely, so that no other person can see their live video. In this case, only the last transmitted frame remains as a static "representative frame", which lets other participants understand the visual context of the regions in that scene.

This flexibility trades off privacy for power. For example, Kate decided not to transmit video of her office to protect her privacy in §3.3, but Josh could still hear the soundscape that was generated from this video. Josh was less concerned about privacy, and allowed video transmission. This allowed Kate to view Josh's underlying video to see how his regions were used to divide the scene, and helped her to understand what activities drove which parts of the ecology.

Region Information includes the region bounds, color, name, and description. These details allow for the regions to be shown overlaying the respective remote camera preview, making it easier for users to get ideas about possible uses for these regions in their own sonic ecology. The continually updated region metrics are accessible to any user by creating a corresponding Region Item in their Ecology Workspace (Figure 3.6). We already saw in §3.2 how Kate and Josh were able to create their own Sonic Ecologies using regions that other people had created on other machines.

Sonic Ecology Shares are the final results of a design that maps video region properties to sound properties. When a design is shared, others can preview and optionally discard it, or save it as a local copy that can be used as-is, or modified further. We should mention that the underlying system does not actually transmit or share audio; rather, the ecology configuration information that is used to generate the audio is transmitted as an XML file. The XML file contains references to any audio files used from the Cambience SFX Library (§5.2.8), which are assumed to be common to all users. However, if custom audio files are used in the design, the user who loads the share is queried for replacement files. If none are provided, the sound item is removed from the sonic ecology configuration altogether. Sharing of actual audio files for sonic ecologies is not supported by Cambience, although this could be easily added. For now we expect people will exchange audio files through other channels as needed (e.g., E-mail, FTP, etc.).

4.1.2 Cambience as a Distributed System

The rich level of sharing described above is possible because Cambience is implemented as a live distributed system, where rich multimedia data are shared between clients. We briefly describe how this is done.

Cambience's distributed architecture is based on a distributed Model View Controller pattern (Greenberg and Roseman, 1999).

- The **Model** is the shared multimedia data store that triggers notifications whenever any changes occur.


- The **Controller** is the video, the resulting video metrics, and any other program inputs that are captured and published to the shared data store.
- The **View** is the Cambience interface: all of the components that display information about the available shared data, and the resulting visual and audio output from a sonic ecology mapping.

Under the covers, Cambience implements the dMVC through a third-party library and client/server runtime architecture called .Networking (“dot networking”). A previous version of this library (called the Grouplab Collabrary) is discussed by Boyle and Greenberg (2005). The relevant .Networking capabilities are listed below.

- It is a Notification Server, where clients can publish and/or subscribe to data. When data is added, modified, or removed by a publisher, all other clients subscribed to that data receive a notification of that action as an event (Patterson, Day and Kucan, 1996).
- It provides a high-level distributed data structure called a *Shared Dictionary*. This dictionary resembles a distributed Hash Table with key/value pairs. Clients can publish or subscribe to these entries. Notifications specify the action performed on a particular entry, as well as the entry value itself (so that it can be examined and the view can be updated).
- The Shared Dictionary works as a hierarchical data structure. Keys are based on a ‘/’ delimited hierarchy. Pattern matching actions can be performed on any part of a hierarchy, e.g., so that people can quickly subscribe to entire sub-trees, or so that they can retrieve particular parts of the tree as specified by a regular expression.
- The Shared Dictionary stores all of its contents in a central server that others can connect to as clients. The server ensures that the key/value hierarchy remains consistent among all connected instances of the Shared Dictionary. For network efficiency, only data that matches a client’s subscription pattern is actually transmitted to the client. Because the Shared Dictionary stores its contents (rather

than only broadcasting updates), latecomer clients have access to the current values of all keys previously entered into its distributed data structure, and can use this data to construct an up-to-date view.

Table 4.1 below explains the structure of the keys under which Cambience stores data in the Shared Dictionary. The table identifies the purpose of each key from the scenario summarized in §3.5, and the data found as the value of each one. Here we see the three connected computers: Josh on *josh-comp*, Kate on *kate-comp*, and the display wall computer on *cambience-serve*. Each computer has a key for its user information (line 1-3) and for its captured video frames (lines 4-6). We can see that *cambience-serve* is the only computer with an ecology share (line 15), and we can count the number of regions belonging to each computer. As expected Josh has 4 regions (lines 8-11), Kate has 1 (line 7), and the display wall has 3 (lines 12-14).

#	Shared Dictionary Key	Stored Data
Basic Information of Participants		
1	/cambience/user/josh-comp	Username="Josh" Location="ILAB", ComputerName="josh-comp" ...
2	/cambience/user/kate-comp	Username="Kate" ...
3	/cambience/user/cambience-serve	Username="Server" ...
Captured Video Frames		
4	/cambience/camera/josh-comp	(Bitmap) 
5	/cambience/camera/kate-comp	(Bitmap) ...
6	/cambience/camera/cambience-serve	(Bitmap) ...
Region Information		
7	/cambience/camera/kate-comp/region/(GUID-A)	Name="kate" Bounds="0, 0, 160, 120" Color="Red"

		Description="Kate's Office", ActivatedPixels=0 CenterX=80 CenterY=60 ...
8	/cambience/camera/josh-comp/region/(GUID-B)	Name="coatRack" ...
9	/cambience/camera/josh-comp/region/(GUID-C)	Name="chair" ...
10	/cambience/camera/josh-comp/region/(GUID-D)	Name="floor" ...
11	/cambience/camera/josh-comp/region/(GUID-E)	Name="desk" ...
12	/cambience/camera/cambience-serve/region/(GUID-F)	Name="front" ...
13	/cambience/camera/ cambience-serve /region/(GUID-G)	Name="displayWall" ...
14	/cambience/camera/ cambience-serve /region/(GUID-H)	Name="back"
Sonic Ecology Shares		
15	/cambience/ecology/cambience-serve/displaywall	(XML)

Table 4.1: Key-value structures in Cambience's shared data.

4.2 Visual Programming

In Chapter 2 we reviewed a number of systems that can use visual input to generate or control audio output – a technique known as sonification. Some of these systems (§2.1.4, §2.1.5) had hard-coded algorithms that determined the effect on the output (Very Nervous System, AROMA, Audio Aura, ARKola, ambientRoom), while others (§2.3.1) allowed users to create their own complex mappings through a visual programming language (Isadora, Eyesweb, MaxMSP + Jitter). Hard coded mappings may be easy to use out-of-the-box, however the programmer's conception of a good mapping may not meet the user's needs, and in fact restrict their freedom to explore other possibilities. Visual programming languages offer more freedom, but sacrifice ease of use and require more set-up time.

As an aside, I stress that none of the visual programming languages mentioned in Chapter 2 were created for the sole purpose of live sonification. The result is that, unlike in Cambience, users must navigate through a bloated list of features and carefully compose complex schemes of programming elements to accomplish video sonification.

Conventionally, computer programmers define the behavior of a program by writing a body of source code and compiling it into an executable program. In Cambience, the visual programming environment (VPE) lets us define program behavior by arranging visual elements (representing objects and data flow) into a desirable configuration that can be executed by the system. As previously defined, the repertoire of objects and elements that make up a set of possible configurations are referred to as the *visual programming language* (VPL). In this case, the advantage of the visual programming language is that it was designed to hide complexity and details that non-programmers may not be equipped to deal with.

By creating an intuitive visual language expressly for distributed video sonification, people who use Cambience can define program behaviour without the need for IDEs, compilers, and coding knowledge. Cambience's VPL offers a great deal of customizability and flexibility by allowing people to choose what videos - or portions of videos – they wish to monitor, what sounds or actions are used to represent change, and how those sounds and actions respond to different types and magnitudes of change. Elements can be easily arranged and connected during program execution, making the resulting behaviors instantly available for interaction and modification. People can test and revise their configurations through trial-and-error, save and load these configurations at will, and share them with other Cambience users.

Figure 4.1 revisits Cambience's main program window, with four of the main interface areas highlighted:

- The *Visual Programming Workspace* on the right is an area for users to arrange programming elements to produce the desired video-to-audio mapping.
- The *Local Camera Panel* shows the video captured by the locally connected webcam, and allows users to define regions and preview detected change.
- Cambience acts as a host for program modules that are loaded at runtime (e.g. Camera Module, Audio Module, User Module, Tools Module), which is discussed in further detail in §5.4. Each module plugs into the *Control Panel Area*, giving

an interface to interact with it, and disseminate its specialized items into the workspace.

- The *Quick Toolbar* is merely an area to quickly access a number of Toolbox items that we deem to be highly used: the Meter Item, History Item, Threshold Item, and the 2-Gate and 1-Gate Items.



Figure 4.1: Cambience Main Program Window

In the next 3 sections I discuss the basic structures common to most VPLs, which also compose Cambience's own VPL. In later sections I will get into specifics about the types and functions of these language building blocks.

4.2.1 Items, Attributes, and Functions

An item is the programming equivalent of an object, its attributes are akin to the object's properties, and functions equate to the methods that affect those properties. Indeed items give access to program input, control program output, and perform tasks in between. For example, in §3.1.3 and §3.1.4 we saw how Josh created and manipulated various items that represented camera regions, audio files, and signal transformations.

Items appear as boxes within the visual programming workspace. The user can create items by dragging desired audio/video/utility objects into the workspace. I go into a more detailed explanation of specific types of items in later sections of this chapter.

Items visually display most of their attributes, and offer ways to interact with the underlying objects. In most cases, attributes and their values are visualized by a linear bar meter on an item. As seen in Figure 4.2, the colored bars represent the current value of each attribute within the possible range of values. These attributes can be changed by clicking on a new position within the meter's range, except in some items where:

- Attributes are permanently disabled since it makes no sense to allow user control (e.g. Region Items: Over All Motion, Center X, Center Y, etc.).
- Attributes are disabled when being controlled by an incoming connection (e.g. Threshold Item: Input Value).

Other specialized controls can appear within items to offer other interaction possibilities outside of their attributes. For instance, the Wave3D Item in Figure 4.2 has buttons to control audio playback, and a drop-down to select the rate of change for volume/pan/fade transitions. Meanwhile the Threshold Item has controls to define the Output Value based on the Input Value.

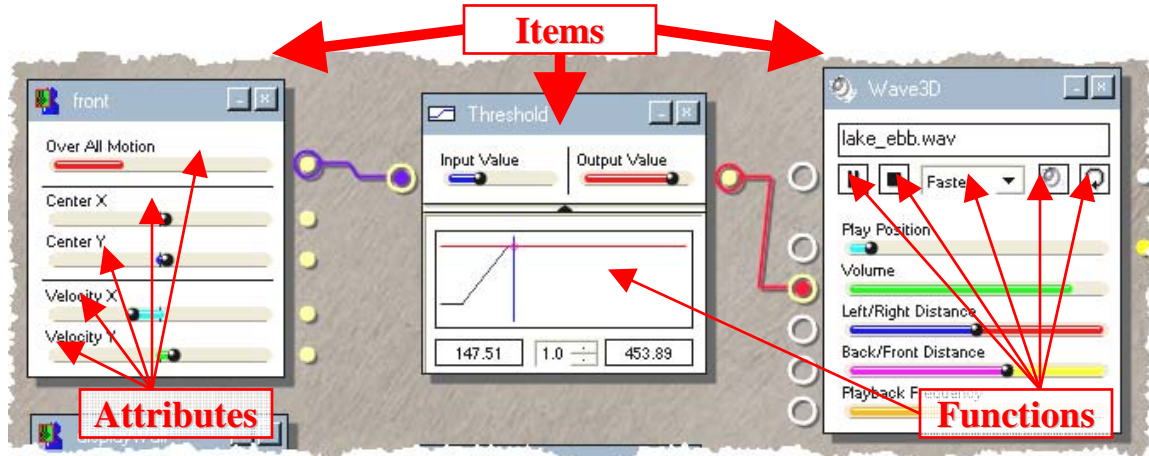


Figure 4.2: Examples of Items, attributes, and functions

4.2.2 Sources, Sinks, and Patches

Items have a mechanism for transporting data into and out of their attributes. The values of attributes are pushed out through a *source*, which appears as a small solid circle beside the item's attribute on the right-hand side of the item. Similarly, data is pushed into an attribute through a *sink*, which appears as an empty circle on the left side of the item's attribute. With sources on the right and sinks on the left, data flows from left to right.

By creating a *patch* (informally referred to as a “connection”) between a source and a sink, we can establish a path of data flow. Patches are created simply by clicking and dragging a line between a source and a sink. Figure 4.3 shows two patches: one from the Over All Motion source of the Region Item to the Input Value sink of the Threshold Item; the other from the Output Value source of the Threshold Item to the Volume sink of the Wave3D Item.

Sinks can only accept one patch input, but multiple patches can emanate from a source. The reason for this restriction is that two or more different signals entering a sink could cause unpredictable behavior on the attribute. However, Cambience provides logic gate

items which can be used to combine 2 signals, or chained to combine more. I discuss these items further in §5.3.3.

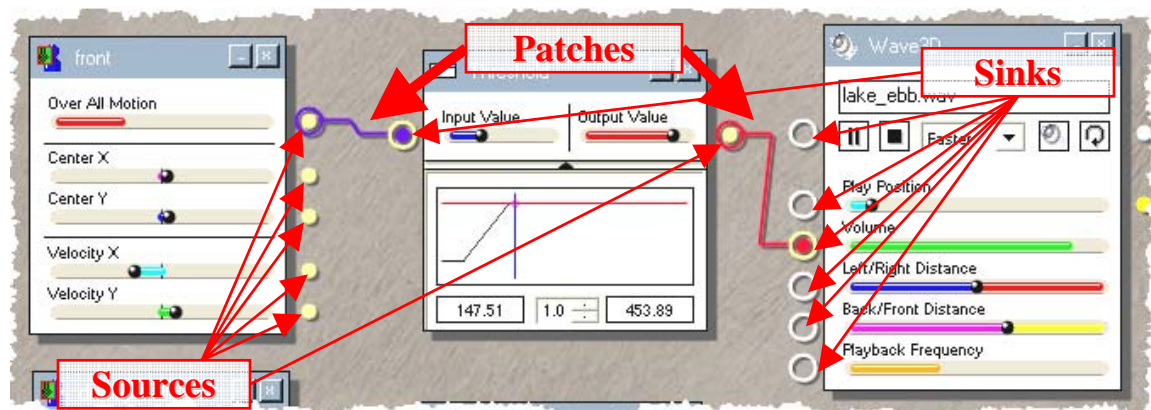


Figure 4.3: Examples of sources, sinks, and connections.

4.2.3 Signals

A *signal* is the term I use to describe the data contents of an attribute as it is transported over a connection. Attributes may change for various reasons: according to input, or from user interaction with the item interface, or from signals that the attribute receives through a sink.

When a connection is first created, an initial signal called a “primer signal” is automatically fired. This ensures that the connected source attribute and sink attribute are immediately consistent; otherwise the sink attribute would retain its previous value until the next signal travels over the connection.

Signals are discrete events that only occur when an attribute changes. Changes in Region Item attributes are driven by the camera polling interval (i.e. whenever a new frame is captured). Otherwise, non-input items are designed to update their internal states and visual displays synchronously, upon receiving a signal or manual input.

All information transmitted as a signal must be adapted to a bounded continuous value. Thus maximum and minimum values are grouped with the actual value of the attribute, with additional origin and default values. Table 4.2 describes the purpose of each signal component.

Signal components depend on the nature of the attributes that generate them, and must be adapted to fit the attributes that they affect. Figure 4.4 isolates attributes from familiar items, and reveals the inherent bounds of their values. For instance:

- Programmatically, a sound's Volume is defined by a value between -10,000 and 0, where 0 is full volume (Figure 4.4c).
- The left/right pan of a sound is specified between -10,000 and 10,000, where 0 is exact center (Figure 4.4d).
- The play position of a sound depends on the sound's temporal length. Thus, the maximum value can vary, but is determined once the sound is loaded (Figure 4.4b).
- The minimum, maximum, origin, and default value of the Center X attribute can vary depending on the bounds of the parent region. These values can change at any time when a user resizes or moves the region (Figure 4.4a).

Component	Purpose
Minimum	The lower bound of the possible range of values for this signal.
Maximum	The upper bound of the possible range of values for this signal.
Actual Value	The current value of the attribute, between the minimum and maximum.
Default Value	An initial value for the signal, between the minimum and maximum.
Origin	Specifies a pivot point for the signal visualization. This value is usually 0.

Table 4.2: Signal Components

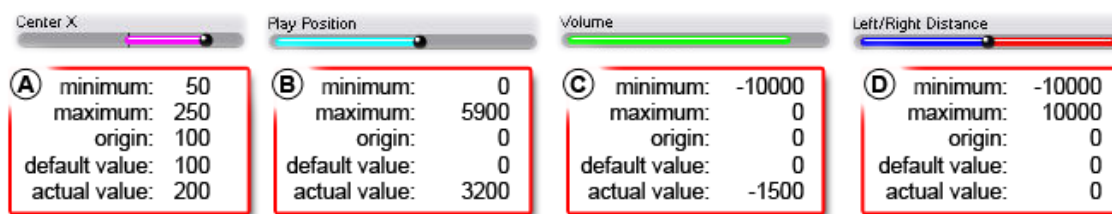


Figure 4.4: Attributes and Signal Components

This signal structure is necessary to ensure compatibility between attributes with different ranges. For instance, a Region Item's Center X attribute (Figure 4.4a) has a completely different range to that of the Wave3D Item's Volume (Figure 4.4c). No matter what the actual value of the signal, we can rescale and translate to an equivalent value that fits the range of the receiving attribute. If a Center X value of 125 is signaled to the volume of a Wave3D Item, Cambience will translate this to the equivalent value of -2,500 within the volume attribute's range.

Not all items necessarily apply this rescaling. Some items (mainly those intended to perform mathematical functions) operate on the actual values, producing a result within the original signal's range. Others may even affect the maximum, minimum, and origin values of the signal.

Boolean functions are implemented by evaluating False if the signal value is equal to its origin, and True otherwise. A prime example of this is the topmost sink on the Wave3D Item (seen in Figure 4.3), which uses a Boolean evaluation to turn audio playback on and off.

A number of simple visualizations are employed to help the user understand how and when signaling occurs as the VPL runs (Figure 4.5). In the programming workspace, the hook and socket endings of a connection flash yellow when a signal occurs. In addition, the color of the connection changes to indicate the value that was last signaled. Values close to the signal minimum cause the connection to be bluer, while signals close to the

maximum cause the connection to be redder. Values in between appear as a shade of purple.

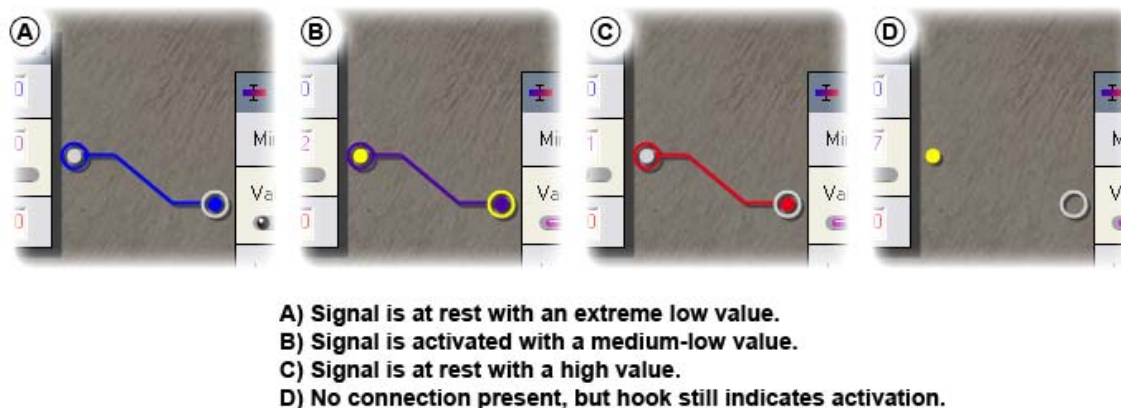


Figure 4.5: Signal visualizations - level and activation.

4.2.4 Live Programming

Most conventional programming languages use a compile-and-run approach, where program flow must be halted to make modifications to its behavior, and all modifications must be validated for errors before it may start again.

A powerful aspect of Cambience is that all programming is done live, while it is being executed. The advantage of this approach is that users may immediately experience the outcome of the modifications they make to their “program”. This greatly speeds up the process of fine-tuning sounds and mappings.

Cambience is also designed to have a closed grammar, so syntax that could cause an “error” is simply not possible. The only form of validation that Cambience performs (in real time) is for infinite signal loops, which occur when a path is created such that a signal flows back into the originating attribute (Figure 4.6). In this case, the patch that closes the loop is destroyed upon the arrival of the next signal.

One disadvantage of live programming arises when it comes time to “debug” the behavior of a mapping. Currently, Cambience has no facility for simulating input data, which often requires the user to do so manually (i.e. by moving in front of the cameras). In many cases this is awkward because one must move away from their computer display and speakers, or they are building a mapping on a remote camera view and have no control over what is happening in the scene.

There is really no solution other than to wait for an event to occur within the region, or ask for the assistance of another person. However, I included the option for an enlarged local camera view window so that the otherwise tiny display can be seen at a distance (Figure 4.7).

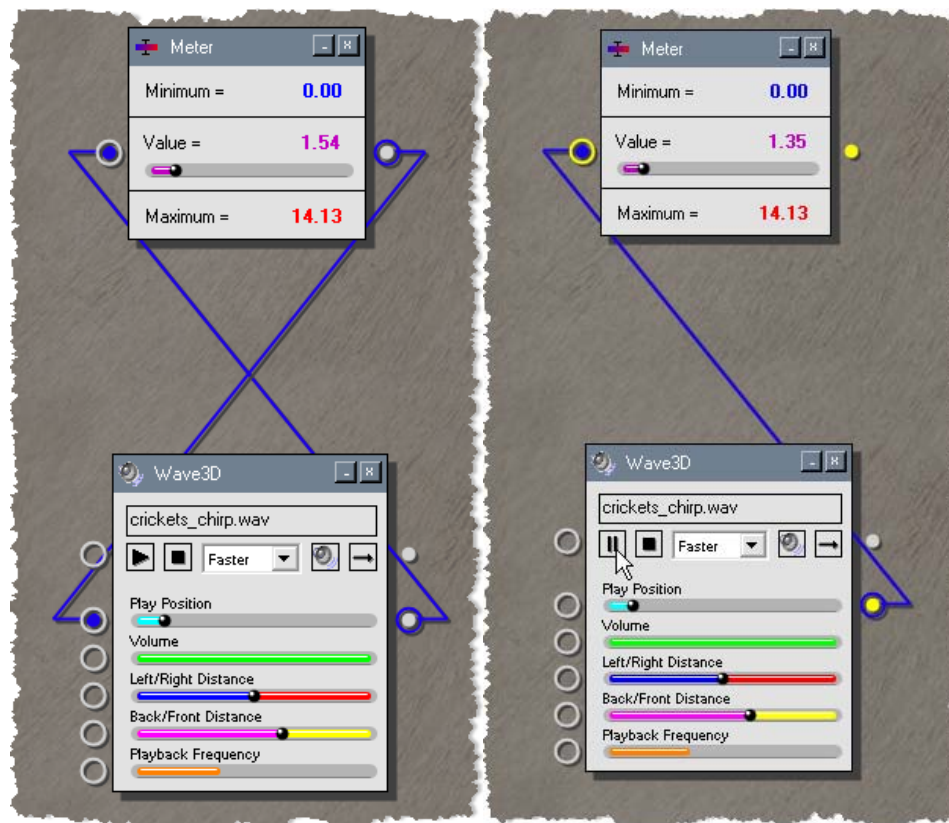


Figure 4.6: Infinite signal loop (left) broken on arrival of next signal (right).

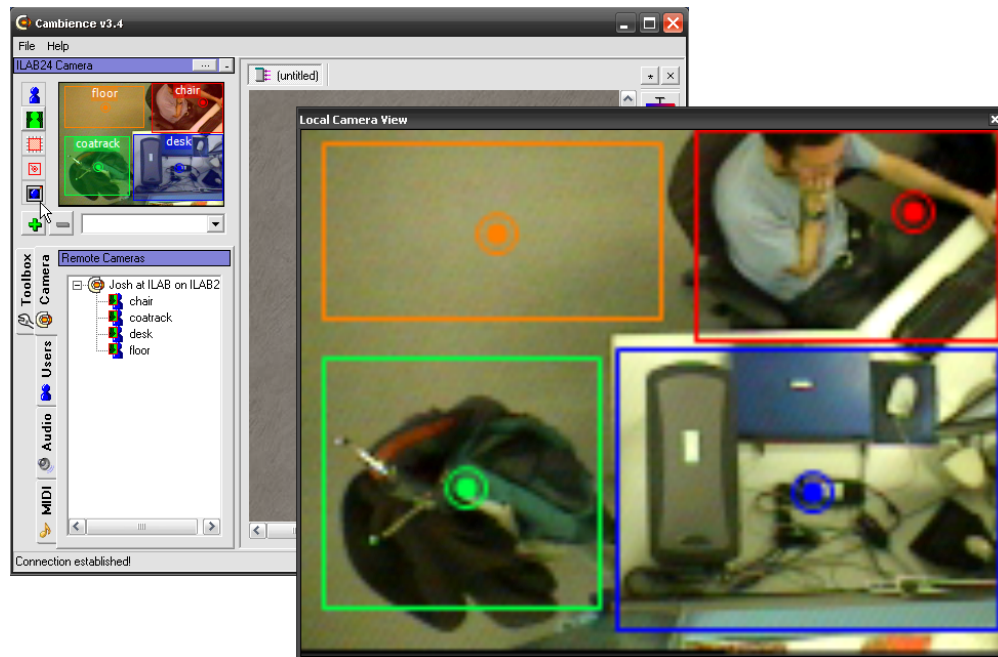


Figure 4.7: The Local Camera View window can be expanded to any size.

4.3 Summary

In this chapter:

- I described how Cambience functions as a distributed system.
- I revisited the discussion of visual programming environments, and explained the motivations behind the design of Cambience's visual programming language.
- I formally introduced items, attributes, functions, sources, sinks, and patches, and explained the role and behavior of each one.
- I explained how signals transport attribute values across patches, thus defining data flow.
- I highlighted Cambience's live functioning, and explained some relevant advantages and disadvantages.

Chapter 5. Mapping Video to Audio

In this chapter I dive into the specifics of Cambience’s visual programming language. In particular I detail these three key item classes, some of which were briefly mentioned in the previous chapter:

- **Visual Items** represent and control attributes of video regions (program input).
- **Sound Items** represent and control attributes of the audio (program output).
- **Signal Items** further control, combine and reshape signals.

5.1 Visual Items

5.1.1 Analyzing Regions

In Chapter 3, we already saw how end-users of Cambience use their webcams as the primary source of input, where video scenes are decomposed into named regions. Each region acts as a kind of ‘sensor’, where the visual properties within a single region or across multiple image regions can be analyzed to generate some kind of signal.

There are many ways that video regions can be analyzed, and Cambience just touches upon a few methods. Since awareness systems motivated my initial development of Cambience, I considered the importance of letting people know about changes in the regions of a scene: if nothing changes, awareness notifications of the status quo are likely pointless and distracting.

The question then was: how can we analyze a region in the scene for changes, and how can these be transformed into useful signal ‘attributes’? The answer to this lies, of course, in Computer Vision. Indeed, while there have been many advances in Computer Vision, recognizing objects within a scene and how they change over time is still an

active research area. However, my goal was not to advance the state of the art in Computer Vision, but to explore aspects of the Cambience interface. At the time of designing Cambience, I had little knowledge of Computer Vision, thus I created an overly simple but still meaningful change detection algorithm purely to get the job done; this will be described shortly. This implementation serves as a placeholder for other more sophisticated and efficient techniques as new generations of Cambience are created, e.g., Optical Flow (Beauchemin and Barron, 1995), 3D Object Recognition (Besl and Jain, 1985), Blob Detection (Rodriguez and Shah, 2007), and so on. Indeed, Cambience is constructed so that swapping analysis modules in and out is fairly straightforward.

5.1.2 Change Detection through Frame Differencing

The change detection that I implemented was based on the technique of frame differencing, as seen in Figure 5.1. Later research led me to discover that my technique is similar to the motion analysis mode of Very Nervous System (Rokeby, 1986-1990) which, in its mid-generation, used a special purpose SCSI device to monitor regions and produce signals from one or two cameras.

Frame differencing employs image subtraction, a pixel-wise operation often used to compare two images of the same size, which produces a third image illustrating how they differ. The resulting difference image has black ($R=0$, $G=0$, $B=0$) in unchanged areas, and has pixels of brighter color in areas with a more dramatic change in light. As illustrated in Figure 5.1 (top), Cambience stores a *Reference Frame* and subtracts it from the most recently acquired frame (*Captured Frame*) to produce the *Difference Frame*. The figure illustrates how the Difference Frame shows the changed pixels of a person moving through the otherwise static background.

While this image visualizes change, we need a single numeric value to form our signal. To generate this, the sum of the RGB values of each pixel from the difference image is compared to a user-specifiable threshold value (*Pixel Noise Threshold* in the Cambience

Settings Panel in Figure 5.2) in order to filter out camera noise, minor lighting changes and other inconsequential differences. Pixels that pass the threshold are marked as change while those that do not are marked as blank, producing the *Binary Frame* (middle of Figure 5.1). The change-activated pixels in the *Binary Frame* are then summed within the bounds of each region from the *Region Definition* to produce a ratio of change-activated area to the total area of the region. This is the *Over All Motion* metric seen as an attribute on Region Items, and previously discussed in §3.1.3.

Other settings in the Cambience Settings Panel (Figure 5.2) let us experiment with different parameters of this algorithm. For example, the Scan Granularity setting dictates whether every single pixel in the region is counted (value of 1) or if every 2nd pixel is counted (value of 2) etc. If no regions are being published from the given computer, change detection can even be disabled to reduce the burden on the computer's processor.

One question is what comprises the reference frame. If it was taken long ago (e.g. at startup), lighting changes or object movement will eventually result in a large different ratio that does not reflect current human activity. If taken a short time ago (e.g. the previous frame) the ratio will be highly sensitive to frame rate. Our strategy is to incrementally update the reference frame by applying the *Captured Frame* as an opaque overlay. The Frame Blend Ratio setting in Figure 5.2 dictates the strength of the blend, and thus affects the speed of the update.

We should mention that the overall motion signal is just an estimate suggested by differencing. The algorithm is affected by things like clothing texture. As a person moves through the camera scene, Cambience typically sees a crest and wake of motion: a crest where their bodies overlap a new area of the background in the direction that they are moving, and a wake where their bodies uncover an area of the background. Cambience typically doesn't notice motion within areas of uniform color, such as a plain shirt or a patch of skin. This is because there is little color difference when comparing two different points from the same color patch, between two subsequent frames.

However textured clothing (e.g. Plaid) will produce a larger difference value. Similarly, turning lights on and off will produce a large Over All Change signal.

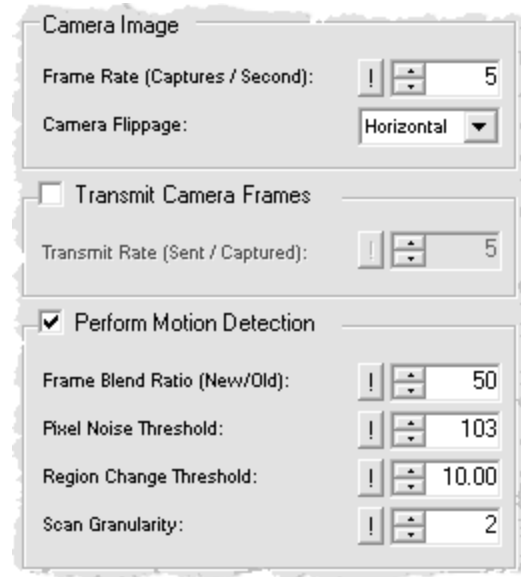


Figure 5.2: Cambience Webcam Capture Settings

5.1.3 Other Change Metrics

Cambience includes 2 other crude change metrics. First, when the change-activated pixels are being summed, an average position is calculated. This is done by summing the X and Y coordinates of each change activated pixel from the *Binary Frame* (middle of Figure 5.1) within the region bounds, and then dividing those two sums by the total number of change activated pixels. This usually determines a point that follows the center of a moving object within a region, which I call the *Center of Change*. The *Region Change Threshold* setting in Figure 5.2 dictates the percentage of *Over All Change* that the region must attain before the *Center of Change* is updated. This is necessary because otherwise the center would bounce around to follow every speck of noise, rather than maintaining the position of the last significant motion.

Second, the *Velocity of Change* metric is a vector produced simply by subtracting the X and Y coordinates of the current *Center of Change* from the previous one.

The signal values produced by these three metrics (Over All Motion, Center of Change, Velocity of Change) for each region are published to the Shared Dictionary underneath the key belonging to that region (see Table 4.1). This makes the updated values available to the local application and any remotely connected applications.

5.1.4 Ambiguity

Cambience's frame differencing algorithm is crude, and suffers from several ambiguities when it comes to determining the meaning of change. The use of more sophisticated vision algorithms may better distinguish the following actions in a scene; meanwhile Cambience merely sees them in terms of greater and less *Over All Change*:

- **Size** – larger objects occlude and reveal more background than smaller objects.
- **Speed** – faster moving objects may be the same size as slower moving ones, but would register greater change with Cambience due to the longer wake of change left behind them.
- **Multiplicity** – Cambience only sees the combined motion of multiple objects moving at the same time.
- **Perspective** – objects that are closer to the camera appear to occlude more of the background than objects of the same size that are further away.

The ambiguities of the *Center of Motion* measurement are as follows:

- **Multiplicity** – more than one object changing within a region produces an indication of the average position of those independent patches of change.
- **Perspective** – the Horizontal and Vertical Positions are subject to the orientation of the camera, and only apply to the 2D image that the camera acquires. These metrics offer no insight into actual orientation of objects in 3D space. Responses must be calibrated based on the orientation of the camera.

In §3.4 we saw Josh take advantage of the bird's eye view of his camera to monitor activities performed on his desk (e.g. typing on his keyboard). However, if his camera captured his desk from an oblique angle instead, the region may also capture the act of him getting up out of his chair. Without a more sophisticated vision algorithm to intelligently distinguish between these actions, it is critical that users leverage camera positioning and region placement to capture expected actions and filter unwanted ones.

5.1.5 Signaling Change with Region Items

All of the above signals are then visualized and animated within a Region Item (Figure 5.3), as mentioned previously in §3.1.3. When created in the ecology workspace, a Region Item subscribes to metric values in the Shared Dictionary that belong to that particular region. To summarize, these values are:



Figure 5.3: Region Item

- The Over All Motion attribute provides the value of the Over All Change.
- The Center X attribute provides the X-coordinate of the Center of Change.
- The Center Y attribute provides the Y-coordinate of the Center of Change.
- The Velocity X attribute provides the X-magnitude of the Velocity of Change.
- The Velocity Y attribute provides the Y-magnitude of the Velocity of Change.

These attributes are available as sources, as seen by the circles on the right-hand side of the item in Figure 5.3. Each source can be patched to one or more sinks, or left unused, at the discretion of the user. Notification of an updated value from the Shared Dictionary triggers a new signal, which propagates from the corresponding source.

5.1.6 Camera Items

At times, a user may want to view the remote video that triggered a sound, in order to gain further information regarding the cause and meaning of visual change. It is possible to do this by viewing the Remote Camera Preview (Figure 3.5b) in the Cambience interface. However, Cambience is designed to be minimized when one is not directly working with it. When this is the case, it has no more screen presence than a notification icon in the system tray.

When created, the Camera Item is linked to an opaque, floating video window that can be repositioned anywhere on the desktop. This window shows the video stream from the camera of choice (local or remote), and remains visible even when the Cambience application is minimized. The Camera Item itself provides a sink that controls the opacity of the floating window. For an example, if Kate is monitoring Josh, she may want his video stream to be fully visible when he moves in his chair, but be opaque or invisible when he is still (Figure 5.4). This makes it possible for the video window to act as a non-invasive peripheral display when little is happening, but to come to the foreground when potentially interesting things occur.

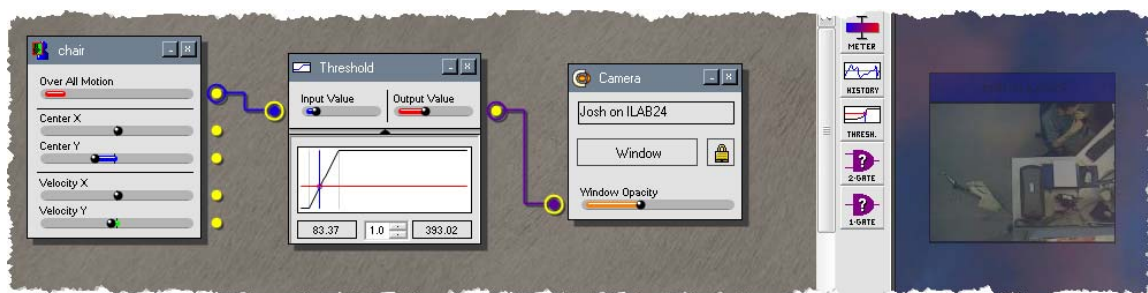


Figure 5.4: The Camera Item (middle) and corresponding video window (right).

5.2 Sound Items

In this section I describe how Cambience handles dynamic sound output through various sound items. Figure 5.5 is a continuation of Figure 5.1, where it shows how the region

metric values from the Shared Dictionary enter a sonic ecology through corresponding Region Items.

In general, a sound item represents an audio clip, where its attributes can be manipulated manually or through incoming signals to affect its playback. The Ecology Workspace acts as a circuit board where users decide how Region Item sources are patched to sound item sinks. Optionally, signals can flow through intermediate items before they reach their destination, however these are not discussed until §5.2.8.

5.2.1 Sonic Ecologies

The concept of a *sonic ecology* or *sound ecology* is one loosely described by Mynatt et al (1998) as a collection of many sounds that occur together. These sounds can be selected based on which ones would be found together in a familiar or natural setting (ie. on a beach, in a forest, on the street, etc), where each sound can have a semantic meaning of its own.

In Cambience we are not restricted to sounds that would occur together naturally, although the listener can certainly design a scheme of sounds based on this criterion. I have typically favored sounds from nature as a means of providing enjoyable and relaxing ambient audio; however users may wish to use man-made or musical sounds as well. For instance an important consideration, assuming it is a desired effect, is to choose sounds that do not occlude one another (as would the roar of a waterfall occlude the sound of light rainfall) to ensure that they are audible in tandem. However, it is also possible that people may want a sound to occlude others if its importance overshadows the rest.

One must also consider that different sound effects demand different levels of attention based on factors such their loudness, sharpness, randomness, or unfamiliarity. For example, the buzz of a mosquito is fairly annoying and attention grabbing, making for a good alarm sound. Other sounds like the ebb and flow of water on a lakeshore are more

passive and peaceful, and are more suitable for sonification of ambient background information.

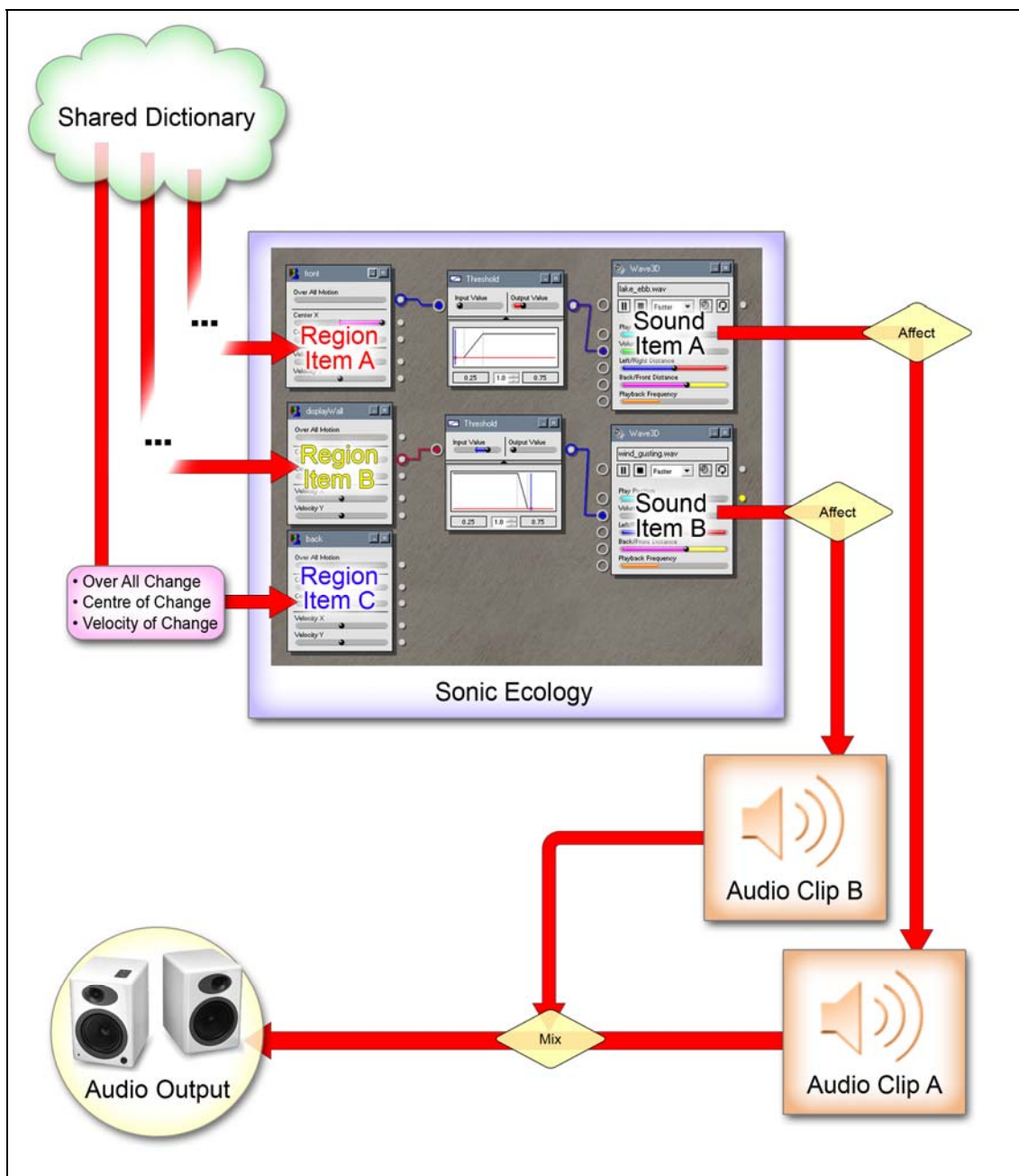


Figure 5.5: Cambience Output Flowchart

Cambience makes sonic ecology design accessible to the non-programmer. However this does not guarantee good sonic ecologies; it is really the task of a professional sound engineer to create a sophisticated and balanced design. In practice, (and analogous to desktop publishing) we expect everyday people will be able to create simple soundscapes of variable utility, while professionals will still be the ones who can consistently deliver a good sonic ecology for complex situations. For that reason, this thesis only makes loose suggestions about how to combine and affect sound through Cambience.

5.2.2 Selecting Sounds

Cambience allows people to import audio files (e.g. music, sound bytes, loops and effects) from their local hard drive into the visual programming workspace. The Audio Browser (Figure 5.6) allows for quick access to sound files, but alternately sounds can simply be dragged and dropped onto the Ecology Workspace. Once a sound item has been created, the playback status and parameters of that sound can be manually set by clicking on the attribute meter, or controlled by sink input.

Audio file formats may differ even among audio files of the same type (e.g. a WAV may be stereo or mono). Consequentially, three different audio items were created to represent three particular classes of audio in the visual programming workspace. The commonalities between all three items are:

- **Playback Controls** – seen in Figure 5.7 as the widgets within the light-colored rectangle at the top of each item. The name of the audio file is displayed (A), with play/pause button (B), stop button (C), mute button (E), and the loop/one-shot toggle (F). Sudden attribute changes (e.g. a jump in volume) are applied gradually so as not to be jarring, and (D) designates the transition speed.
- **Playback Source/Sink (G)** – allows the playback state to be controlled by an incoming signal. A signal that has its current value equal to its origin (see Table 4.2) would evaluate to false, and would cause playback to cease. Any signal value not equal to its origin causes playback to continue. If looping is turned on

(F), playback is paused and resumed respectively. Otherwise if it is one-shot, the sound is stopped and played from the beginning again. The Playback Source on the opposite side sends updates of the playback state when affected by the playback sink or when manually altered by the user. Figure 5.8 demonstrates how to synchronize playback in a chain of sounds. Pressing Play on the leftmost sound causes a chain reaction that starts the other 3.

- **Play Position Attribute/Source/Sink (H)** – controls the temporal position of an audio file if it is currently playing. When the sink is patched, the play position may scrub or jump around. The Play Position Source reports the new position when affected by the sink, when the Play Position is manually changed by the user, or when it is updated as the file plays.
- **Volume Attribute/Sink (I)** – affects the volume of the sound. Once again this can be controlled by input received by the sink, or manually affected by the user while no direct source is attached.

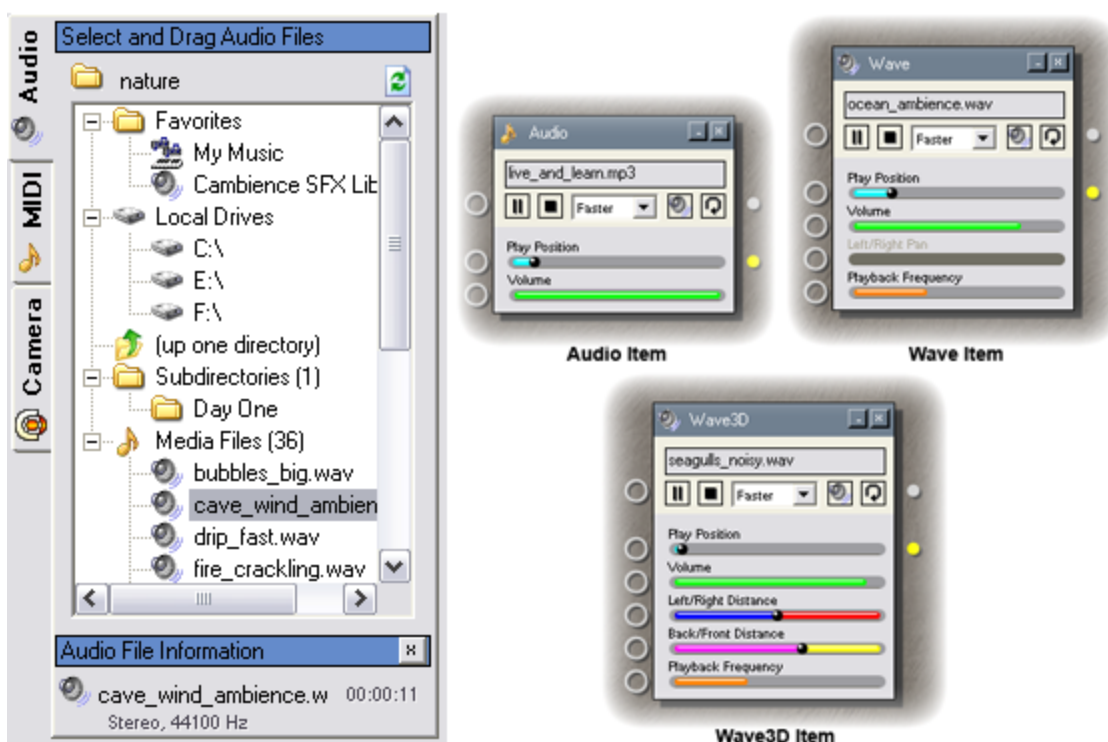


Figure 5.6: The Audio Browser (a) and the three sound items (b-d).



Figure 5.7: Common sound item components, seen on an Audio Item.

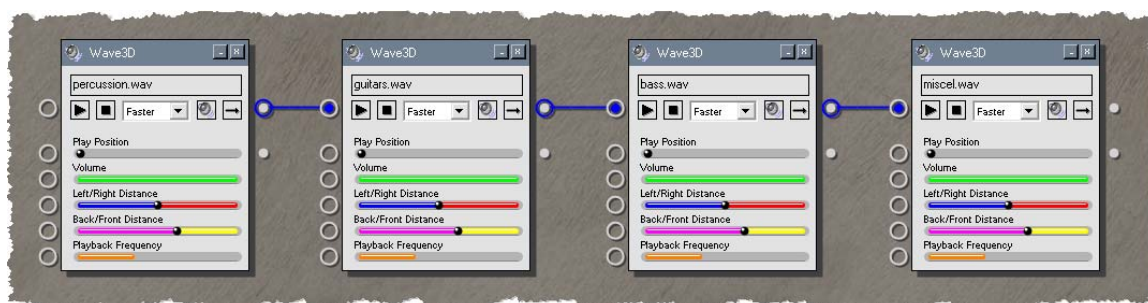


Figure 5.8: Synchronizing a chain of sounds using their Playback Sources/Sinks.

5.2.3 The Wave3D Item

The *Wave3D Item* (Figure 5.6d) has the largest variety of parameters for playback control, but it can only be used with monophonic WAV files. A single channel audio file can be subjected to positioning effects in virtual 3D space, whereas an audio file containing data for multiple channels (most commonly stereophonic) already has built-in position information. In Microsoft DirectX, a mono sound is treated as originating from a point in virtual 3D space, and the loudness of that sound is scaled per speaker relative to the virtual 3D position of the listener.

Changing the left/right distance and front/back distance attributes of the Wave3D Item alters the location of the sound in virtual 3D space. Although changing the altitude (Y-axis) of the sound is also possible, I decided to exclude it since the effect it produced wasn't as easily understood as Pan (X-Axis) and Fade (Z-Axis). Figure 5.9 shows different left/right and front/back distance values, and their resulting positions in 3D space.

Finally, the Playback Frequency attribute controls the rate (in Hertz) at which the sound is played, affecting sound speed and pitch.

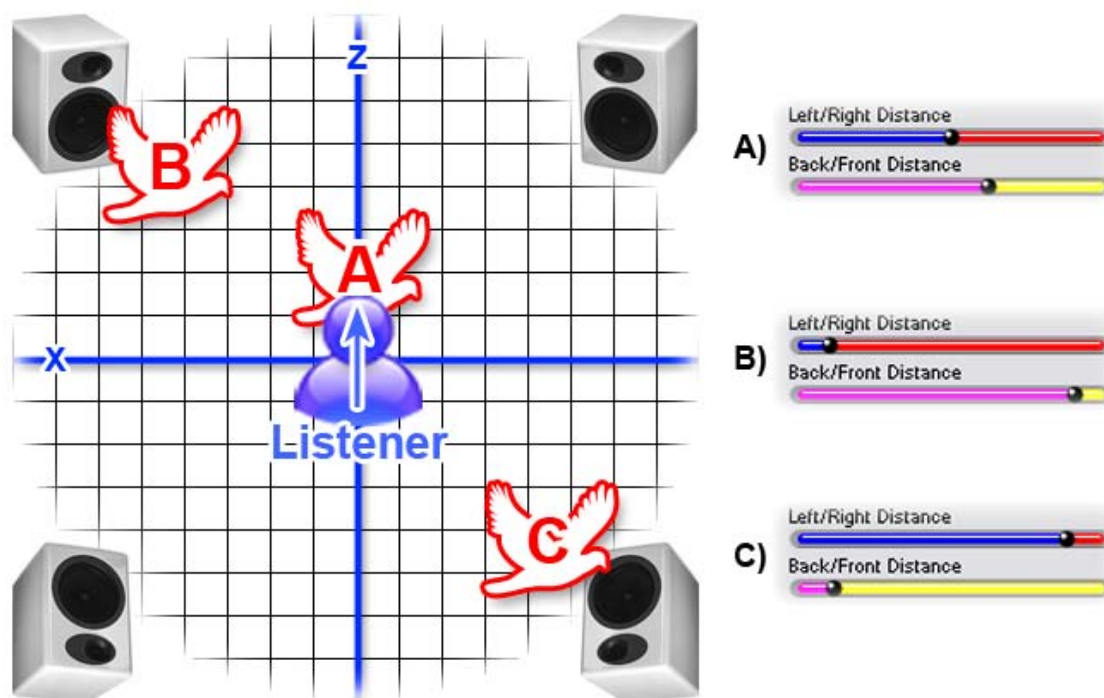


Figure 5.9: Wave3D Item Sounds in Virtual 3D Space

5.2.4 The Wave Item

The *Wave Item* (Figure 5.6c) has a subset of playback parameters from the Wave3D Item. Front/back distance is removed, and left/right pan is relabeled, as the 3D effects of DirectX are not utilized in this item. Left/right pan is only enabled for single-channel (mono) sound effects, but mono files normally default to the *Wave3D* Item discussed in

the previous section. Wave files with more than one channel are wrapped with this item by default. A single-channel (mono) WAV file can be downgraded to this item by holding CTRL when dropping it into the workspace.

5.2.5 The Audio Item

The *Audio Item* (Figure 5.6b) is the lowest common denominator for audio formats. It draws on the capabilities of Windows Media Player and can thus handle encoded formats such as MP3s and WMAs. The disadvantage is that the item can only control playback status, playback position, and volume. As well, simultaneous playback of several audio items may be compromising. Formats such as MP3s require a certain degree of processing to decode for playback, and are thus affected by available processing power. When more than one MP3 plays at the same time, the various audio streams can suffer from dropouts. Somewhat similarly, multiple MIDI files can be played at the same time only if the instrument tracks are mutually exclusive. The Audio Item exists because users may want to use encoded sound files, however these must be used sparingly.

5.2.6 The Playlist Item

The prospect of playing only one or two MP3s at a time through Audio Items does not bode well if a user wants Cambience to have a repertoire of more than one or two unique MP3 sounds or songs. The Playlist Item overcomes this limitation by allowing a large pool of MP3 sounds where only one is playing at a time. Playlist Items (Figure 5.10a) are created from M3U or PLS files, which are plain text lists of file locations on the local machine. This item gives the same capabilities of an Audio Item, but with additional attributes to control playback position in the track list. Sinks are provided that, when triggered with a non-origin value (similar to the Playback Sink), will cause the item to advance to the next file or return to the previous one. A separate player window can be invoked to allow user control of the playlist even when Cambience is minimized (Figure 5.10b).

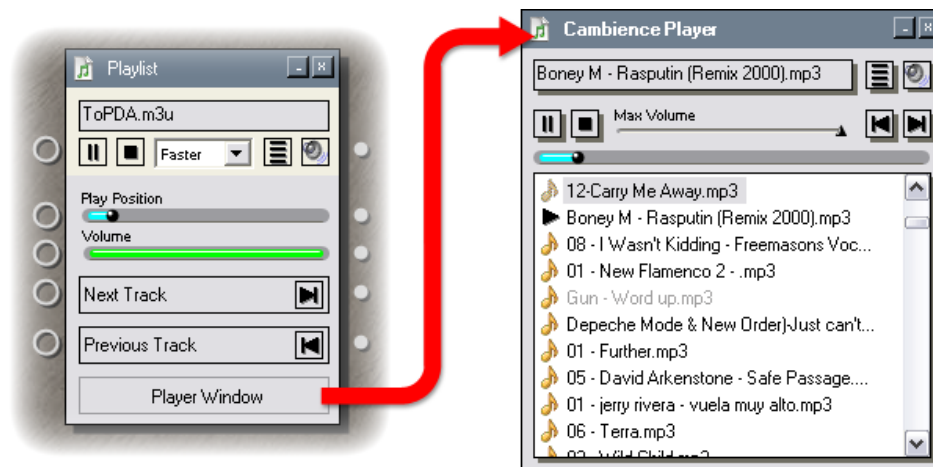


Figure 5.10: (a) Playlist Item, (b) Floating Playlist Window

5.2.7 MIDI Instrument and MIDI Track Items

Cambience also allows for generation of basic MIDI events. The MIDI module allows users to choose an instrument preset for up to 16 MIDI tracks, and then drag it into the workspace where it appears as an Instrument Item (Figure 5.11b). Instrument Items map signal levels to keys on a musical keyboard, and allow control over the velocity of each note press. If the CTRL key is pressed when dropping it into the workspace, a Track Item (Figure 5.11c) is created instead. This item gives control of a few of the MIDI track's controllers: master volume, modulation, reverberation, and pitch bend. Although the modulation and reverberation controllers do not produce noticeable effects on all soundcards, the changing controllers would apply to any instrument items belonging to that track.

Because instrument presets can be assigned to each track, I also created the facility to save and load preset configurations in to an MSC (Midi Scheme) file. The contents of this file format are stored using simple XML. The Scheme Menu button (seen as a "...") in the top right-hand corner of Figure 5.11a) provides a save, load, and reset function.

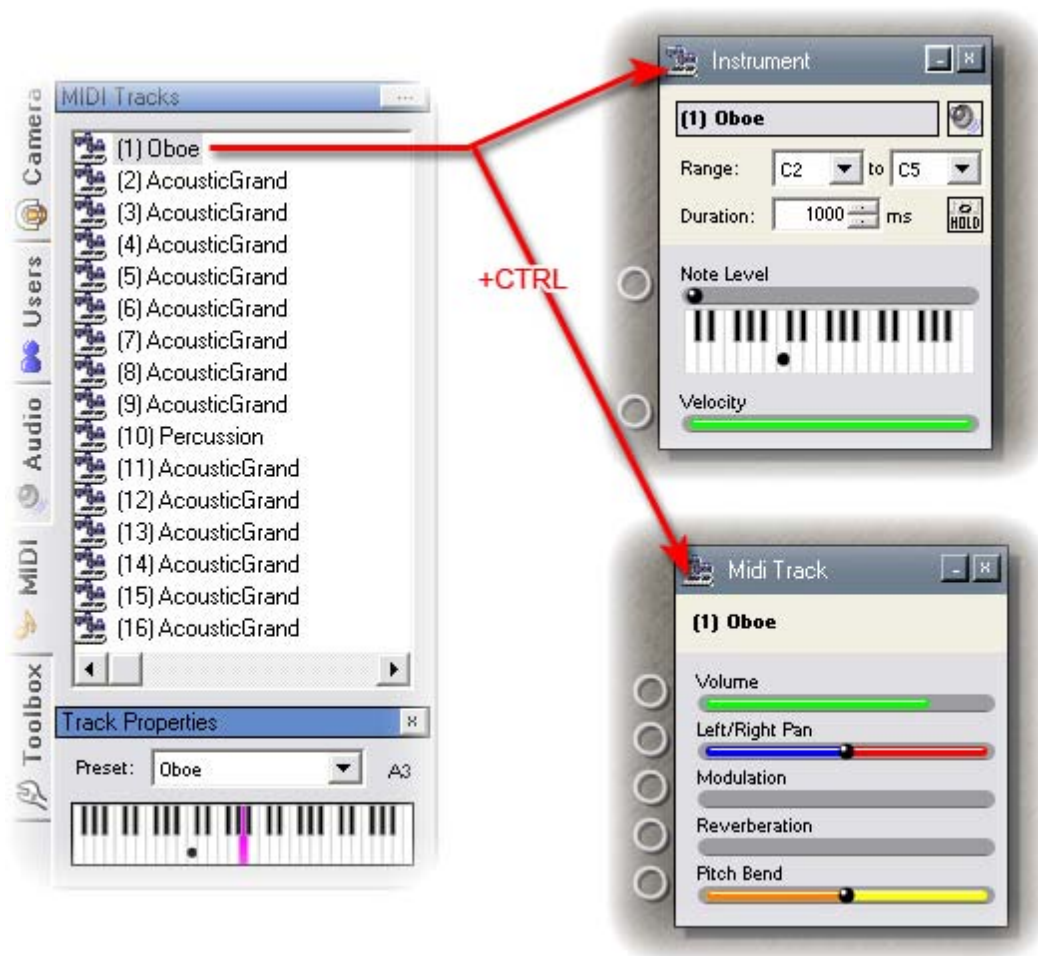


Figure 5.11: (a) MIDI Module, (b) Instrument Item, (c) Track Item

5.2.8 The Cambience SFX Library

When I first released Cambience for my colleagues to try out, the primary complaint was that they did not have any good sounds to use with the program. Granted there are many free sound effects available on the internet, but the average person does not know how to find them, and even so they still may need to edit the sounds to suit their purpose in Cambience.

For that reason I decided to create a small collection of ready-to-use sound effects that I named the *Cambience SFX Library*. These samples were selected from a much larger

sound effects library used for professional film production. They are mostly ambient and natural sounds, many of which I prepared for seamless looping.

Another advantage of having a standard set of sounds, as we saw in §3.1.5 and §3.2.3, is that connected users can easily share the mappings that they create in their workspaces without requiring the transfer of audio files.

5.2.9 Audio Configuration

Ideally Cambience should be run on a computer with a 4-speaker surround-sound system to take full advantage of its 3D positional audio capabilities, powered by Microsoft DirectX (see Figure 5.9). Unfortunately it is more common for computers to have only 2 speakers. In this case DirectX is aware of the limitation and applies volume scaling and muffling effects to mimic 3D sound as best it can for two-channel audio output.

Cambience has a dialog window (not shown) to specify the speaker configuration, and allow users to select which audio output device they would like the program to use.

5.3 Signal Items

While region and sound items suffice to map video region attributes onto audio attributes, the mapping may not produce the desired effect. This is because attributes of video regions do not always correspond nicely with the attributes of audio output, even though all signals are scaled to be compatible (see §4.2.3). Consequently, I include signal items to perform transformations, mathematical and history operations, and logical evaluations on one or more signal. Other signal items are used for debugging and graphing signals. This section goes through the gamut of items available from the Toolbox Panel (Figure 5.12).

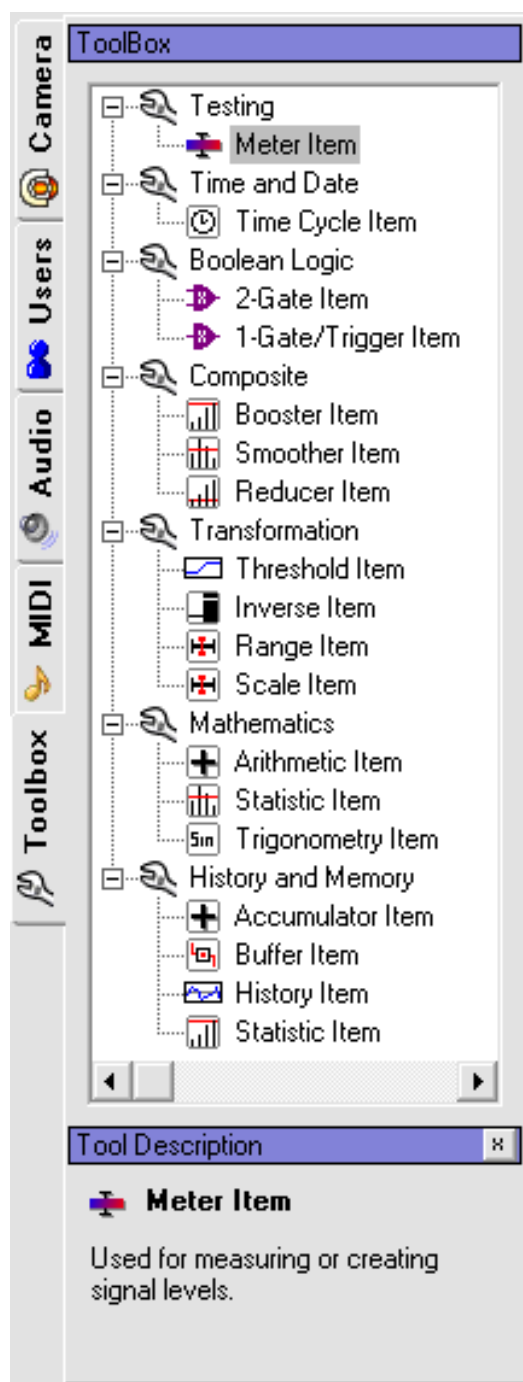


Figure 5.12: The Toolbox Panel contains a list of Signal Items.

5.3.1 Monitoring and Testing

To develop and debug mappings, it can be useful to monitor and/or generate signals.

The *Meter Item* generates output (Figure 5.13a), measures input (Figure 5.13b), and acts as a pass-through that monitors the signal between two items without altering it. The item can also be used to generate a signal within the range of values, or give visual feedback about the properties of the inbound signal (i.e., min, max, and value). The item changes its function between these two scenarios depending on the presence of a sink connection, as seen between Figure 5.13a and b.

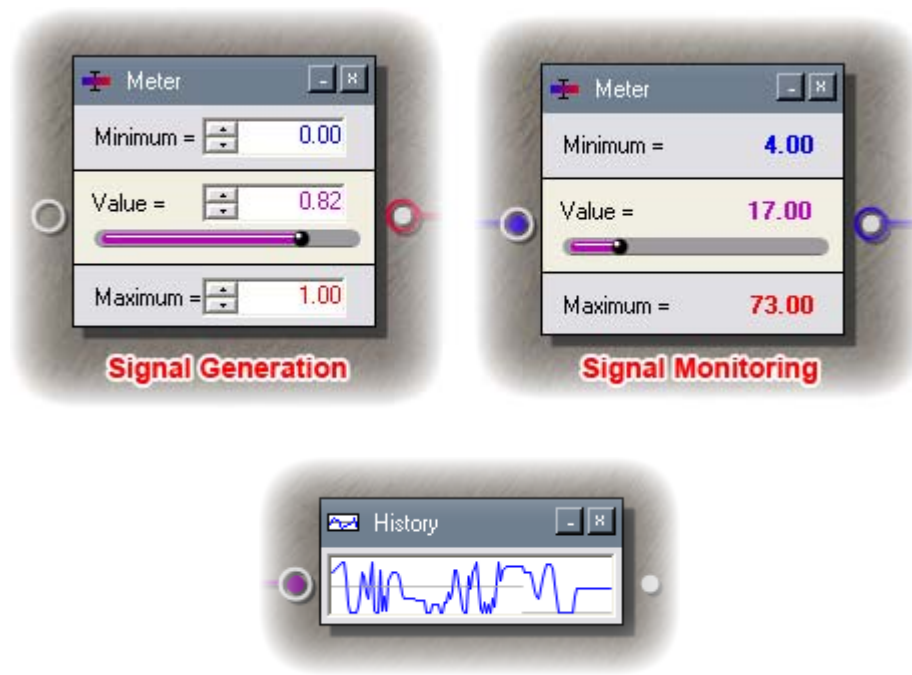


Figure 5.13: (a) & (b) Meter Item, (c) History Item

Although classified under a different category, the *History Item* (Figure 5.13c) is useful for monitoring. Whether used as a pass-through or merely as a sink, it produces a simple graph of the inbound signal values over time, relative to the maximum, minimum, and origin of the signal. This animates a 2-3 minute history of the signal behavior for review.

5.3.2 Signal Transformation

Video attributes do not always affect sound attributes in exactly the way we want them to. For instance, instead of a linear mapping, we may want a sound to be audible with no motion and then go up to maximum volume when modest change is detected. However, if we use a direct mapping of Over All Motion to Volume, the sound would only grow to full volume when the entire region is activated by motion (ie. when every pixel within the region value changes significantly).

Consequently I created several interesting items that give the user more control over this correspondence. The *Threshold Item* (Figure 5.14) is the main work horse. It allows people to reshape any signal value, within the signal's range, using the transformation graph seen as the central white box. The width of the box is representative of the signal input range where the left edge is the minimum, the right edge is the maximum, and the actual signal value falls somewhere in between (shown by the vertical blue line in Figure 5.14d). Similarly, the height of the box also represents the range of the signal (bottom=minimum, top=maximum), but the actual value of the output is determined by the location where the input signal line intersects the graph (visualized as a horizontal red line in Figure 5.14e). Of course, these lines do not appear unless the sink is patched.

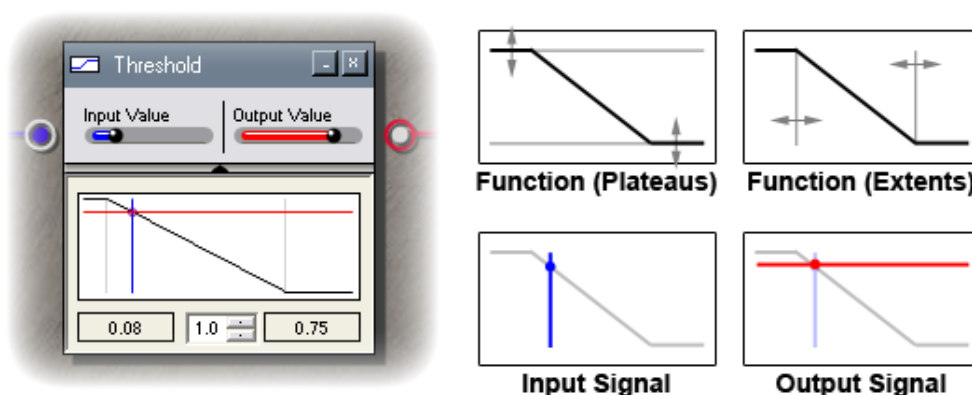


Figure 5.14: (a) Threshold Item, (b-e) Transformation Graph Components

The following subsections explain how the graph can be adjusted, and what effect it has on the signal. To help demonstrate, I will refer to the effect of a Region Item's Over All Motion on a Wave3D Item's Volume when patched through a threshold item, as shown in Figure 5.15.

Plateaus and Extents

The horizontally level areas on the left and right edges of the function are plateaus. They represent a range of signal input values that produce a steady signal output. Meanwhile the vertical grey lines, called extents, mark the boundary between the left and right plateaus and the central blend function. Setting the extents defines what the threshold item considers low, medium, and high input signal values.

Figure 5.17 shows how low, medium and high values can be designated by adjusting the extents. Figure 5.17c depicts how to create an all-or-nothing (Boolean) function by bringing the left and right extents together.

For the sonic ecology in Figure 5.15:

- Figure 5.16a would play the sound at half-volume for low values, taper off with medium values, and be inaudible at high values.
- Figure 5.16b would play the sound at half-volume for low values, ramp up with medium values, and play at maximum volume with high values.
- Figure 5.16c would play the sound inaudibly for low values, ramp up with medium values, and play at maximum volume with high values.
- Figure 5.17a and b would play the sound at maximum volume for low values, taper off with medium values, and be inaudible with high values. Figure 5.17b has a much narrower range of medium values with high values starting a lot earlier in the signal range, thus requiring less change to reach full volume than Figure 5.17a.

- Figure 5.17c would play the sound at maximum volume for low values, and be inaudible at high values.

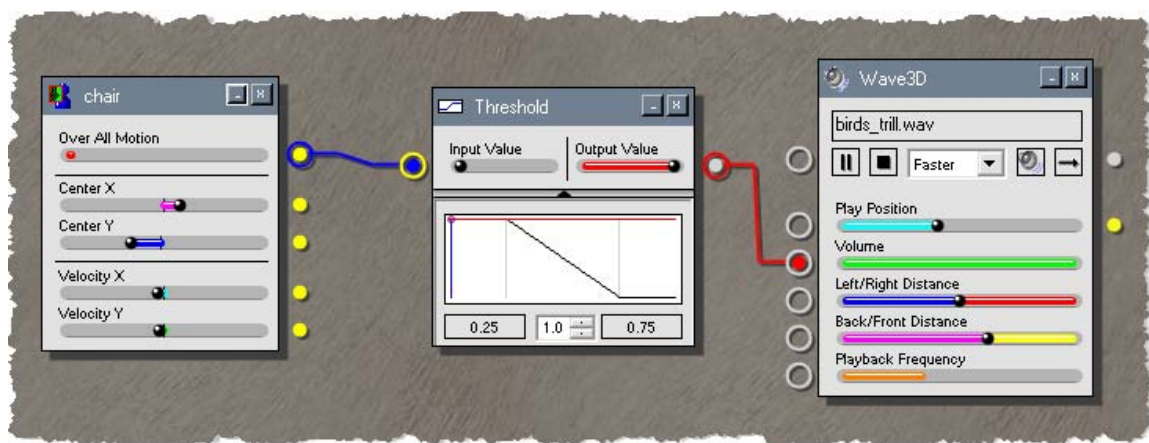


Figure 5.15: A demonstration of the Threshold Item.

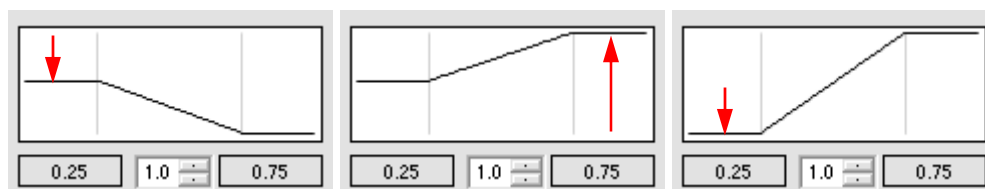


Figure 5.16: A sequence of plateau adjustments (left, right, left).

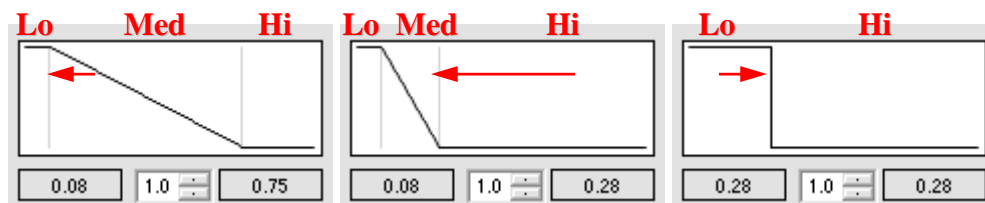


Figure 5.17: A sequence of extent adjustments (left, right, left).

Blend Functions

The Blend Function defines the transition between the left and right plateaus. These functions can take on a number of shapes, and therefore result in various transitions between low and high signals. We already saw the Binary Function in Figure 5.17c, and the Linear Function in the graphs in Figure 5.16, and Figure 5.17a and b.

The Numeric Up-Down widget centered underneath the threshold graph controls a degree parameter, which affects the shape of the blend function. The Linear Functions that were shown prior all have a value of 1.0. Adjusting the value to be greater than 1.0 (to a maximum of 2.0) creates the spectrum of vertically-inclined Cubic Functions shown in Figure 5.18. This function is analogous to the Boolean Function, except the jump is made less abrupt by the gradual entrance and exit slopes on either side of it; the higher the degree, the sharper the jump and the shorter the slope.

Adjusting the degree parameter to be less than 1.0 (to a minimum of 0.1) produces the family of horizontally-inclined Cubic Functions in Figure 5.19. These functions have sudden jumps near the left and right plateaus, but can create a third plateau mid-way; the higher the degree, the sharper the jumps and the wider the plateau.

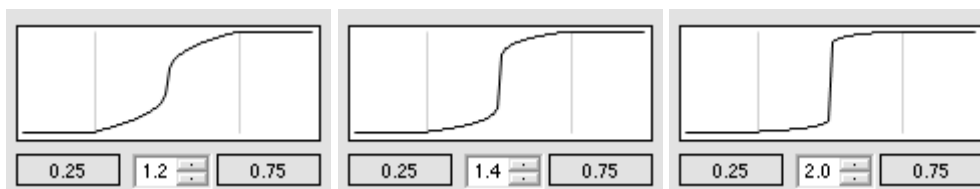


Figure 5.18: The family of vertical cubic functions.

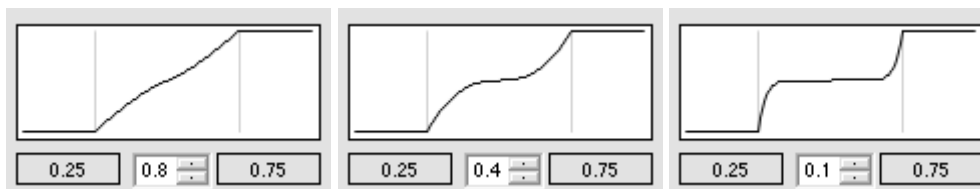


Figure 5.19: The family of horizontal cubic functions.

Figure 5.20 shows three special case graphs:

- *The Identity Graph (a)* – with a linear blend function, the left and right plateaus pushed all the way to the bottom and the top respectively, and the left and right extents pushed all the way to the left and right sides respectively, the output signal is exactly the same as the input signal.
- *The Inverse Graph (b)* – with the same properties as the Identity Graph except the positions of the left and right plateaus swapped, the output signal is complimentary to the input signal.
- *The Constant Graph (c)* – with the left and right plateaus at the same height, regardless of the blend function, the output signal is constant. The height of the plateaus determines the constant value.

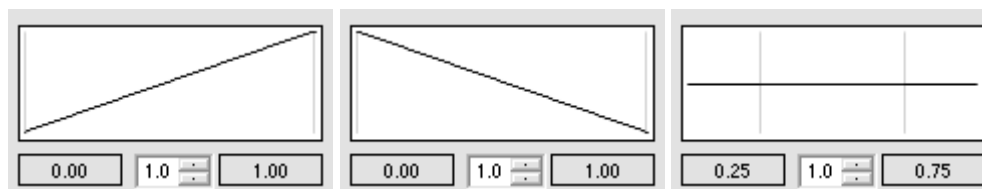


Figure 5.20: Special case graphs. (a) Identity, (b) Inverse, (c) Constant

Threshold Item Summary

To recap, the purpose of the Threshold Item is to provide a rich range of possibilities for reshaping a signal. It does this by projecting the relative value of an input signal onto a user-definable function, which then produces a relative value for an output signal. It is used as an intermediary to boost and scale signals between input-driven and output-driving items that would otherwise not generate the desired effects for a direct patch.

In retrospect, defining the graph as a spline with control points might have allowed for a simpler interface. While straight forward to do, I leave this for future work.

Other Transformation Items

There are three other less commonly used Signal Transformation Items:

- The *Inverse Item* (seen in Figure 5.30) simply produces a complementary signal value. In other words, it implements the Threshold Item with the graph shown in Figure 5.20b.
- The *Range Item* (not shown) allows the maximum and minimum values of a signal to be reassigned without affecting the actual value. However, if the actual value does not fall between the new maximum and minimum values, it is reassigned to the nearest valid value, clipping if necessary.
- The *Scale Item* (seen in Figure 5.25) allows the maximum and minimum values of a signal to be reassigned, and the actual value is rescaled to correspond with the new range.

5.3.3 Logical Operations

When designing a sonic ecology, one may want the ability to activate a sound if a hypothetical *Region A* exceeds a certain amount of change. Slightly more complex, one may wish to only activate the sound if *Region A* and *Region B* exceed certain change amounts simultaneously. For situations like these, the logical operation items introduce a means of performing *if...then* logic on a single signal (1-Gate Item, Figure 5.21a) or two signals (2-Gate Item, Figure 5.21b).

Signal inputs are labeled as Operands and visualized with the standard meter control in both of these items. In order to perform standard Boolean evaluation, each signal must resolve to a value of true or false. Below each meter control is a special bar with an arrow indicator. The user can slide the indicator from left to right to designate the level that the operand must exceed to evaluate as *true*; otherwise it evaluates to *false*.

After the operands are evaluated, they are used to find the result of the selected Boolean operation. The 1-Gate Item has a choice between two unary operators: identity (thru) and inverse (NOT), while the 2-Gate has a much broader range of binary (two-operand) operations: AND, OR, XOR, NAND, NOR, XNOR. The operation is selected by activating the desired radio button. The icons on these buttons show the name and the standard circuit-design logic gate notations.

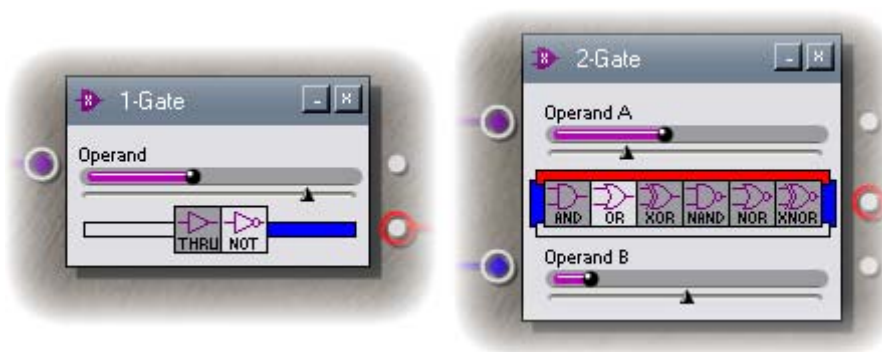


Figure 5.21: (a) 1-Gate Item, (b) 2-Gate Item

Evaluation of the Boolean functions is visualized as well. In the 1-Gate Item, the bar to the left of the operation buttons turns red if the operand evaluates to true – otherwise it is blank. The bar on the right turns blue if the entire operation evaluates to true, and is blank if not. The 2-Gate Item has red bars above and below the operation buttons, corresponding to the operands. The blue bars to the left and right indicate the output of the operation.

The result of the operation is pushed out through the second source from the top in each item, as a signal with a range from 0 to 1, and values of exactly 0 or 1 to represent *false* or *true*, respectively. Other sources on these items correspond to the operands, and act as a pass-through for the operand signal only when the entire operation evaluates to true.

I conclude this section with a sonic ecology that simultaneously demonstrates the use of both items (Figure 5.22). In this example, the *splash.wav* sound is triggered to play one-shot when the Over All Motion from the *chair* and *desk* regions exceed their designated

thresholds. At the same time, the *peacock_frantling.wav* sound is triggered when Over All Motion from the *desk* region exceeds its threshold.

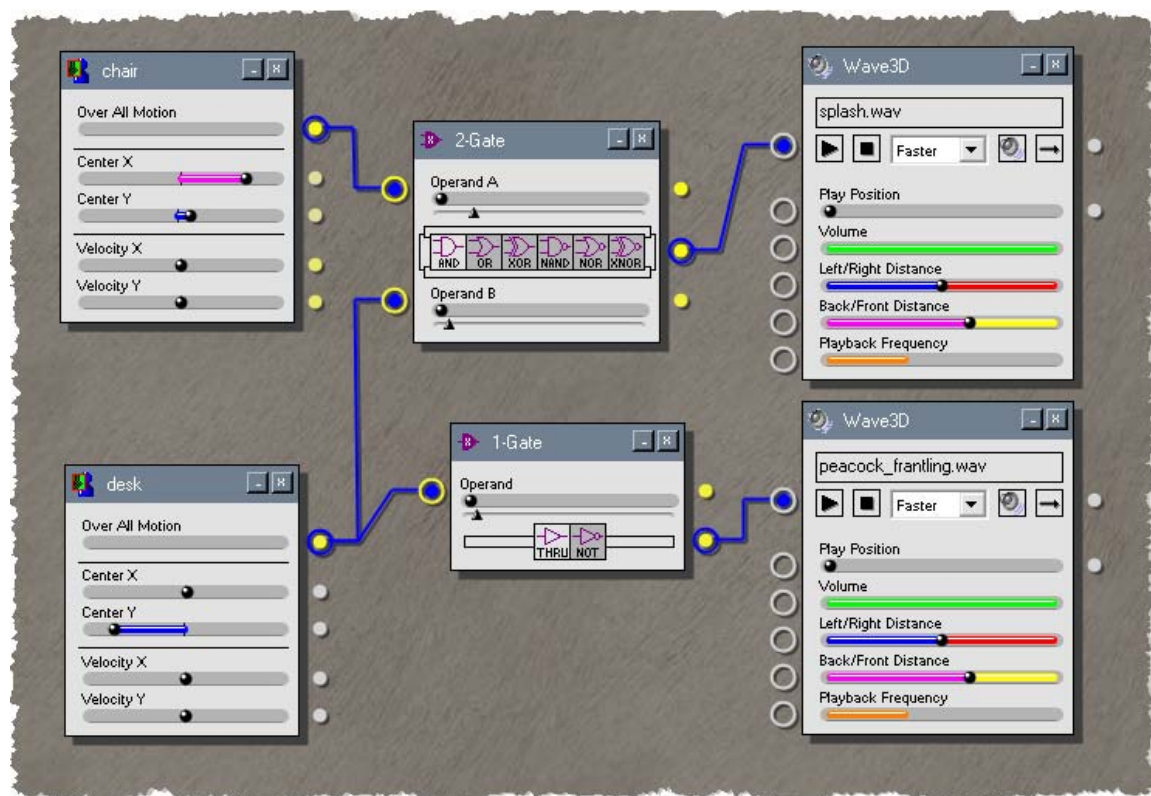


Figure 5.22: A sonic ecology demonstrating 1-Gate and 2-Gate Items

5.3.4 Mathematical Operations

Another useful capability might be to vary a sound's volume based on the sum of Over All Motion from the hypothetical *Region A* and *Region B*. Or one may want motion in *Region B* to cancel motion in *Region A*. These behaviors and many more are possible through the three math items.

Most basic is the Arithmetic Item (Figure 5.23), which can perform a variety of arithmetic operations on two signals (a), or a signal and a constant (b). The operations possible are addition, subtraction, multiplication, division, exponent, root, and logarithm.

The item has sinks for up to two operands: Y and Z. A mathematical formula is shown for the selected operation, and the signal value from the patched sink is fed into the corresponding variable - otherwise the user-definable constant is used for the operand. The radio button beside each operand is a means of setting the dominant sink. This comes into play when performing infinite loop detection (not discussed further in this thesis).

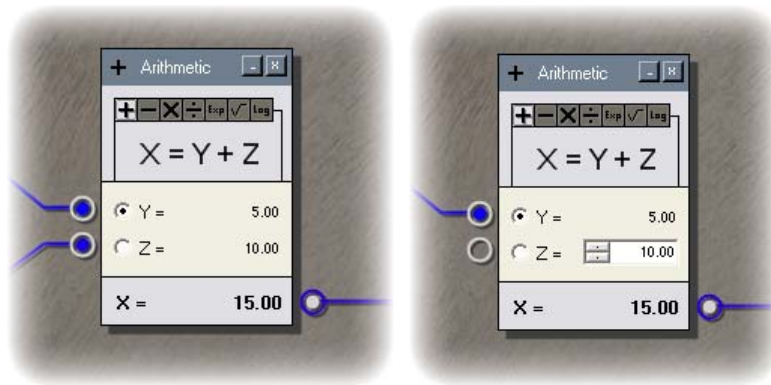


Figure 5.23: The Arithmetic Item with (a) two patches, (b) one patch and a constant.

The Trigonometry Item (Figure 5.24a) is similar to the Arithmetic Item except that it is single-operand and can perform the 6 standard trigonometric functions. Users can select whether the signal value is treated as degrees or radians. Figure 5.25 demonstrates how this item can be used to affect 3D sound rotation.

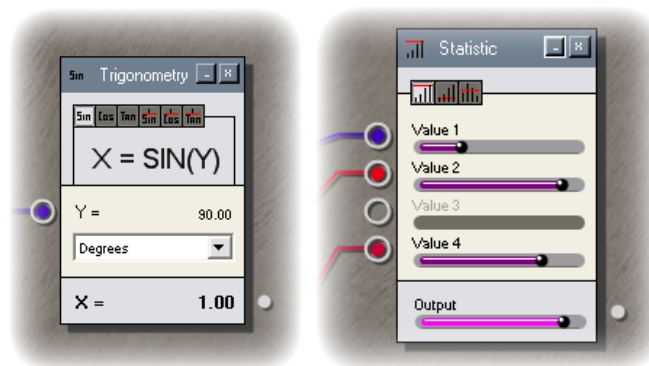


Figure 5.24: (a) Trigonometry Item, (b) Statistic Item

The Statistic Item can evaluate the average signal level of up to 4 signals, using one of 3 functions: Maximum, Minimum, and Average. Sinks that are not patched, such as *Value 3* in Figure 5.24b, are not included in the calculation. Figure 5.26 demonstrates the use of the maximum function of a Statistic Item, on two regions. It shows the Over All Motion attributes from each of the two regions connected to the Statistic Item, where the *chair* region is detecting motion and the *desk* region is not. The output value is the relative maximum of the two signals, and thus the value from *chair* is selected.

5.3.5 History and Memory

All of the items that we have seen to this point have dealt with immediate signal input – that is, the outputs are dependent on the most recent signals received through sinks, or manually specified by the user. History and memory items are a slightly different breed in that they keep a buffer of signal inputs, which are used to calculate an output.

The Accumulator Item (Figure 5.27) is a variation of the Arithmetic Item except with only one input. Each new signal is placed into a buffer, and the selected mathematical operation (sum, difference, multiplication, division) is performed across all values. The new signal is treated as the Y operand while the result of the operation on buffered signals is treated as the Z operand. The output is constrained to fit within the input signal range, so accurate values may not always result. Figure 5.27 shows a demonstration of how the subtraction function can be used to find the difference from the last signal value. This causes the volume of the sound to jump only when Over All Motion is increasing (since $Y > Z$, and therefore $Y - Z > 0$). Note that when Over All Motion is decreasing, the difference produces a negative value ($Y < Z$ therefore $Y - Z < 0$) but is rounded up to 0 to fit in the signal range for Over All Motion.

The Buffer Item (Figure 5.28a) is a first-in first-out queue with a customizable depth. Signals are stored in the queue until it is full. Then as new signals arrive, the old signals are pushed out the other end. It has the effect of postponing signal throughput, and can

be used to create a delayed response. In this case, the delay is not based on a set amount of time, but rather on the frequency of signal input.

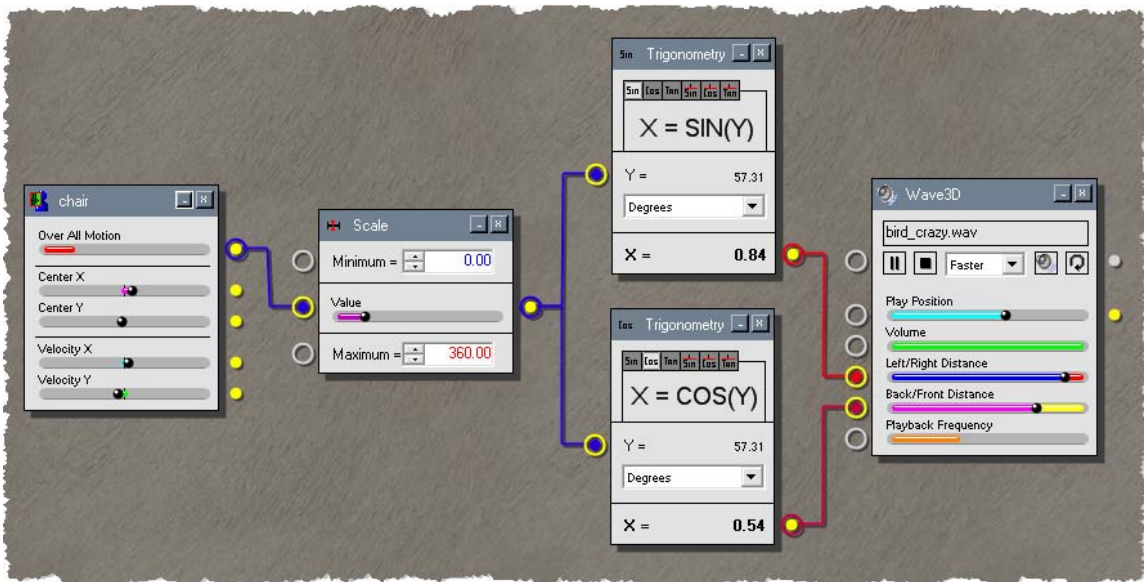


Figure 5.25: Utilizing the Trigonometry Item to perform 360° sound rotation.

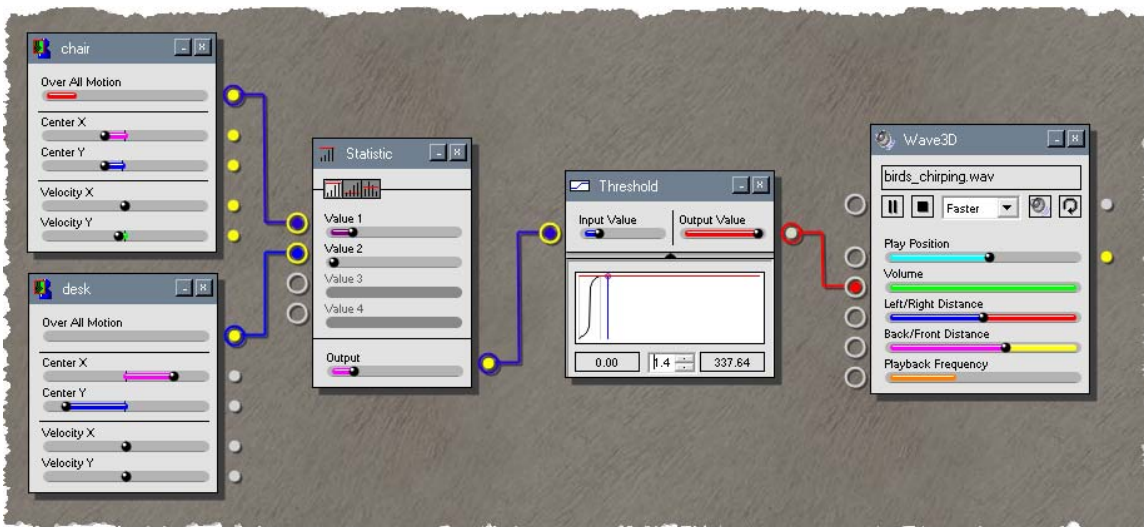


Figure 5.26: A demonstration of the Statistic Item's maximum function.

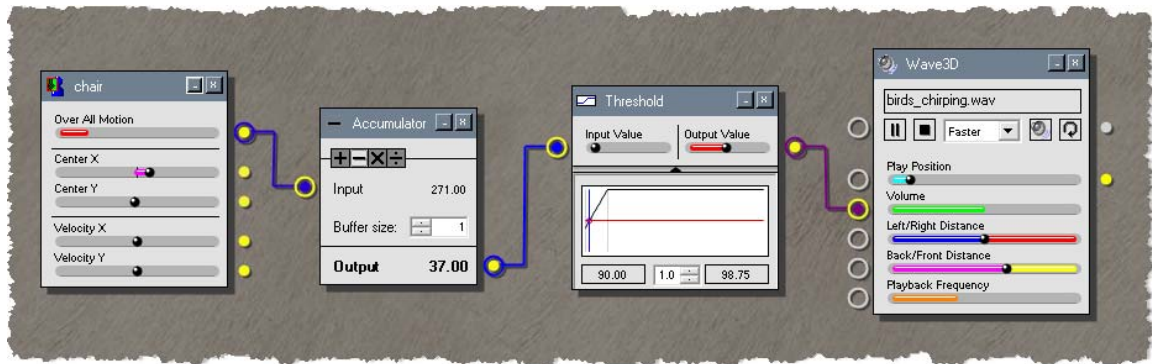


Figure 5.27: A demonstration of the Accumulator Item.

The Buffer Statistic Item (Figure 5.28b) has the same three functions as the Statistic Item (Maximum, Minimum, Average), but it performs the selected function on a buffered history of incoming signals. All three functions are useful in smoothing a signal history, where Maximum favors higher values and Minimum favors lower ones. Again, the queue depth can be customized. A higher buffer size can cause Maximum or Minimum values to hold for longer, as more time is needed to push them out of the queue. When the Average function is selected, a higher buffer size means that the output changes more gradually. The output value of the item changes only when a new signal arrives.

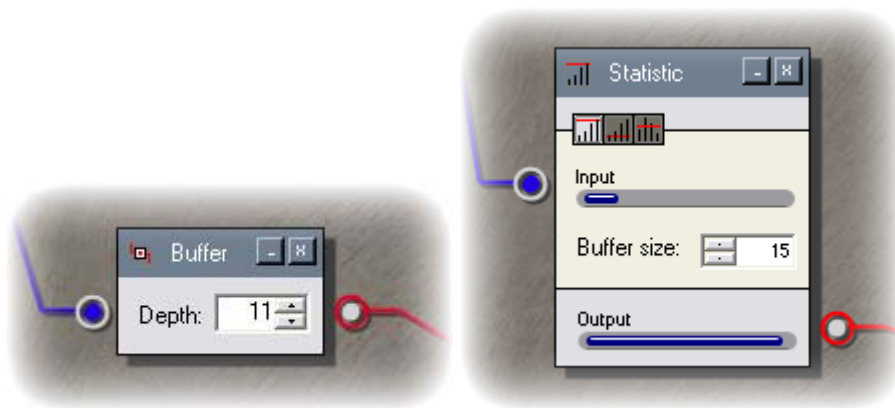


Figure 5.28: (a) Buffer Item, (b) Statistic Accumulator Item

5.3.6 Time and Date

Cambience is a program that is meant to be left running for long periods of time. Certain sonifications may only be appropriate at certain times of the day, or adjustments may need to be made to signals as the scene's lighting differs between daytime and nighttime. For instance, lower lighting levels at night may cause change detection from a Region Item to produce lower values than the ecology configuration was designed to handle. It may therefore be necessary to boost the signal level when it is night.

The Time Cycle Item was created to bring an awareness of time into the visual programming language. The cycle interval can be set to one of six different time scales: minute, hour, day, week, month, and year. The cycle graph is much like the graph in a Threshold Item except that it has a plateau in the middle, a wrapping plateau on the left and right, and two transition functions. The plateau levels can be dragged to the desired heights, and the left and right extents of each transition function can be repositioned. The four beige boxes below the cycle graph show the locations of the left and right extents for each of the two transition functions, in the selected time cycle unit (year, month, week, day, hour, or minute) as seen in Figure 5.29b-g.

Aside from being used to boost or fade signals at nighttime, the Time Cycle Item can also be used to create time-based alarms, or fade continuous sounds on an interval. Using this item, a different sound scheme can even be played between day and night! Figure 5.30 demonstrates how to use a Time Cycle Item to transition between two sounds based on day versus night. An alternate approach could be using a 1-Gate Item immediately after the Time Cycle Item to start one sound while stopping the other.

5.3.7 Composite Items

With all of the tool items at hand, it can be tedious to compose complex behaviors (especially if multi-item structures need to be duplicated within an ecology configuration). The ideal solution would be to create items that iconify a sub-ecology,

allowing for abstraction and replication of customized behaviors. However, because of its complexity, I have left such a feature as future work. In the meantime, to simplify a number of common multi-item behaviors, I created a handful of “Composite Items”.

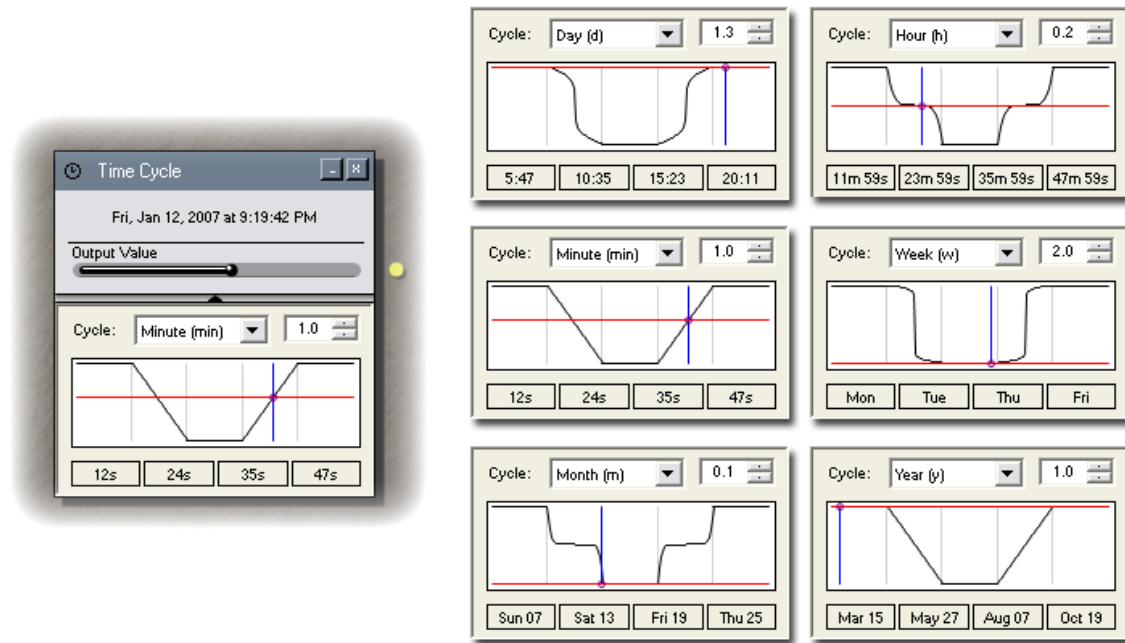


Figure 5.29: (a) Time Cycle Item, (b-g) Various Time-scales and Graph Functions

All three items serve a common purpose: to smooth a signal history and optionally apply an adjustment factor from a Time Cycle Item. These items can be described as a combination of an Arithmetic Item (providing multiplication of the time adjustment), a Threshold Item (to transform the signal), and a Buffer Statistic Item (to sustain or smooth the signal). Uniquely however, an inactivity timer will add zero-value signals to the buffer if no new signals arrive for a certain period of time. This ensures that sounds do not get stuck in an activated state when there is no continued change.

The main difference between these three types of items is the mode of the buffer statistic function. A Booster Item utilizes a MAX function to favor higher signal levels, while a Reducer Item uses a MIN function, and a Smoother Item uses an AVG (average)

function. Each of them starts off with a ready-to-use preset, however parameters can be adjusted as needed. Figure 5.31 shows the complexity difference between (a) using a Booster Item, and (b) nearly achieving the functionality of the Booster Item manually since the timed drop-off is unique to this breed of item.

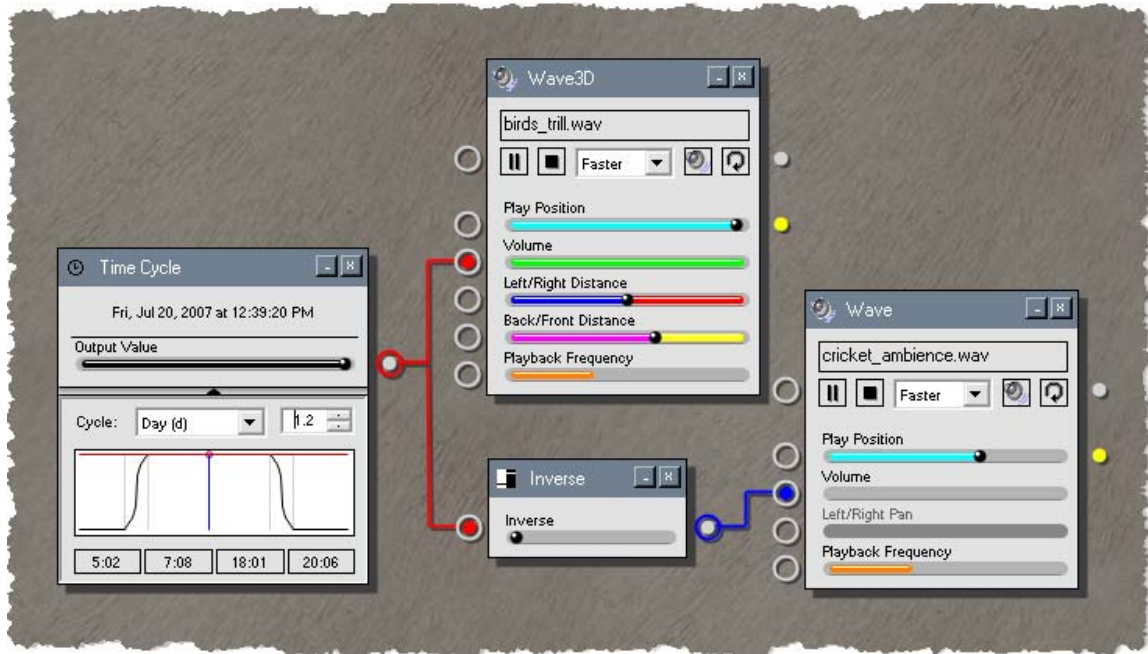


Figure 5.30: Cross-fading two sounds between day and night.

5.4 Extensibility

Some of the items mentioned previously have obvious value; others are more complex and their actual use may be somewhat more questionable. However, all illustrate how Cambience can accommodate a wide range of items through a standard interface of sources, sinks and patches. Indeed, Cambience is written to be extensible, allowing programmers to add, delete and modify modules to experiment with different capabilities.

On an architectural level, Cambience is a framework that accepts a series of plug-in modules, each of which contain a set of Cambience components. A module can add its own control panel into the Control Panel Area (Figure 4.1), a program menu, quick

toolbar items, and most importantly a set of ecology items that can be dropped into the ecology workspace. Cambience comes standard with the Camera Module, User Module, Audio Module, and Tools Module. Table 5.1 lists the contribution of each module to the Cambience interface.

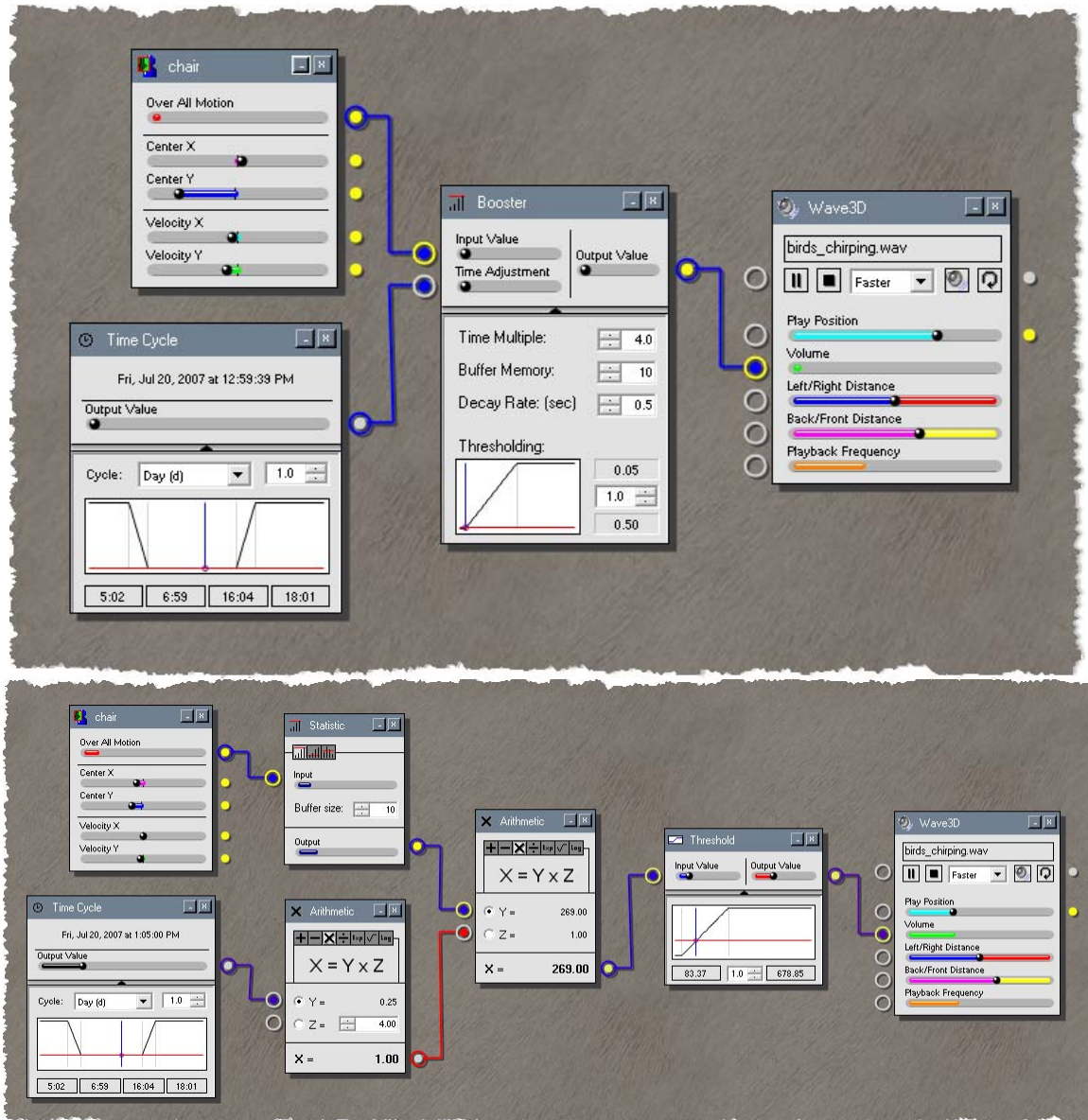


Figure 5.31: (a) Using a Booster Item, (b) A simulation of Booster Item functionality.

Module	Control Panel(s)	Ecology Item(s)
Camera Module	Camera Control Panel	Region Item Camera Item
Audio Module	Audio Control Panel MIDI Control Panel	Audio Item Wave Item Wave3D Item Playlist Item Instrument Item Track Item
User Module	User Control Panel	User Item
Tools Module	Tools Control Panel	(See Figure 5.12)

Table 5.1: Contributions of Cambience's standard modules.

It is possible to program one's own custom modules to extend the program's realm of capability. A few other module ideas are listed below, such as:

- A module to interface with physical widgets known as Phidgets (Greenberg and Fitchett, 2001). This would enable Cambience to receive input from physical knobs and switches, dials, light sensors, pressure sensors, proximity sensors, RFID tag readers etc. and map output to LEDs, servo motors, digital LCDs and more.
- A module to interface with Distributed Phidgets (Marquardt and Greenberg, 2007), such that Cambience would connect to a Distributed Phidgets server and receive input from and/or map output to remote physical widgets.
- A module to interface with an instant messenger such as MSN Messenger, to import status change notifications of specific contacts on the local user's list.
- A module to detect and export keyboard and mouse activity, so that other users may monitor these values much as they would a remote region.

To maintain focus on the main goal of my thesis I have not implemented any of these modules. Nevertheless, they are entirely possible and would provide interesting possibilities for monitoring and exhibiting information outside of strict video to audio mapping.

Modules exist as .DLL files that Cambience loads on startup, so new modules can be added to any Cambience configuration without requiring a program rebuild or reinstall. Software Developers create modules by following the guidelines of the Cambience API (not yet published as of writing this thesis). When loaded, modules are given access to various containers in the Cambience interface and must insert their own controls (e.g. control panel, quick toolbar items, program menus). The specific programmatic details of implementing a custom Cambience module are beyond the scope of this thesis.

5.5 Summary

In this chapter:

- I explained the process and limitations of Cambience's current vision algorithm and video metrics. Two Visual Items were revealed: The Region Item as the gateway to video input, in that it delivers various change measurements as signals from a particular area of live video. The Camera Item as a means of presenting the literal video as peripheral information.
- I revealed and explained Cambience's repertoire of Sound and MIDI Items. Most importantly, the Wave3D Item which offers the highest degree of playback control for specific sound formats, while the Audio Item has wider compatibility with less control.
- I explained the disadvantages of direct mappings between Visual and Sound Items, and introduced the use of Signal Items to remedy the issue. In many cases I demonstrated how these items can contribute to generating unique and useful output effects.
- Lastly I explained Cambience's modular setup, and how the four standard modules contribute to the interface and item repertoire discussed throughout the chapter.

In the next chapter we will demonstrate Cambience's worth by using it to accomplish the goals of a number of past sonification research projects.

Chapter 6. Applications and Case Studies

After taking a look at a theoretical usage scenario in Chapter 3, outlining the program components and features in Chapters 4 and 5, we can now reflect on the realm of practical and artistic applications for Cambience. Particularly, I will examine the strengths and weaknesses of the system in a variety of situations.

I will focus on four main themes:

- Cambience as a facilitator of informal awareness between intimate collaborators, family members, and more.
- Cambience as an interactive art piece, whether used to augment an existing artistic space, or as a stand-alone audio environment.
- Cambience as a security system for providing live notification of change in private and public areas.
- Cambience as a system that replicates and extends capabilities of prior systems.

6.1 Case Studies

In this section, I will examine the strengths and weaknesses of the system that were discovered through a number of informal case studies. These are not formal evaluations. Rather, they are experiences that give insight to Cambience uses, successes, and limitations in various domains.

6.1.1 Workgroup Space

Observation

“Eat your own dog food” is an informal evaluation methodology that suggests that developers use the product being developed in their day-to-day operations. While far from an unbiased test, issues, concerns and usability problems often arise that provide the designer with insight into how the system would fare in a wider audience, and what problems should be fixed before such a deployment happens.

Following from this methodology, I decided to try Cambience within my own environment before deploying it to others. I was particularly interested in seeing how Cambience’s soundscape matched what was actually happening in the scene, and how others who co-habited the environment would react to it. I deployed Cambience in the research laboratory where I work. My desk was situated behind a short partition; on the other side of it was the main entrance to the laboratory, a large lunch table and small kitchen area, the office doors of the three resident professors, the lab bulletin board, etc. I mounted one web camera near the ceiling above my desk, which captured the scene on the other side of the partition. I mounted a second web camera (attached to a different computer) that mostly captured the same area from a different angle but that also captured part of the hallway outside the main entrance. I could see some of this scene by glancing around the side or over the top of the partition. I could also hear the physical sounds as they were generated by people in the space. At any time, I could check the correspondence between the soundscape and what was actually happening.

In agreement with our expectations, Cambience proved personally useful even in this simple context.

- **Informal Awareness** – Cambience added to my awareness of what was happening in the monitored space. I normally wore headphones (as many people do in public spaces to keep audio private), which reduced my ability to hear natural cues that would draw my visual attention into the space. Cambience’s

audio response through the headphones transcended this barrier; it made me aware of actions that I might have missed otherwise.

- **Surveillance** – As my desk was the closest to the main entrance, I was responsible for spotting strangers that entered the lab, and confronting them so that they would not wander through the lab space unattended. With a region around the main entrance driving an audio notification, I knew exactly when someone entered the lab, and could thus keep an eye on them and assist them as needed. Cambience was very reliable in detecting this change. However, Cambience also introduced ‘noise’ into the system, for I received the same notification as people left the lab through that entrance.
- **Casual Interaction** – My supervisor’s office door was in this scene. To help me know when he was around (he tends to be a ‘hall wanderer’) I set up a region and notification around his office door. Thus I knew when someone walked in or out of his office, giving me a sense of his availability. Johnson and Greenberg argued that transitions, such as when people enter and leave offices, are good opportunities for interruptions and conversations (1999).

Yet problems did arise, not from the technology and my personal use of it, but how others in the surrounding environment perceived and reacted to different aspects of the system.

- **Reaction to being captured on video** – A few individuals did not feel comfortable being captured by a camera in this public space for fear of being recorded. This fear existed in spite of the fact that Cambience was *not* maintaining a video record, and that Cambience was only showing its information to a person (myself) who could already monitor the scene by being physically co-present. Part of the problem is that people could only see the camera; they did not know how its video stream was being used. Consequently, I turned off frame transmission (sending only the region analysis data) to ensure no video was being transmitted over the network, and explained to these individuals that Cambience does not record the stream – it only reacts to video in real time. Even so, they felt uneasy.

- **Reaction to audio presentation** – When I left the laboratory, I occasionally left Cambience active, with the volume of the generated soundscape set at what I believed to be a reasonable level. Others in the environment could hear this soundscape. When I returned, I would consistently find the speakers turned down or completely off. It was difficult to pinpoint exactly who had done it and why. Presumably the people who worked near my desk found the sound distracting.

Analysis and Discussion

In spite of its limited deployment and sample bias, this case study revealed both potential strengths and weaknesses of Cambience. As hoped, it facilitated my awareness of events in a physical area that was somewhat outside my direct view. This was personally useful for security purposes, as well as for encouraging casual interaction. The weaknesses are trust and privacy concerning cameras, and sound etiquette in a shared workspace. The presence of a webcam made some individuals feel like they were being watched and - whether working or socializing - this made them uneasy. Furthermore, even a carefully designed sonic ecology proved distracting and not acceptable to some people who had no buy-in to the system.

In terms of camera mistrust, the issue is that people automatically assume that they are being recorded (with audio). Yet this is not the case with Cambience, which actually uses the camera as a sensor. However, camera shyness is a widespread issue faced by many other researchers (Boyle, 2005; Jancke et al., 2001; Lee, 1997; Hayes and Abowd, 2006; etc.)

In a common environment, it takes the objection of only one individual to veto the use of a system like Cambience (Jancke et al., 2001). In order to continue using Cambience when I was personally present, I had to restrict my computer's audio output to headphones. In this case, my belief that Cambience could create an unobtrusive public sound presentation was wrong. Several problems likely caused this:

- People did not know what the sound meant as they had not seen how sounds were mapped to video regions. Thus the sound effects were perceived as noise.
- While I benefited from the sound, others did not (Grudin, 1994). For soundscapes to be beneficial in a shared environment, the sound must be both meaningful and beneficial to all.

6.1.2 Dance Performance

Observation

As mentioned in Chapter 2, performing artists have used video motion detection to generate soundscapes that react to a dancer's motion. Cambience offers similar capabilities, where we believed it adds further value over other systems by letting dancers engineer their own video/soundscape mappings in real time.

To see if this belief bore out in practice, I asked a professional dancer to try out Cambience to create a dance piece, where I would help engineer the soundscape. That is, I would act as chauffer, where I would engineer the video/audio mapping following directions from the artist. This is reasonable, as it may reflect how a dancer would work with a sound engineer. During this process, I elicited reactions about the dancer's experience working through Cambience.

I began with an explanation of how Cambience functions, bringing the artist to the point where she had a basic understanding of its premise. One of the dancer's first reactions during this introduction was the strong preference towards the musical sound generated by MIDI instruments versus the use of ambient sounds. Instruction was by example. I began with a single region encompassing the center of the room, and I connected the X-coordinate from that region to the note playback level of a piano MIDI instrument. I showed her the area of the room that the webcam was capturing, and she was able to move into the region with little difficulty.

At that point, she tried a number of dance maneuvers that moved her arms, legs, hands, head, hips, and torso. After a few minutes she noted that she was able to better keep control of the sounds by using subtle motions, while coarse motions just caused a cacophony of seemingly random notes. She stood completely still until Cambience became silent, and then made some slight flicks with her wrists to hear two or three notes play before settling back to silence. She noted that the sounds being produced seemed a little sluggish in response to her motion, and continued beyond her movements. I made some adjustments and she felt it was better but still a little off.

After she had experimented sufficiently with this simple set-up, she suggested that I reconfigure the scene so that her upper body and lower body would control different instruments – thus a region for the upper half of the camera scene, and one for the lower half. The X-position of the upper region controlled the notes of a piano as in the first configuration, and the X-position of the lower region controlled the notes of a lower-pitched acoustic bass instrument. She experimented by moving her upper body separately from her lower body and was able to get some degree of control. At times, moving her upper body caused her lower body to shift slightly, so it was difficult to keep the two halves completely isolated.

Analysis

This experience revealed several strengths of Cambience's design, in particular, that its working premise was easy for the dancer to understand.

First, she understood the link between video to audio mapping, and was able to exploit it to generate sound from movement. Although she hadn't configured the system herself, it was intuitive for her to interact with its mapping via dance. This was in spite of the lack of visual feedback - she could not see the screen from where she was dancing. Yet she was easily able to keep an approximate mental model of the region volumes because she knew how that space was constructed.

Second, it took her only a few minutes to master each of the two simple configurations that we tried. She was able to adjust her movements to control the soundscape to her satisfaction. That is, she was able to identify the connection between her motions and the sounds, and to control them in an imprecise but still playful manner.

Third, the range of sounds offered the dancer appropriate choices. In particular, the inclusion of MIDI Instruments proved useful because, as a dancer, she was better able to relate to musical sounds rather than to ambient ones.

The experiences also revealed several weaknesses of Cambience, mostly technical. The first issue arose from a less than ideal setting for video processing. In particular, low lighting compounded by similarities between clothing color and room color (floor and walls) affected Cambience's ability to accurately detect changes. For example, because I was wearing dark clothes, it was difficult for me to demonstrate the system to the dancer. However, the dancer's lighter clothes provided better contrast to the room color, and consequently the system responded better to her. A solution is to tailor the setting to favor image processing (e.g., lighting and color choices of costumes), or to consider other methods of gathering movement information in less than perfect environments (e.g., motion trackers, infra-red lighting, etc.)

The second issue arose from the slight time delay between the artist's actual motion and Cambience's audio response. The delay made it challenging for the dancer to control the system. This slight lag exists because of the processing time taken to difference captured frames, produce the binary image, and count the activated pixels in each region. Because of the real-time demands of dance, this delay must be reduced (e.g., by a more efficient algorithm or by moving some of this processing into hardware).

The third issue was that of overly simple mappings. In this case, we simply mapped the X-position of motion to note levels. While the subsequent soundscape resulted in notes, these lacked any kind of interesting structure. Of course, other mappings could have been made. Yet we suspect that dancers (and sound engineers) would likely want fuller access

to MIDI capabilities than Cambience currently offers if it were to be used as a professional instrument.

Finally, the limited camera view and region bounds imposed movement limitations on the dancer. Moving outside of the region and the camera view meant that the dancer was no longer able to interact with the program. This could be solved by better camera (e.g., wide angle cameras) or multiple cameras.

Discussion

Cambience lent itself well to a single person dance performance, where the dancer remained in one spot and moved parts of her body. A number of technical issues did impose some limitations on how the dancer could work with it, most of which could be corrected by routine refinements and extensions to Cambience.

Reflecting on this experience, we see other possibilities for Cambience and the performing arts. One option is to point a camera at an entire stage, and use regions to control instruments or even to trigger sound effects. This strategy could be adapted for a multi-dancer performance, where dancers moving on different areas of the stage could control different instruments. Another option is to consider performing arts other than dance, e.g., plays, improv and comedy routines (all of which usually involve movement by actors). The stage could contain a number of “settings”, and appropriate sounds for each setting could be triggered by the actor’s motion within them. In order to minimize accidental sound generation or to generate special sound effects in a timely way, regions could be placed in areas that are normally outside of the actor’s motion (ie. above the head level); in these cases, the actor would invoke sounds and effects with explicit actions that breach these areas (e.g., lifting their hands above their head).

6.1.3 Art Gallery

Observation

For my next case study, I deployed Cambience as an interactive installation within an art gallery in a popular artist-oriented commercial building. Here, I created a single-machine setup (ie. no network connection to other instances of Cambience), where people's local actions in that space would generate a soundscape that they could hear. In this setting, I expected quite different feedback about the use and effect of Cambience - it would likely involve factors other than the acceptance issues encountered in §6.1.1.

The webcam was mounted to capture a view of the front area of the gallery, including the entrance and a seating area, as well as a number of paintings mounted on the wall on either side of the room (Figure 6.1). The speakers were mounted in two corners of the square area. I taught the gallery manager how to use Cambience's Visual Programming Environment so he could design and customize his own soundscapes.

During our discussions, I initially suggested mood-setting sounds and music playback as people approached particular paintings. The gallery manager mentioned that he had an art exhibit coming in centered on paintings of angels: building on my suggestion, his idea was to fade in choir music when people stepped up to view them. Yet another more immediate idea took over, where the gallery manager decided instead to build a soundscape for an upcoming Halloween party in the building. The manager then found online and downloaded a number of Halloween sound effects, and attempted to construct an interactive Halloween-themed ecology himself. He could not complete this task unassisted: he ran into a few minor snags where I had to help him, e.g., he did not know how to import his custom sound effects into Cambience.

With this minor assistance, he successfully created a playful sonic ecology. Movement in the seating area controlled the volume of the sound of rain, while the front entrance triggered the sound of thunder. One of the paintings on the wall caused a high-pitched scream. The manager had a prop chainsaw that he planned to run at people with, so we

set up a region close to the ceiling that triggered the sound of a chainsaw; this way he was able to trigger the sound when he held the prop above his head.

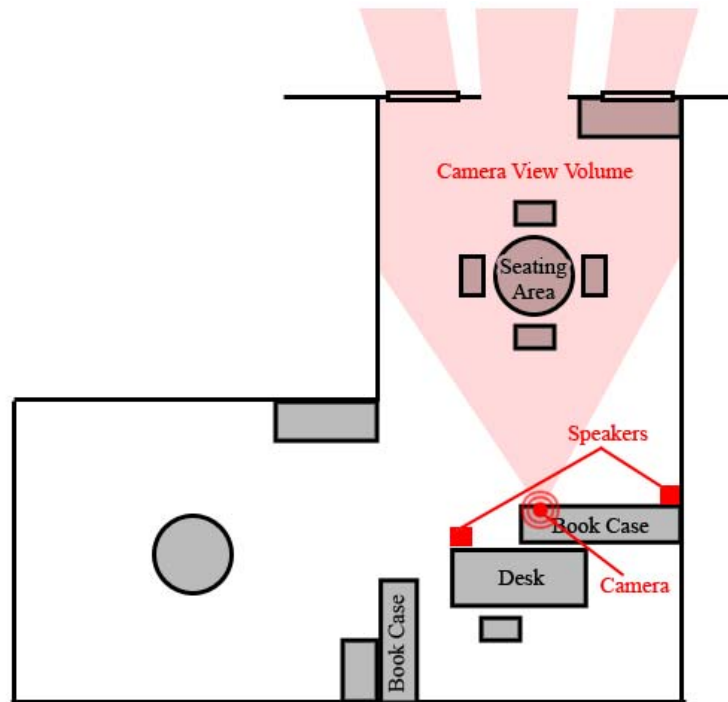


Figure 6.1: Floor plan of gallery and camera/speaker deployment.

Initially this set-up was a success, and received many favorable comments from visitors to the gallery. However, problems arose as daylight faded outside. In particular, Cambience could not cope with the reduced lighting conditions. It did not detect a person that was clearly moving within a region, and the triggered sounds stopped responding - or worse - went off accidentally due to camera noise. Eventually the gallery manager shut the system off, as he didn't understand what was going wrong or how to fix it.

The gallery manager also commented that, after having used the system, he wished he could do away with all of the intermediate items, so that Cambience would adjust everything automatically. This raised a Cambience design question regarding threshold items: in the current implementation, they are almost always necessary between Region

Items and sound items. I discussed it with the gallery manager, and we arrived at the conclusion that it would be easier if some basic signal transformation functions were built into the connections, and not accomplished through separate items. Since this change would require major alterations to Cambience's framework, I opted to leave this idea for future work.

Analysis

The scenario demonstrated Cambience's ease of use, such that an artist with moderate computer experience could learn to compose soundscapes in its Visual Programming Environment after three 1-hour demonstration sessions (albeit with some assistance).

Yet a major implementation flaw was found: how to deal with variable lighting in some environments. Low lighting conditions had only been a nuisance in the earlier dance case study, as it was done in a basement room with artificial lighting. In the art gallery, the factor of varying daylight made the system not only unreliable, but impossible to use at certain times of day. The problem was made worse as the camera driver occasionally adjusted its brightness and contrast settings to compensate, leading to disruptive noise and false positives.

Discussion

In general, the case study reveals that Cambience can be applied as an interactive art installation, where artists can create soundscapes that exploit movement in a physical space. Yet it also uncovered technical problems that had to be solved to handle real world conditions.

Specifically, to mitigate the variable lighting issue as revealed in this case study, I created the Time Cycle Item to handle cyclical lighting changes. Based on time of day, we could direct Cambience to apply a signal boost factor accordingly (e.g., during dusk). However, this required many additional items to calculate the boost from the Time Cycle Item (Figure 5.31b). Consequently, I created the Booster Item to wrap the functionality

into a single item (Figure 5.31a). Although this solution makes it possible to boost change detection signals that are diminished due to insufficient contrast, it does not address the problem of increased noise in low lighting conditions. Many of these problems can be attributed to the use of a low-grade web camera. A better quality camera might reduce the noise in low light, and reduce the necessity for abrupt automatic brightness/contrast adjustments, or at least provide the option to disable them.

6.2 Applications to Prior Research

We can partially evaluate the power of Cambience by asking a) whether the effects found in prior systems that generated soundscapes from video can be replicated, and b) how much effort is required to do so. I do this briefly by looking at two systems that inspired Cambience.

6.2.1 Very Nervous System

The Very Nervous System (VNS), first mentioned in Chapter 2, was an exploration by David Rokeby into the realm of sonifying dance. He originally used a specialized light-sensitive device to capture changes in lighting caused by a person in motion, and used this to generate a dynamic MIDI presentation. The input from the capture device resulted in a coarse grid of lighting levels. The most recent version of the system (known as *softVNS*) is implemented as a Max MSP module, and has migrated away from this proprietary vision device to more generalized video input (Cooper, 1995).

Though little has been published about the actual algorithm that Rokeby used to connect the detected physical motion to the audio presentation, we make two basic assumptions about early versions of VNS: First, that the software driving his system extracted features from individual squares, or perhaps combinations of squares, captured from his input device; Second, that these features were then used to trigger MIDI events, or affect the properties of existing ones in a manner that could be parameterized.

Wrapping the VNS as a Cambience module could thus attain virtually the same functionality, but also have several additional benefits. Many limitations of the system would be lifted, allowing it to cross function with other Cambience items. Multiple instances of the VNS could be connected together through Cambience's networking infrastructure, and interact with one another. Furthermore, the mapping from input to output could be altered on the fly by the user, instead of requiring a recompile of the software by a programmer. This would produce a much more powerful and general system than the early VNS, and the main challenge would be reduced to designing the soundscape so as to replicate Rokeby's compositions.

6.2.2 Iamascope

The Iamascope project by Sid Fels (1997) takes visual input to produce dynamic audio output, as well as visual output that appears as a kaleidoscope. The system uses a single camera for input, and projects onto a 170" screen accompanied by two stereo speakers. Pie-slice shaped regions are extracted from each captured webcam frame and reflected to create a fully circular image, while musical responses are affected only by visual features within the slice.

Currently Cambience does not allow for arbitrary region shapes, therefore, integrating Iamascope as a Cambience module might require reimplementing its own vision algorithm, provided the webcam frames Cambience already captures. Conceivably, Iamascope could be wrapped into two Cambience Items:

- A Kaleidoscope Item might generate the visualization in a separate window (seen as possible from the Camera Item and Playlist Item described in Chapter 5). This item might have output sources for the various feature extractions used by the music production algorithm.
- A Music Production Item could thus have input sinks to accept the values from the Kaleidoscope Item outputs, and handle the dynamic sound production through DirectSound or another library.

Creating and connecting these two theoretical items would replicate the original Iamascope functionality; however a much richer range of possibilities would be available if they were used in conjunction with other ecology items.

For instance, sound items could be affected by the Kaleidoscope Item's output metrics, as could the Music Production Item be affected by Region Items. The Kaleidoscope Item could provide input sinks to control the algorithm's slice angle, offset, and other parameters in real time, which could be affected by Region Items.

6.3 Summary

In this section, we explored a number of themes surrounding the practical application of Cambience. We saw that the system lends itself to the following areas of use:

- As a facilitator of informal awareness between intimate collaborators, family members, and more.
- As an interactive art piece, whether used to augment an existing artistic space, or as a stand-alone audio environment.
- As a security system for providing live notification of change in private and public areas.

We also saw that many existing applications could be made more general and powerful if designed to function through the Cambience framework. Where the programmer is usually needed to fine tune mappings from input to output, here they are needed only create any specialized input or output algorithms as ecology items for the end-user to experiment with.

In the next chapter I conclude this thesis with contributions and future work.

Chapter 7. Conclusion and Future Work

At the start of my thesis, I wrote about my motivations for creating Cambience – a program to monitor video streams that drive audio mappings. I identified this program as having applications in areas such as awareness, security, and performance/art. However, the focus of my thesis was on creating the Cambience system as a gateway to further explore these applications, and other potential for the use of sound at the interface.

In this chapter I discuss my thesis contributions, and ideas for future research that are inspired or based on this thesis project.

7.1 Contributions

The main focus and contribution of my thesis was the creation of a working Cambience system. In order to offer a high degree of flexibility in mapping video to audio, Cambience was developed to be a Visual Programming Language wrapped in a specialized interface.

In review, Cambience met my goals as follows:

- It allows users to indicate regions of interest in live video collected from distributed sites.
- It maps characteristics of these video regions onto audio samples.
- It assembles these audio samples in a way that creates a meaningful soundscape.
- It handles networking as part of the interface rather than as a function of the language elements, to allow ad-hoc program changes without networking interruptions.

- It functions as a media space, where live-captured webcam frames are shared among multiple distance-separated collaborators, and updated regularly.
- It unifies construction and execution modes (interactive not batch), so that users can receive immediate feedback as they compose their visual program.
- It hides unnecessary low-level details by creating high-level designs for the language elements, specifically purposing them for live video input and dynamic audio output.

All of the above goals were met with Cambience's current implementation (v3.4.1), producing a unique tool whose potential applications stretch beyond its original purpose for video to audio mapping. Its creation brought about a great deal of insight, and its premise has already inspired the imaginations of others. This version is only a starting point for future research.

7.2 Future Work

At many junctures, I hinted at research questions that were beyond the scope of this thesis. The remainder of this chapter is an open list of tasks, topics and unanswered questions that follow from my thesis project, which could be worth further research.

7.2.1 Revision and Evaluation of Cambience

Although this thesis produced a working program, there is still much that can be done to improve Cambience, both from a technical and a usability standpoint. For instance:

- A better or more efficient vision algorithm has yet to be implemented.
- Connections between ecology items would benefit from built-in functionality from the Threshold Item. Furthermore, use of a spline to define the threshold function would allow much more flexibility.
- More can be done to expand audio output capabilities. This might include using environmental audio effects on wave sounds, better control of 3D effects, and

more structured interaction with MIDI. Parameterized sound synthesis is another particularly interesting possibility.

- Facility for transmitting custom audio files between users would be a useful addition.
- The framework and existing module interfaces could benefit from some degree of refactoring.
- Other useful modules have yet to be implemented. Earlier in this thesis I specifically mentioned a Phidgets/Distributed Phidgets module, an Instant Messenger (MSN) module, and a Mouse/Keyboard Activity module. There are countless other possibilities.

It would also be helpful to conduct user studies to properly evaluate the Cambience interface, visual programming language elements, and general usefulness of the program.

7.2.2 Sound as an Interface

In this thesis I brought up a number of observations regarding the use of sound as an interface. I questioned the status quo of redundant, underused, and detached audio, and examined the idea of using continuous audio to represent ongoing or background processes in everyday computing.

It would be revealing to conduct a study to identify the benefits and detriments of increased sound use in a program interface. How beneficial is it to give users an increased awareness about the operation of their programs? What useful information can sound convey about program actions and operations? What details are better understood through audio than through visual displays? What types of sound can be metaphorically applied to program operations? Where is the boundary between awareness and distraction?

7.2.3 Social Protocols of Public Audio

I also observed some very stark social protocols that frowned on the public use of sound from computers, especially when sound's meaning is not understood or not relevant to that individual.

It would be beneficial to find out where lines are drawn regarding public audio and social protocols. Why is it that we tolerate natural noise, but not artificial noise from computers? How willing are users to be tethered to their computers by headphones to preserve the solitude of others? Can audio be engineered to minimize disturbance to others while still being useful to the computer user?

7.3 Possible Future Applications

7.3.1 Store Security

Recently in the news we saw an unbelievable string of heists on liquor stores in Calgary pulled by a group of gypsy refugees. The incidents were caught on camera, but this provided no use to the store clerk until the thieves were apprehended in the act. The thieves' faces were obscured with shawls so it was difficult to identify them from the camera footage afterward; as well the clerk didn't notice the store had been robbed until well after the thieves walked out with the store safe!

I can see Cambience being applied to monitor a small set of security feeds within a store, and produce an audio presentation throughout the entire space. This would ensure the clerk is immediately aware of the general presence and actions of customers throughout the store, even if his focus of attention is elsewhere.

In particular, areas associated with items of value or at high risk of theft can be monitored. In the aforementioned situation, a wisely constructed sonic ecology would notify the clerk of the thieves' entry to the back office, tampering with the safe, and

transport of the safe out the front door, even though the clerk may be distracted in the back room or in another part of the store.

7.3.2 Interactive Music in Malls

Malls typically play elevator music in the background; however Cambience could be used to make this presentation more interactive and interesting. Most malls already utilize a large number of security cameras, so existing feeds could be used in Cambience to globally broadcast interesting information (e.g. how busy is the food court, the parking lot, the children's play area, the weather outside, etc.), or simply create a playful sense of engagement.

7.3.3 Interactive Museum Exhibits

I recently paid a visit to the Royal Alberta Museum in Edmonton, which had an exhibit on wildlife in the Canadian Rockies. It consisted of a number of dioramas behind glass, showing life-size models of various animals in their typical habitat. There were directional speakers hanging from the ceiling above most displays, playing a looped audio recording of the ambient sounds representative of the animals and their habitats.

Some of the displays required you to push a button in order to listen to the animal sounds (such as wolf howls). I believe this was necessary because:

- The sound sample was too short or too low-quality to loop and still sound natural.
- The sound was too distracting to have playing continuously with the background ambient sound.

Cambience could be used in this situation to control and trigger sounds without requiring explicit actions such as pushing a button. For example, a person approaching the wolf diorama could trigger the sound of wolf howls or otherwise make the diorama come alive. The background ambient sound could change in volume or timbre based on the amount of motion in the room. Or the presentation could begin as ambient sounds

characteristic of the display, and change to foreground narration as people stop in front of the exhibit.

7.3.4 Aging in Place

In family situations where elderly parents live independently, Cambience could bring an informal awareness of their activities to concerned family members. One issue is whether the aging parents are ok, and a reasonable heuristic is that they are up and about. To capture this, certain ambient sounds may play in the distant family's kitchen as the aging parents go through their daily routine, but the absence of these sounds may prompt a phone call to check that the parents are alright.

Particularly critical events can also be captured and presented. For example, Cambience could be set up to modify the aging parents' medicine cabinet, such that the neigh of a horse could notify the distant family if and when the cabinet is opened. If the family does not hear the sound at the expected time of day, they may call the parent to remind them to take their medication. Likewise, if the sound plays at an odd time of day, or more than once, the family may call to ensure the parent is not in danger of taking an incorrect dose.

7.3.5 Home Security and Notification

When in the home, residents are often not aware of what is happening outside. Someone may be standing on their door step but they are not notified until that person chooses to ring the door bell or knock. Vandalism, theft, and voyeurism can all occur because those outside the house have more awareness cues about the occupants than those inside do about their surroundings.

In a simple situation, Cambience could be used to monitor the front and back doors by playing a sound when someone approaches. Furthermore, different sounds could be used to distinguish the origin of the notification, as is already common with door bells in most homes.

Monitoring areas within the house could provide information to family members as they arrive home, immediately answering questions such as: Is someone else home already? Where in the house are they?

Such monitoring could also provide cues of suspicious activity: e.g. the entire family is in the basement but motion is reported from upstairs. It could also keep parents updated as to the activities of their children in the house or yard, notifying them of their presence in dangerous areas such as a swimming pool or the street.

7.3.6 Abstract Art Piece

There are many different possibilities for harnessing Cambience in this fashion. An example piece might be to position a webcam to observe a fish tank, placing regions over areas of the tank that affect a soundscape. The soundscape may follow the theme by producing an aquatic or underwater soundscape at a remote location, or it may produce something entirely unrelated, such as a construction site soundscape. It is up to the imagination of the artist, breaking down many technology barriers that would be present without the existence of Cambience.

Such installations may be done either privately (e.g. in a home), publicly (e.g. in a mall), or a hybrid divided by capture and presentation, or by computer.

7.4 Conclusion

In this thesis we saw that Cambience has relevance in a number of active research domains and unifies them in a novel way. We saw that Cambience is a complex program under the hood with multimedia, live visual programming, and networking capabilities. However, to the end user it offered a flexible, powerful, and yet elegant visual programming language with vast possibilities. Though we focused on its applications for sonifying video, it was clear that the system has potential for many other uses.

This project raised many questions, especially about the current paradigm of sound usage at the interface. My hope is that this project may serve as a first-step platform toward a better understanding and richer use of audio in standard computer applications. As I have experienced myself, inspiration arises when one is allowed to think about and experiment with possibilities that were difficult to conceive and realize in the past.

References

- Abel, M.J. (1990) *Experiences in an exploratory distributed organization*. In Galegher, Kraut & Egidio (Eds), *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, Lawrence Erlbaum Associates, 489-510.
- Alexanderson, P. (2004) Peripheral Awareness and Smooth Notification: the Use of Natural Sounds in Process Control Work. In *Proceedings of NordiCHI '04* (Tampere, Finland, October 23-27, 2004). ACM Press, New York, NY, 281-284.
- Beauchemin, S. S. and Barron, J. L. (1995). The computation of optical flow. *ACM Comput. Surv.* 27, 3 (Sep. 1995), 433-466.
- Besl, P. J. and Jain, R. C. (1985). Three-dimensional object recognition. *ACM Comput. Surv.* 17, 1 (Mar. 1985), 75-145.
- Bly, S., Minneman, S. (1990) Commune: a shared drawing surface. In *Proceedings of the Conference of Office Information Systems*, Cambridge, MA, April 1990. pp. 184-192.
- Bly, S., Harrison, S., Irwin, S. (1993) Media spaces: bringing people together in a video, audio, and computing environment. In *Communications of the ACM*, 36 (1), January, p. 28-46. ACM Press: New York.
- Boyle, M. (2005) *Privacy in Media Spaces*. PhD Thesis, Department of Computer Science, University of Calgary, Calgary, Alberta CANADA T2N 1N4. April 2005.
- Boyle, M. and Greenberg, S. (2005) The Language of Privacy: Learning from Video Media Space Analysis and Design. *ACM Transactions on Computer-Human Interaction (TOCHI)*. 12 (2), June, 328-370, ACM Press.

- Boyle, M. and Greenberg, S. (2005) Rapidly Prototyping Multimedia Groupware. Proceedings of the 11th Int'l Conference on Distributed Multimedia Systems (DMS'05), (Sep 5-7, Banff) Knowledge Systems Institute, IL, USA.
- Buxton, W., Gaver, W., Bly, S. (1989) The Use of Non-Speech Audio at the Interface. Tutorial Notes from CHI '89 (Austin Texas).
- Buxton, W. and Moran, T. (1990) EuroPARC's Integrated Interactive Intermedia facility (IIIF): Early Experiences. Proceedings of the IFIP WG8.4 Conference on Multi-user Interfaces and Applications, Heraklion, Crete, September 1990, pp. 24.
- Conversy, S. 1998. Wind and wave auditory icons for monitoring continuous processes. In *CHI 98 Conference Summary on Human Factors in Computing Systems* (Los Angeles, California, United States, April 18 - 23, 1998). CHI '98. ACM Press, New York, NY, 351-352.
- Cooper, D. 1995. Very Nervous System: Artist David Rokeby adds new meaning to the term interactive. In *Wired News* 3.03, March 1995. Retrieved March 2008 from http://www.wired.com/wired/archive/3.03/rokeby_pr.html.
- Dabbish, L. and Kraut, R. E. 2004. Controlling interruptions: awareness displays and social motivation for coordination. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work* (Chicago, Illinois, USA, November 06 - 10, 2004). CSCW '04. ACM Press, New York, NY, 182-191.
- Dourish, P. and Bly, S. (1992) Portholes: Supporting Awareness in a Distributed Workgroup, in *Proceedings of Human Factors in Computing Systems, CHI '92*, pp. 541-547. ACM Press: New York, 1992.

- Fels, S. 1997. Iamascope: an interactive kaleidoscope. In *ACM SIGGRAPH 97 Visual Proceedings: the Art and interdisciplinary Programs of SIGGRAPH '97* (Los Angeles, California, United States, August 03 - 08, 1997). L. Pocock, R. Hopkins, D. Ebert, and J. Crow, Eds. SIGGRAPH '97. ACM, New York, NY, 76-77.
- Gaver, W. W., Smith, R. B., and O'Shea, T. (1991) Effective sounds in complex systems: the ARKOLA simulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Reaching Through Technology* (New Orleans, Louisiana, United States, April 27 - May 02, 1991). S. P. Robertson, G. M. Olson, and J. S. Olson, Eds. CHI '91. ACM Press, New York, NY, 85-90.
- Gaver, W. W. 1993. Synthesizing auditory icons. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (Amsterdam, The Netherlands, April 24 - 29, 1993). CHI '93. ACM Press, New York, NY, 228-235.
- Goodman, G., Abel, M. (1986) *Collaboration research in SCL*. In Proceedings of the First Conference on Computer Supported Cooperative Work (CSCW), Austin, TX, December 1986.
- Gould, L., Finzer, W. (1984) *Programming by Rehearsal*. Xerox Palo Alto Research Center Technical Report SCL-84-1. May, 1984. 133 pages. Excerpted in *Byte*. 9(6) June, 1984.
- Greenberg, S. and Kuzuoka, H. (1999). Bootstrapping Intimate Collaborators. In OzCHI99 Workshop: Issues of Use in CSCW Technology Design (held as part of the OZCHI'99 Australian Conference on Computer Human Interaction). Organized by Robertson, T., Fitzpatrick, G. and Greenberg, S., November 27, Wagga Wagga Australia.

- Greenberg, S. and Roseman, M. (1999). Groupware Toolkits for Synchronous Work. In M. Beaudouin-Lafon, editor, *Computer-Supported Cooperative Work (Trends in Software 7)*, Chapter 6, p135-168, John Wiley & Sons Ltd, ISBN 0471 96736 X. 258pp.
- Greenberg, S. and Fitchett, C. (2001) Phidgets: Easy Development of Physical Interfaces through Physical Widgets. *Proceedings of the UIST 2001 14th Annual ACM Symposium on User Interface Software and Technology*, November 11-14, Orlando, Florida, p209-218, ACM Press. Includes video figure.
- Grudin, J. 1994. Groupware and social dynamics: eight challenges for developers. *Commun. ACM* 37, 1 (Jan. 1994), 92-105.
- Gutwin, C., Greenberg, S., Roseman, R. (1996) Supporting Awareness of Others in Groupware. A short paper suite, in *ACM SIGCHI'96 Conference on Human Factors in Computing System, Companion Proceedings*, p205-215.
- Halbert, D.C. (1984) An Example of Programming by Example. Masters of Science Thesis. Computer Science Division, Dept. of EE&CS, University of California, Berkely and Xerox Corporation Office Products Division, Palo Alto, CA. June, 1981.
- Hayes, G. R. and Abowd, G. D. 2006. Tensions in designing capture technologies for an evidence-based care community. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Montréal, Québec, Canada, April 22 - 27, 2006). R. Grinter, T. Rodden, P. Aoki, E. Cutrell, R. Jeffries, and G. Olson, Eds. CHI '06. ACM, New York, NY, 937-946.
- Hindus, D., Ackerman, M. S., Mainwaring, S., and Starr, B. 1996. Thunderwire: a field study of an audio-only media space. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work* (Boston, Massachusetts, United States, November 16 - 20, 1996). M. S. Ackerman, Ed. CSCW '96. ACM Press, New York, NY, 238-247.

- Ishii, H., Wisneski, C., Brave, S., Dahley, A., Gorbet, M., Ullmer, B., Yarin, P. (1998) ambientROOM: Integrating Ambient Media with Architectural Space. In Proceedings of CHI '98 (April 18-23, 1998). ACM Press, New York, NY, 173-174.
- Jancke, G., Venolia, G. D., Grudin, J., Cadiz, J. J., and Gupta, A. 2001. Linking public spaces: technical and social issues. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Seattle, Washington, United States). CHI '01. ACM, New York, NY, 530-537.
- Johnson, B. and Greenberg, S. (1999). Judging People's Availability for Interaction from Video Snapshots. Proceedings of the Hawaii International Conference on System Sciences, Distributed Group Support Systems Minitrack, January, IEEE Press.
- Kimura, T.D., Choi, J.W. and Mack, J.M. (1986) A Visual Language for Keyboardless Programming. Technical Report WUCS-86-6, Department of Computer Science, Washington University, St. Louis, June 1986.
- Kimura, T.D., Choi, J.W. and Mack, J.M. (1990) Show and Tell: A Visual Programming Language. Invited paper in Visual Computing Environments, E.P. Glinert (ed.), IEEE Computer Society Press Tutorial, Washington, D.C., 1990, pp. 397-404.
- Kraut, R., Fish, R., Root, R. and Chalfonte, B. (1990) Informal Communication in Organizations. In *People's Reactions to technology in Factories, Offices and Aerospace*, S. Oskamp and S. Spacapan, (Eds.). pp. 145-199. Sage Publications: New York.
- Lee, A., Girgensohn, A., and Schlueter, K. 1997. NYNEX portholes: initial user reactions and redesign implications. In *Proceedings of the international ACM SIGGROUP Conference on Supporting Group Work: the integration Challenge* (Phoenix, Arizona, United States, November 16 - 19, 1997). GROUP '97. ACM, New York, NY, 385-394.

- Mantei, M., Baeker, R., Sellen, A., Buxton, W., Milligan, T. and Wellman, B. (1991) *Experiences in the Use of a Media Space*, Proceedings of CHI '91 Human Factors in Computer Systems, New Orleans, Louisiana.
- Marquardt, N. and Greenberg, S. (2007) Distributed Physical Interfaces with Shared Phidgets. Proc. 1st International Conference on Tangible and Embedded Interaction. (Feb 15-17, Baton Rouge, Louisiana, USA).
- McEwan, G. and Greenberg, S. 2005. Supporting social worlds with the community bar. In *Proceedings of the 2005 international ACM SIGGROUP Conference on Supporting Group Work* (Sanibel Island, Florida, USA, November 06 - 09, 2005). GROUP '05. ACM, New York, NY, 21-30.
- McEwan, G. (2006) Community Bar: Designing for Informal Awareness and Casual Interaction. MSc Thesis, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada T2N 1N4. September. Supervisor: Saul Greenberg.
- Myers, B. A. 1986. Visual programming, programming by example, and program visualization: a taxonomy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, Massachusetts, United States, April 13 - 17, 1986). M. Mantei and P. Orbeton, Eds. CHI '86. ACM Press, New York, NY, 59-66.
- Mynatt, E. D., Back, M., Want, R., Baer, M., and Ellis, J. B. (1998) Designing audio aura. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Los Angeles, California, United States, April 18 - 23, 1998). C. Karat, A. Lund, J. Coutaz, and J. Karat, Eds. Conference on Human Factors in Computing Systems. ACM Press/Addison-Wesley Publishing Co., New York, NY, 566-573.
- Neustaedter, C., Greenberg, S. and Boyle, M. (2006). Blur Filtration Fails to Preserve Privacy for Home-Based Video Conferencing. *ACM Transactions on Computer Human Interactions (TOCHI)*, 13, 1, March, p1-36.

- Patterson, J. F., Day, M., and Kucan, J. (1996). Notification servers for synchronous groupware. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work* (Boston, Massachusetts, United States, November 16 - 20, 1996). M. S. Ackerman, Ed. CSCW '96. ACM Press, New York, NY, 122-129.
- Pedersen, E. R. and Sokoler, T. 1997. AROMA: abstract representation of presence supporting mutual awareness. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, United States, March 22 - 27, 1997). S. Pemberton, Ed. CHI '97. ACM Press, New York, NY, 51-58.
- Rodriguez, M. D. and Shah, M. (2007). Detecting and segmenting humans in crowded scenes. In *Proceedings of the 15th international Conference on Multimedia* (Augsburg, Germany, September 25 - 29, 2007). MULTIMEDIA '07. ACM, New York, NY, 353-356.
- Rokeby, D. (1986-1990) Very Nervous System.
<http://homepage.mac.com/davidrokeby/vns.html>. Posted in 2000, accessed September 2007. A video of VNS is published as:
 Rokeby, D. (1988) Very Nervous System. ACM SIGGRAPH Video Review, Issue 40, 1988.
- Rounding, M. (2004) Informal Awareness and Casual Interaction with the Notification Collage. MSc Thesis, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, April. Supervisor: Saul Greenberg.
- Smith, D. C. (1977) *Pygmalion: A Computer Program to Model and Stimulate Creative Thought*. Basel, Stuttgart: Birkhauser, 1977.
- Smith, I. and Hudson, S. E. 1995. Low disturbance audio for awareness and privacy in media space applications. In *Proceedings of the Third ACM international Conference on Multimedia* (San Francisco, California, United States, November 05 - 09, 1995). MULTIMEDIA '95. ACM Press, New York, NY, 91-97.

- Stults, R. (1986) Media space. Xerox PARC technical report, pp 20.
- Stults, R. (1988) Experimental uses of video to support design activities. Xerox PARC technical report SSL 89-19.
- Tang, J. and Minneman, S. (1990) VideoDraw: a video interface for collaborative drawing. Proceedings of CHI '90, pp. 313-320.
- Tang, J. C. and Rua, M. 1994. Montage: providing teleproximity for distributed groups. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Celebrating interdependence* (Boston, Massachusetts, United States, April 24 - 28, 1994). B. Adelson, S. Dumais, and J. Olson, Eds. CHI '94. ACM Press, New York, NY, 37-43.
- Watts, J. C., Woods, D. D., Corban, J. M., Patterson, E. S., Kerr, R. L., and Hicks, L. C. 1996. Voice loops as cooperative aids in space shuttle mission control. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work* (Boston, Massachusetts, United States, November 16 - 20, 1996). M. S. Ackerman, Ed. CSCW '96. ACM, New York, NY, 48-56.
- Wittaker, S., Frolich, D. and Daly-Jones, W. (1994) Informal Workplace Communication: What is it Like and How Might we Support It?, in Proceedings of Human Factors in Computing Systems, CHI '94, ACM Press: New York.