4 The Implementation of *Talking in Circles*

> *Sociofugal space is not necessarily bad, nor is sociopetal space universally good. What is desirable is flexibility and congruence between design and function so that there is a variety of spaces, and people can be involved or not, as the occasion and mood demand* – Edward T. Hall

4.1 Design Requirements and Iterative Prototypes

The requirements for *Talking in Circles* focused on full-duplex audioconferencing between a substantial number of simultaneous users, where we defined substantial as approximately 15 Figure 4.1 shows a typical desktop setup for use with the system, including a microphone and headphones to prevent feedback from speakers. Even experimentally, we were interested in low-latency audio. Unlike with pre-recorded streaming, in live audioconferencing it's not possible to pre-buffer or pause when there is network congestion. Lag is known to be detrimental to the use of speech for social interaction, for example leading to greater formality [O'Conaill 1993]. However, we were also interested in creating a system that could have as wide a user base as possible, important given our focus on social applications as well as to facilitate wider, extended study of the system's use. This meant that we could not use proprietary broadband networks or high-speed LAN's, as previous systems have typically done.

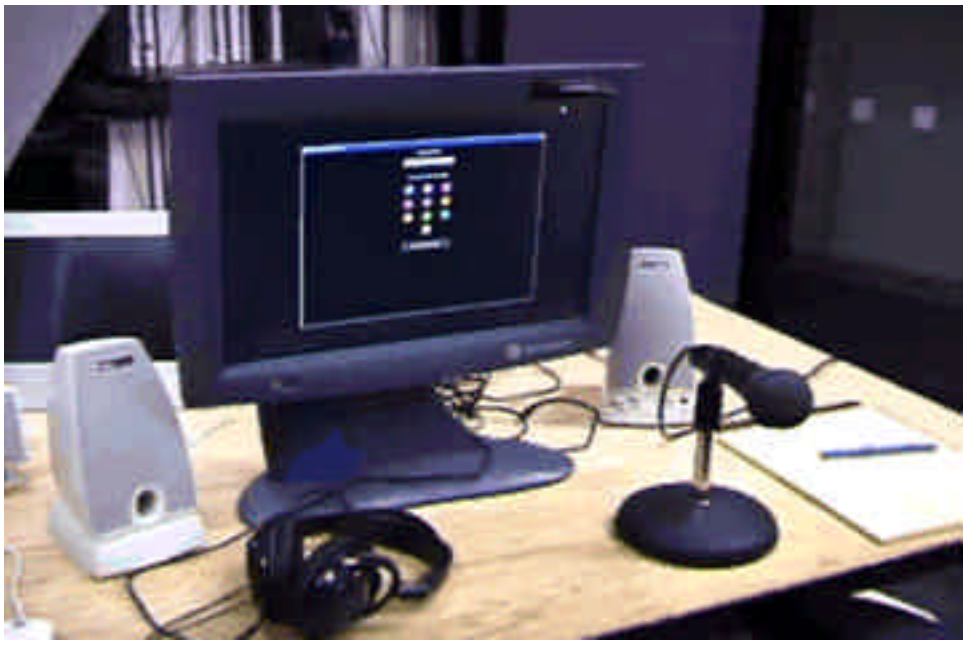


Figure 4.1: Typical desktop setup for a *Talking in Circles* session

The initial aim was to design for the internet as a whole, but this proved intractable, as even highly compressed protocols such as RealAudio occasionally suffer from unpredictable network delays and must pause to rebuffer [Progressive]. A prototype was then attempted using the Java Sound API [Java Sound], but resulted in measured end-to-end lag of two to three seconds for machines on the same high-speed LAN subnet. Finally the *RAT* package was taken as a basis and modified for use with *Talking in Circles*. *RAT* is the *Robust Audio Tool*, an open-source audioconferencing tool from University College, London [UCL]. *RAT* uses the MBONE, the internet's multicast backbone, for network transport [Savetz 1996]. Thus we avoided inefficient strictly client-server and peer-to-peer architectures. A client-server architecture, adding an extra hop and centralized processing between each transmission, does not scale when there is a large number of clients or when these clients are geographically distributed. Peer-to-peer connections also require *O(n)* work for sending out the audio to each participant. Multicast, by contrast, allows each client to transmit its audio only once and then replicates it to the other clients, resulting in a fully-connected graph architecture where the outgoing edges need be traversed only once for constant *O(1)* transmission cost.

RAT was adapted to support communication of participant state (x/y location, circle color, instantaneous audio energy, and so on) and end-to-end lag was then measured at approximately 0.5 seconds, considerable but not detrimental unless participants can also hear each other directly

[Krauss 1967]. The bottlenecks in our current implementation are local ones, primarily high-frequency reading of the microphone, due to the Windows Task Manger's coarse threading, as well as screen redraw, and accordingly we have noticed no substantial performance degradation when varying the number of users from one to eleven. An additional advantage of using multicast throughout for communication is that data besides audio, particularly drawing, are also transmitted to all participants with very low latency, which enables participants to observe and discuss drawing as each pixel or stroke is added.

Although the audio code, including compression/ decompression and MBONE transport, is written in C, the user interaction portion is maintained in Java using the Java Native Interface in Sun's Java 2 platform. For example, computations including instantaneous audio energy, background noise suppression and logarithmic normalization to map the energy value onto the circle's area are performed in C, communicated to the *Talking in Circles* interface via the Java Invocation API, and the bright inner circle (see Figure 3.4) is then updated several times per second in the Java component. Lag was a problem with Java's mouse-motion reporting during freehand drawing, which was adequately resolved using Bresenham's line-interpolation algorithm. A port of the system was made to Linux, but the Java Native Interface was not well-developed on Linux and thus the system did not run with full capabilities; performance of the basic system, however, was far superior to the original Windows version. Audio bandwidth use is moderate at 5KB/s per client, too high for scalable use with modems but quite appropriate for use on various forms of broadband network connections. Ongoing and future developments —including the spread of always-on, high-bandwidth internet access; the deployment of networking protocols such as Ipv6 and RSVP, which can provide for bandwidth guarantees for data requiring low-latency transmission; and the development of new data-compression algorithms— promise to make real-time, multimodal social environments such as *Talking in Circles* useable at high quality by anyone on the open Internet perhaps within two years.

4.2 Shared and Subjective System Rendering

By way of summarizing, this section notes the various data elements of the system as produced by the conversants and how these are rendered for each. Figure 4.2 shows the path of these data through the system.

Each participant directly multicasts the following:

- circle x/y coordinates
- freehand drawing
- icon selection
- instantaneous audio energy
- speech/audio

The interface is rendered from these features, and all participants' displays share identical views of:

- circles' location
- circles' drawing/icon display
- participants' audio

Finally, the local user's relative location produces a subjective rendering of:

- speech volume (audio)
- speech rendering (graphics)

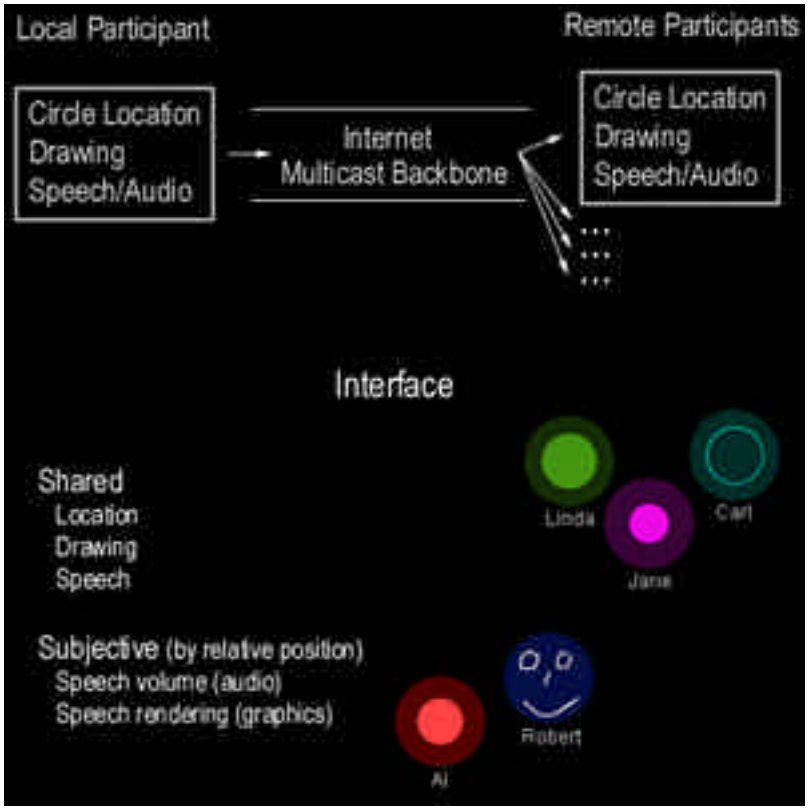


Figure 4.2: ***Talking in Circles*** **system diagram showing an overview of network transport of data and its rendering for participants**

4.3 Basic *Talking in Circles* Graphical Elements

Figure 4.3 shows the *Talking in Circles* login panel. Users enter their name, then choose a color for their circle and join the session. Circle colors were hand-designed for equal perceived brightness, to minimize potential biasing effects of, for example, certain colors seeming more "happy" or "sad" than others. While color design is a complex perceptual problem, Jacobson and Bender note that color perception, while subjective, is in many ways free of cultural and individual influences, and that "predictable visual sensations can be elicited by adjusting the relationships among colors" [Jacobson 1996]. The rest of this section's figures show the other basic graphical elements used in *Talking in Circles*.

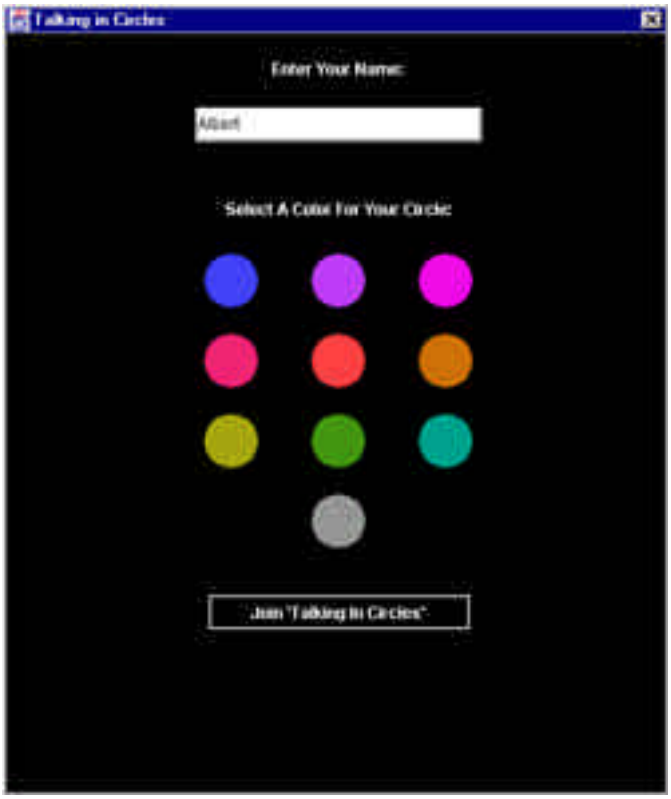
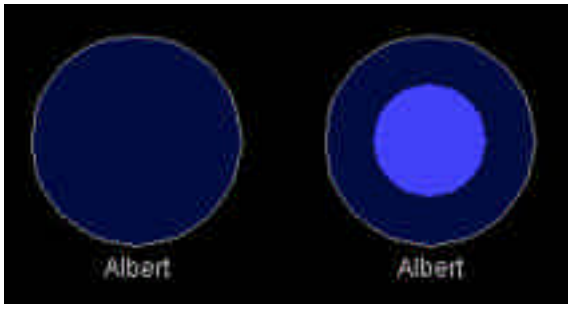


Figure 4.3: *Talking in Circles* login panel.

Figures 4.4a, 4.4b: The music and sound booths.

Figures 4.5a, 4.5b: Built-in icon bar and how each appears when displayed by a participant.



Figures 4.6a, 4.6b: Albert's circle as it appears in *Talking in Circles* when not speaking (left) and when speaking at normal volume (right).

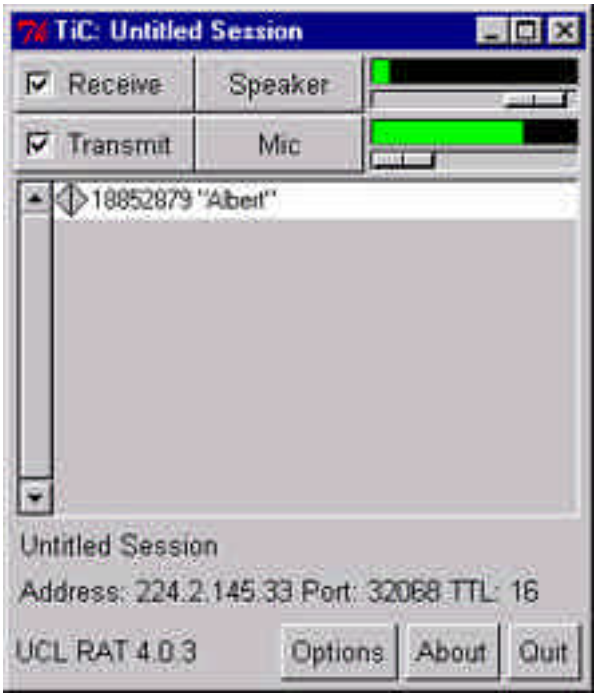


Figure 4.7: *RAT* control panel. Though not generally used during a *Talking in Circles* session, this control panel permits adjustment of the local microphone and speaker amplification.