ANTECHAMBER SOUND

Brian Karr and William Bricken
April 1993

Brian's Proposal

Antechamber designers, builders and users: here are some thoughts on audio for the antechamber. The intent here is to start some dialog around the AUI (owie) for the Antechamber.

- o -

We are free to use plain stereo, 'pre-spatialized', or real-time spatilized sound here and in some cases we may wish to use combinations. By pre-spatialized, I mean that a 3D sound sequence can be recorded beforehand and played back later as a stereo sound with the 3D image intact. We can use effects such as reverberation on any of the above.

Here are some ideas to chew on for how and when to use audio.

-- Cues and messages should be short and simple and the participant should not be required to refer to them later.

-- The messages deal with events in time, such as when a new participant has entered the chamber, when a world is loaded and ready, confirmation of a selection or deselection, etc.

-- Warning and help tones.

-- When the message calls for immediate action. A 'virtual pager' mechanism may indicate that someone wishes to join them.

-- When continuously changing info is presented. This is useful for presenting the 'opportunity density' (term coined by cocteau) at any moment for the participant. An example of this is to indicate how far down a world hierarchy the user is by increasingly muffling an iconic tone as they go down the tree of worlds.

-- When speech channels are fully employed, when a verbal response is required, a la speech synthesis and recognition.

-- When the use of vision is not sufficient. The visual frame rate may fall or the image may not update at all during transitions. In this case audio can be used to reassure the user that the system isn't hung and give them contextual information about the world they are about to enter. The context audio can be identified as such through the world building API (see below).

-- When the receiver moves from one place to another. The participant shouldn't need to have to look at objects, such as other participants, to gather information from them.

-- The participant should be able to choose and change how much audio feedback they receive.

-- Consistency that can be taken into VE's for message passing. Certain reserved tones may be set up for use in the Antechamber only. These tones, such as the 'virtual pager' tone are them immediately recognized as an Antechamber message to the participant, regardless of which world they are in.

We may wish to have a locator audio stream coming from, say, the entrance to the antechamber to help overcome problems with the narrow FOV HMDs. This also gives serves as a reference to a location the participant is already familiar with.

We may want to set up an 'audio API' for world builders. The builder can associated a custom earcon for themselves in a standard way, for the 'portal' (or whatever we use) to the world from the antechamber. A stream of text can be delineated as transition text, to be spoken from a DecTalk for instance, that is given to the listener during Antechamber-world and world-world transitions.

- o -

Comments on the Antechamber Document (William)

The focus topic is, of course, excellent. And the idea of a lab software project is great, and perhaps can replace the DEMO mode.

First some global (super-chamber) observations:

My guess is that the effort level for the project is six person months, with not much chance of compressing the time by involving extra people. These kind of programming projects usually center on one or two people doing 90% of the coding, and that is what I expect here too.

Group projects are a source of joy, but there are some external constraints that the project should at least recognize. I'll order them in terms of importance:

-- Folks are scheduled to complete a thesis during the next few months. In my opinion, it is a fatal error for them to get involved in any other project work. The lab has consciously screened them from project involvement

so that they can finish their thesis work. (90% of failing theses happen during the last 10% of work. There are tremendous deceptive pressures toward loss of focus.)

-- We may be heading toward a project structure, which means that resources for this kind of spontaneous group project will change. Be sure to scope to _finish_ in two months or get affiliated with a funding source. The resource allocation exercise for this project is a good idea regardless.

-- The subprojects are DIFFICULT. Each should be a thesis. Yes, even when SIMPLE. Seems to me the primary source of difficulty is not to get it running, but to meet the needs of generality, robustness, and structured coding. Only a couple of folks in the lab have the previous training to do structured programming (the XLISP code shows it). [Please understand, the issue here is _how much effort_ and training it will take to achieve the simple coding objectives, not our ability to do it.]

ENVIRONMENT

The Antechamber manager is making excellent management decisions about simplicity and about leadership. He is also expecting performance beyond training.

Of course, I like the Environment/Space/Participant partition. Since Space is the substrate of an Environment, I might suggest Context or World Design or Interface instead of Environment.

The management of multiple worlds ("Space") is conceptually independent of the "Environment" and transparent to it. I believe that Environmental Design (what kind of portal...) is currently experimental, which implies either 1) many designs or 2) iterative design. Both actually. This means that the effort for Environment is substantively different than that of Space; the first seeks flexibility and modification and customizability, the second seeks solid singular efficient non-muckable code.

Standard interface protocols (world <==> participant) will be implementation, language, and architecture dependent, so "standard" is not a good word. Perhaps "transparent" (implying that the interface talks to many different mechanisms) or "integrated" (implying that the World and Participant have the same substrate).

IMHO, the first environments should be cubelands, followed by Metro and Needle.

Environmentalists: Do not consider "common portals" (ie doors) as a good idea. Revisit Virtual Seattle and the Division and Autodesk worlds. Look at how others do it. A few alternatives:

-- nomination ("put me there")
-- dissolves (morph this environment into that)
-- scalings (shrink to find)
-- seamless traversal (feels like one world)
-- turn arounds (the other world is behind you)
-- charms (hold this to go there)
-- sonics (tinkle to travel) and
-- out-of-body (change selves)

I disagree that portals should be literal rather than metaphoric. They should be what works well. We simply don't know what will work well yet.

This leads me to suspect that the Environment subproject is premature and needs to consider objectives. Note that none of the above options suggest that the Space Manager be different; they do suggest that the Space Manager be general. Note that some options have an "antechamber" feel, some are not spatial/physical, some can be location-specific to take effect, some require modes. I'll bet that data structures and control regimes will interact with possible methods of portaling.

If the Environment group is just going to build a hallway-world, then this is a simple, one-person job. We should not (IMHO again and as usual) try to build compelling structures like better hallways, that is not where is action is.

So, I get the feeling that this subproject is kind of upside-down. This group needs to specify what it needs from Space, what functionalities are required for inter-world navigation, what database architectures and transformations.

Docking and orientation issues are critical. We must address where and how a participant enters a world, including the antechamber itself.

Does the antechamber know the physiological model of the participant? Does it know the sensing suite? In my mind, the antechamber is the place you "get into" your virtual body, prior to going to a world. It is where calibration, intentional filtering, and self-image are established. Ideally, it may be an indistinguishable overlay on the physical, which gradually fades into immersion in the virtual. We currently ignore the Gestalt shuttling effects we create by tossing participants directly into and out of new environments (this effect has been shown to have traumatic potential).

This is important enough to say again: the antechamber is not only for traveling between worlds, it has the essential function of providing a smooth transition from physical to virtual.

DESIGN DICTATE: prototyping emphasizes CONCEPT first, implementation second.

Andy is advocating stubbing or postponing complex subprojects, a deeply correct attitude. But the entire conceptual structure needs to be in place first for this to work. Or if the objective is a neat hardwired demo that will be scraped...

[POLITICAL NOTE: every week now people tell me that our demos are not at all impressive. About the same as you can get for a buck at the local arcade. We all know the value of giving good demo. We might even have pride enough to expect ourselves to put on a state-of-the-art demo. I'd think that the antechamber project would have this in mind. In the current example, common portals will bring us up to the capabilities we demoed in 4/90 for CHI. Incidentally, the VPL demo of that era permitted objects to be carried across worlds. Come to think of it, even the Autodesk demo of 6/89 permitted objects to cross portal boundaries. I am making a point of scholarship: design an *improvement*.]

SPACE

This is the essence of the project. Merging pools is the essence of the divergent world concept. Yes, let's assume the pools will be consistent, so no contradictory structures. Pool merging should handle:

```
--  private/public partitions for individual entities,
--  entity/environment partitions
--  multiple participants
--  multiple worlds
--  physical/virtual partitions of resources
```

[Multiple worlds: whatever that means. There is only *one virtual world*, but lots of access routes. The master space concept recognizes this.]

I'd put distributed management of partitions on the later stack.

The idea of not running display characteristics of a world when no one is there is good, but in general, autonomous processes will care about specific aspects of the world even when no human participant is present. Slaving world on-off to participants is parochial. More generally, aspects which no entity (human or non) perceives do not happen, that is because *events are relationships*. I hope Absolutism is no creeping into our conceptualizations.

```
Event[world, participant] = experience
Event[world] = database change            [=/= nothing]
Event[] = void
```

There is a BIG IDEA here: make no design decisions based on implementation efficiency. Instead, build the prototype for conceptual integrity, then unroll, cut back, and optimize to get performance. You know, when we buy

that fast machine (which may be cheap next year), those implementation hacks will be irrelevant.

I like the proposed functionality of the SM.

PARTICIPANT

Inter-participant coordination should not be different that inter-entity coordination. Changes that effect only one participant _do not exist_ unless contradictions are incorporated. Eg: scaling, if a participant changes scale, others see that participant change scale. Yes, the display pipeline to a participant is individual, but this pipeline _is not in the world_, it is in the physical perceptual apparatus of the person.

But yes, one person's scale change cannot effect the world, this is a simple matter of scope of control. Don't provide global access to individual entities. (There are exotic configurations where this gets mucky. If you as a participant are also an environment...)

The Antechamber manager talks about code separation and cleanliness. There is stickiness here: most of us don't know how to achieve this. New students won't know either. Basically, STRUCTURED PROGRAMMING should be enforced by the SM, it is the language syntax arbitrator. The problem is, IMHO, far deeper than multi-userizing current code. Current LISP code that I have looked at ignores binding regimes, namespace packages, abstraction barriers, data-structure dependency, logical variables, wrappers (ensuring closure), and has bad habits of file loading (rather than function call loading), global variables, setqs rather than lets, etc.

It takes about a year of concentrated training to learn how to achieve modularity. [And don't trip off about C++, it requires more expertise than LISP to get right.]

Actually, I have a difficult time understanding what is unique in the participant subproject. Toss it out.

SUMMARY

The SM is the core of this project. Other subprojects feel like over-engineering. If SIMPLICITY is the goal, then I recommend a working group of three for the whole thing. One project, one team, one goal.