

## Unity を用いた空間音会議インターフェイスの開発

### Unity-developed interface for spatial sound conferencing featuring narrowcasting and multipresence with network control

○児島弘将\*, 公園 マイケル\*\*

○Hiromasa Kojima\*, Michael Cohen\*\*

会津大学コンピュータ芸術学講座

Spatial Media Group, Computer Arts Lab., University of Aizu

キーワード : インターフェイス (interface), 空間音響 (spatial sound), 特定少数向け情報配送 (narrowcast), 多数の存在 (multipresence), 自動焦点 (autofocus)

連絡先 : 〒 965-8580 福島県 会津若松市  
会津大学 コンピュータ芸術学講座

E-mail: s1210235, mcohen@u-aizu.ac.jp

## Abstract

We are exploring groupware interfaces implementing narrowcasting. Narrowcasting interface is realized by the functions of *Solo*, *Mute*, *Attend*, & *Deafen*. Utilizing Unity for cross-platform deployment and CVE (Collaborative Virtual Environment) developed by the Spatial Media Group enables the interface to exchange information with various platforms and devices.

## 1. Introduction

The Spatial Media Group at the University of Aizu is researching user interfaces to control source → sink transmissions in synchronous groupware. We developed Multiplicity figurative narrowcasting interface with Java3D, but it is outdated. So we transition from Java3D to Unity

(integrated development environment) and improve it as conferencing interface. Conferencing interface can support cross-platform deployment, including smartphones.

### 1.1 Background

#### 1.1.1 Related Research

Audio narrowcasting interface on workstations and mobile phone was developed 10 years ago by the Spatial Media Group <sup>2)</sup>. Interface on workstations was made with Java3D <sup>1)</sup> and needs to be started at the terminal. Another interface on mobile phone was made with J2ME (Java ME) and only works on iϕpli mobile phones. In either case, people using such mobile phones and workstations are very few. So we explore more modern, cross-platform alternatives.

### 1.1.2 Unity

Unity is a cross-platform software framework. It is mainly used for game development, but can be used for research and development. Supported platforms include PCs (Windows, MacOS, Linux), consoles (Playstation 4, Xbox 360, Nintendo 3DS, etc.), mobile devices (Android, iOS, etc.), HMDs, and websites. If we develop with Unity, we can develop software that can be deployed on various platforms.

### 1.1.3 CVE

“CVE” is a initialization for “Collaborative Virtual Environment”, a protocol developed by the Spatial Media group (Fig. 1). This is based on client-server architecture for groupware applications and supports to connect various devices synchronously. We used it to develop communication between cooperations clients for groupware.

### 1.1.4 Narrowcasting

Conventional conferencing interface shares voice of each participating user to all users using that interface. But there is a demand to select a specific person or group and exchange media. Narrowcasting meets that demand and controls exchange of media according to articulated intentions of users.

### 1.1.5 Formalization of Narrowcasting

Traditional audio sources interfaces employ **Mute** and **Solo** functions which selectively enabling and disabling respective channels. But our interface has not only sources, but also sinks, motivate the generalization of **Mute** & **Solo** to exclude and include, manifested for

sinks as **deafen** & **attend** as elaborated by Fig. 5

### 1.1.6 Multipresence

Humans are not divisible and cannot exist in more than one place simultaneously. But our unique interface has function to let multiple avatars represent distributed presence of a user at the same time. Multipresence arises from sources and sinks in various place around the same time. By means of this, we can hear the sound in two or more places in virtual space.

### 1.1.7 Autofocus

For spatial sound narrowcasting, sources must determine which sink to share media with. That is decided by autofocus. This time, standards for judgment of autofocus is the distance between source and sink. The shorter the distance, the higher priority of autofocus.

## 2. Implementation

We used MacOS and Unity to develop Narrowcasting interface. It is written in the C# programming language. There are six color-coded channels on the interface. On that, operation toggles for each channel are placed on the UI panel and avatars for each channel are arranged in 3D space. The screenshot of interface is shown in Fig. 4.

### 2.1 Operation toggles

Avatar on the interface becomes Sink (listener) by turning on the Self toggle and Source (speaker) by turning off the that toggle. When an avatar is Self, that avatar works as user’s pairs of ears in 3D virtual space. By setting

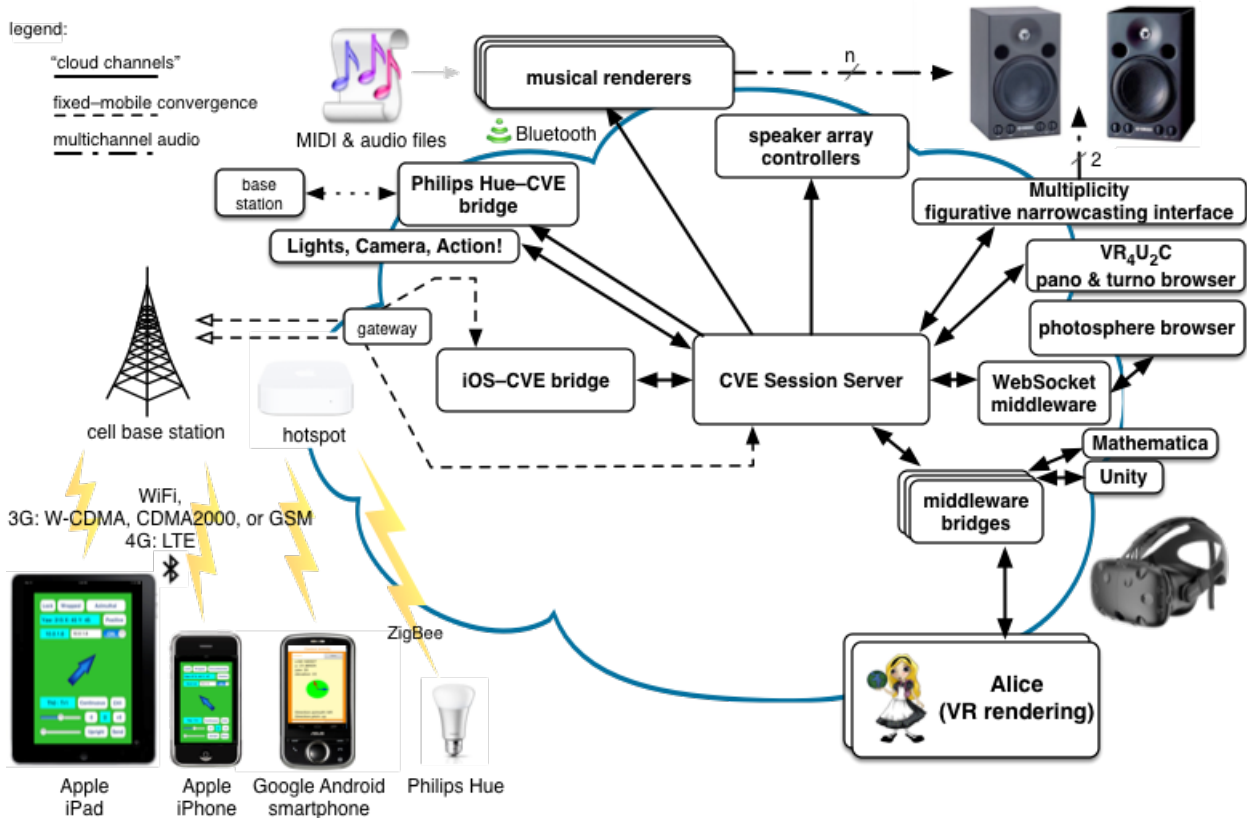


Fig. 1 CVE architecture

two or more sinks, there are two or more pairs of in that space. This is a multipresence. User can listen to all the sounds that each avatar is listening to. The Function of Mute is to turn off the sound of source. On the contrary, the function of Solo toggle turns off complement of selection. When Deafen toggle is on, The role of Sink of that avatar is disabled. That avatar don't hear the sounds in that space. Sources autofocus to Sink other than Deafen. Attend's role is Users can only hear sound that the avatar is attending. Formalization shown is Fig. 2 describes for these implementations. Also, the avatar's action when toggle is switched is shown in Fig. 3.

## 2.2 Spatial sound

Unity includes bundled function of spatial sound. Unity calculates the output of the spa-

tial sound from the placement of the avatars that produces sound.

## 2.3 Movement of avatars

Avatars change position and rotation according to user's input and networked control. Each time, spatial soundscape is recompiled with narrowcasting attributes and autofocus policy.

## 2.4 Sound map

This interface has the ability to display ego-centric spatial sound map, including compilation of narrowcasting state and multipresence. The map is displayed by pressing the shift key and shown in Fig. 5.

The general expression of two-tier activation is

$$\begin{aligned} \text{active}(\text{object}_x) &= \neg \text{exclude}(\text{object}_x) \wedge \\ &\quad (\exists y (\text{include}(\text{object}_y) \wedge (\text{self}(\text{object}_y) \Leftrightarrow \text{self}(\text{object}_x)))) \\ &\Rightarrow \text{include}(\text{object}_x). \end{aligned}$$

A channel is active unless it has been explicitly disabled or a relevant peer has been focused upon to the exclusion of the respective source under consideration. So, for `mute` and `select (solo)`, the source relation is

$$\begin{aligned} \text{active}(\text{source}_x) &= \neg \text{mute}(\text{source}_x) \wedge \\ &\quad (\exists y (\text{select}(\text{source}_y) \wedge (\text{self}(\text{source}_y) \Leftrightarrow \text{self}(\text{source}_x)))) \\ &\Rightarrow \text{select}(\text{source}_x). \end{aligned}$$

For `deafen` and `attend`, the sink relation is

$$\begin{aligned} \text{active}(\text{sink}_x) &= \neg \text{deafen}(\text{sink}_x) \wedge \\ &\quad (\exists y \text{attend}(\text{sink}_y) \wedge (\text{self}(\text{sink}_y) \Leftrightarrow \text{self}(\text{sink}_x))) \\ &\Rightarrow \text{attend}(\text{sink}_x). \end{aligned}$$

Fig. 2 Formalization of two-level narrowcasting and selection functions in predicate calculus notation, where ‘ $\neg$ ’ means “not,” ‘ $\wedge$ ’ means conjunction (logical “and”), ‘ $\exists$ ’ means “there exists,” ‘ $\Rightarrow$ ’ means “implies,” and ‘ $\Leftrightarrow$ ’ means mutual implication (equivalence). The suite of inclusion and exclusion narrowcast commands for sources and sinks are like analogs of burning and dodging (shading) in photographic processing. The duality between source and sink operations is strong, and the semantics are analogous: an object is inclusively enabled by default unless, a) it is explicitly excluded with `mute` (for sources) or `deafen` (for sinks), or, b) peers are explicitly included with `select (solo)` (for sources) or `attend` (for sinks) when the respective object is not. Because a source or sink is active by default, invoking `exclude` and `include` operations simultaneously on an object results in its being disabled.


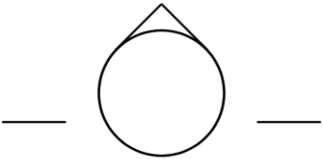

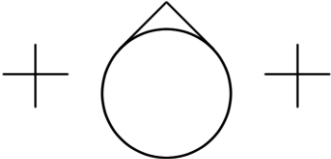
	<b>Source</b>	<b>Sink</b>
<b>Direction</b>	Output	Sink
<b>Instance</b>	Speaker	Listener
<b>Organ</b>	Mouth	Ear
<b>Exclude</b>	Mute	Deafen
<b>Inhibit in Multiplicity</b>		
<b>Include</b>	Solo	Attend
<b>Assert in Multiplicity</b>		

Fig. 3 Sources and Sinks

## 2.5 Network control

Thanks to CVE, this interface can exchange information such as location, **Solo**, **Mute**, **Attend**, and **Deafen** with other devices (iOS, Android, HMD).

## 3. Conclusion

The prototype development of the interface has gone well, but there are issues.

## 4. Future work

### 4.1 Implement “Attend”

It took time to programming for connecting Unity and CVE-server.”Attend” is very important for conferencing.

### 4.2 Build on each platform

It works on Unity. But we didn’t build on other platforms. At first, we targeted to develop groupware that runs within Unity on MacOS. we need a preparation of build. We plan to test on iOS.

### 4.3 Realtime audio streams

We are using music instead of voice now. Realtime audio stream is indispensable for spatial sound conferencing, so we hope to eventually handle realtime audio streams as well.

## Acknowledgement

I would like to offer my special thanks to the Spatial Media Group. They gave me lots of advice and help in my study.

## 参考文献

- 1) Cohen, Michael. “Exclude and include for audio sources and sinks: Analogs of mute & solo are deafen & attend.” *Presence* 9.1 (2000): 84-96.
- 2) Fernando, Owen Noel Newton, et al. “Audio narrowcasting and privacy for multipresent avatars on workstations and mobile phones.” *IEICE Trans. on information and systems* 89.1 (2006): 73-87.

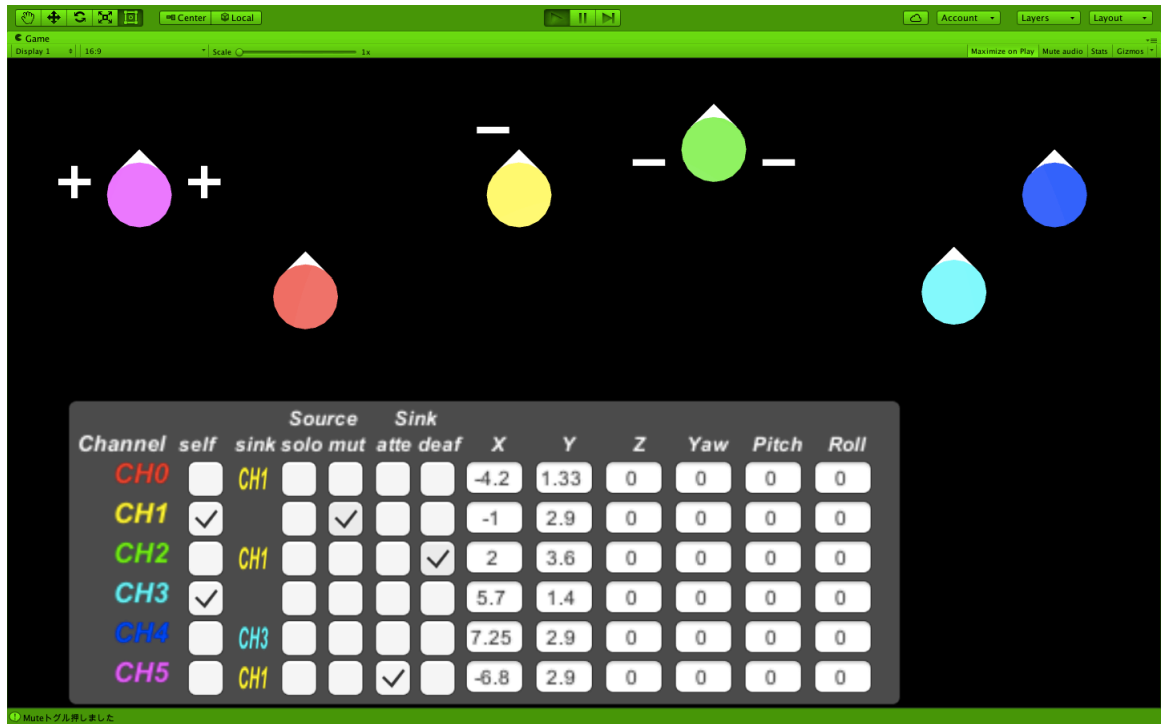


Fig. 4 Unity-developed narrowcasting interface

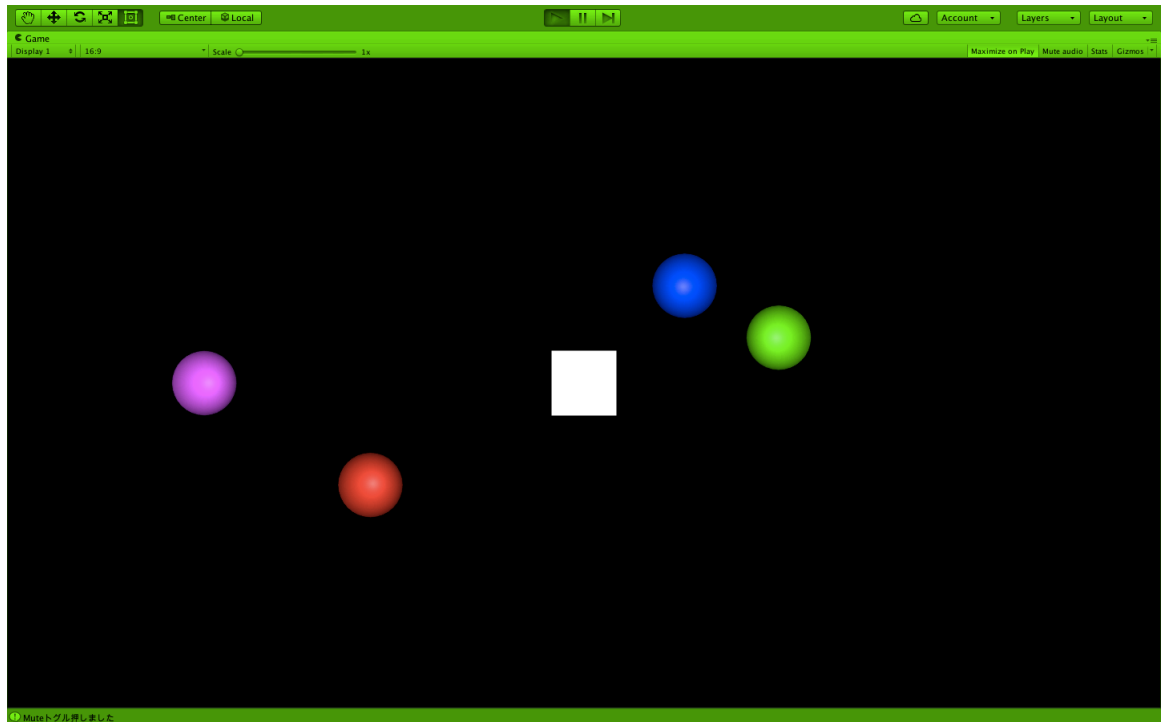


Fig. 5 Egocentric spatial sound map, including compilation of narrowcasting state and multi-presence