Narrowcasting visualization using particles for spatial sound conferencing

Koki Tsuda,[1] *Michael Cohen,*[1] *& Rintarō Satō*[1]

[1] Spatial Media Group, Computer Arts Lab., University of Aizu; Aizu-Wakamatsu, Fukushima 965-8580; Japan

Abstract. The goal of this study is to visualize duplex auditory communication filtered with narrowcasting. Narrowcasting is technology that enables media stream filtering for privacy. Audio narrowcasting has four operations (solo & mute and attend & deafen). In this research, communication from source to sink is determined according to these four operations, including "autofocus" determination of multipresent sinks, and visualized using flying particles. Soundscapes are compiled according to narrowcasting attributes. With such multimodal representation, users can more easily understand narrowcasting communication.

1 Introduction

1.1 Related Research

An audio narrowcasting interface was developed by the Spatial Media Group [1]. We can use a CVE (Collaborative Virtual Environment) to connect heterogeneous clients. We can connect our phone to Unity game engine, control player directions, and audition (listen to) spatial sound.

1.2 Narrowcasting

Narrowcasting is technology in communication applications such as chat-spaces or conferences that enables media stream filtering. It extends broadcast and multicast systems with articulated control. For audio communication, a source is a speaker, such as a mouth. A sink is listener, represented by ears. Narrowcasting has four attributes: solo & mute (for sources) and attend & deafen (for sinks).

1.3 Multipresence

Multipresence means having (virtual) presence in multiple places at once. In our application, a user designates multiple avatar icons as "self" to realize multipresence.

1.4 Autofocus

Multipresence introduces some ambiguity, as sources can be in the presence of multiple self-designated sinks. Each source must determine which single sink can hear its sound. Such determination is called "autofocus." For our simplified model, sources are considered omnidirectional, and orientation is not considered for estimation of loudness. Since intensity falls off with distance, active sources are each autofocused to their closest sink.

1.5 Narrowcasting predicate calculus formalization

The general expression of narrowcasting activation is

```
active(x) = ¬exclude(x) ∧
     ((∃y (include(y) ∧ (self(x)⇔self(y))))
     ⇒ include(x)),
```

where '¬' means "not," '∃' means "there exists," '∧' means "and," and '⇔' means equivalence.

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

```
active(source x) = ¬mute(x) ∧
       ((∃y (select(y) ∧ (self(x)⇔self(y))))
       ⇒ select(x)).
```

For deafen and attend, the sink relation is

```
active(sink x) = ¬deafen(x) ∧
       ((∃y (attend(y) ∧ (self(x)⇔self(y))))
       ⇒ attend(x)).
```

1.6 Unity

Unity is a game engine for multiple platforms that can manage development tools, 2D/3D graphics, sound reproduction, and user interface management. It can be programmed in the C# and JavaScript programming

languages. Unity was used to develop a particle animation to visualize the narrowcasting interface.

1.7 Sources and Sinks

For auditory modeling, a source is virtual speaker, and its metaphorical organ is a mouth. A sink is a virtual listener, and its corresponding metaphorical organ is an ear. Narrowcasting operations include four commands— mute & deafen (excluding operations) and solo & attend (focusing operations). We use a 2D iconography to express these four orthogonal (independent) states (Fig. 1).

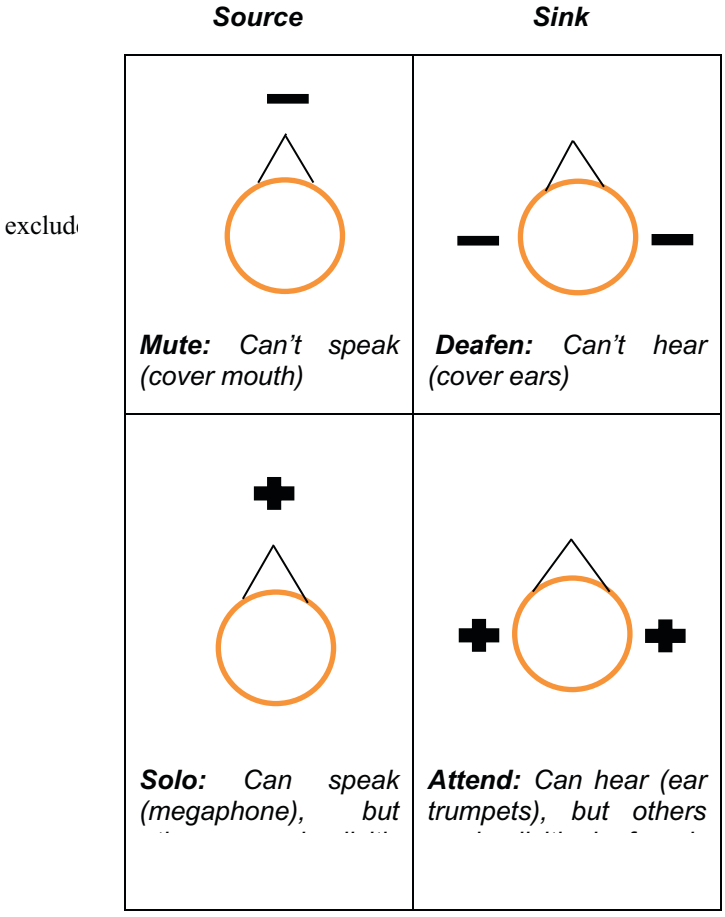

Fig. 1: Auditory narrowcasting attributes iconic representations

1.8 Particle

Particles were animated with Unity's particle system. Settings from the particle system components of Unity can be used to create various effects. For example, one can create effects such as rain, snow, and fire. We adjusted various parameters (duration, looping, start lifetime, speed, size, color, simulation space, emission, shape, triggers, renderer) to implement narrowcasting visualization particle animation.

2. Implementation

2.1 Make particle animation

We used Unity to extend narrowcasting interface, developing in the C# programming language. We extended a script written in C# (Channel Manager KJM) to implement particle animation in the narrowcasting interface. We configured this particle animation using Unity's particle system. As seen in Figs. 2 and 3, we used various parameters (duration, start lifetime, speed, size, color, simulation space, emission, shape, triggers, renderer) to animate particle flow from sources to sinks. Particles are continuously generated. We set particles as follows.

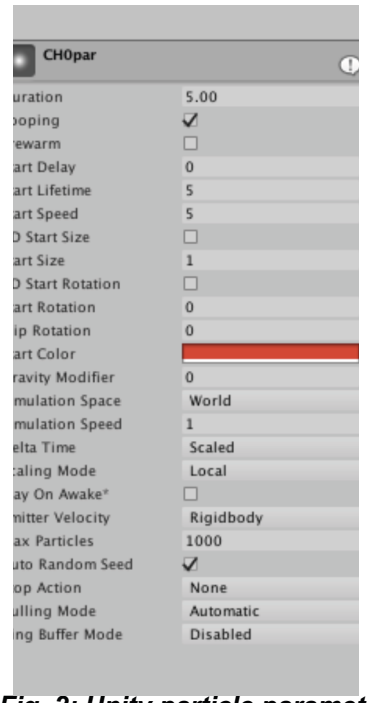

Fig. 2: Unity particle parameters

For Start Color attribute, color is that of source. For Simulation Space attribute, we used "`World`". By doing so, particles fly automatically from each source to their respectively determined ("autofocused") sink.

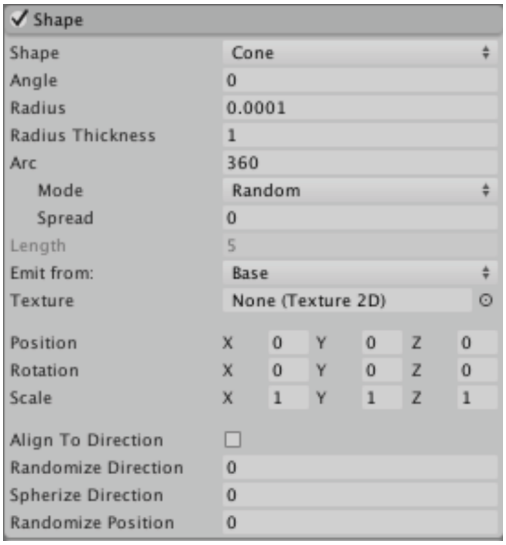

Fig. 3: Unity particles Shape module

Shape attribute: Shape is Conical. Angle is 0°. Radius is 0.0001, very tightly beamed, because it is necessary to make the particle fly in nearly straight direction.

Fig. 4: Unity Triggers module

The `Colliders` attribute sets the collider of an object to trigger when a particle touches it. The `Enter` attribute is configured to invoke the `OnParticleTrigger` method with callback when a particle enters the collider. When each particle reaches its respective sink, the size of the colliding particle is set to 0.

2.2 Launch particles

Particle color (livery) is set to match that of its source. We use the function `Makeparticle` to launch particles towards the determined (autofocused) sink (as shown in Fig. 5).

Fig. 5: Particle animation– Source is CH5; sink (asterisked) is CH0.

2.3 Control stop of particles

We use the `StopParticle` function to arrest the particles, thereby preventing them from continuing to move after they have reached their "destination."

2.4 Control particle collision

We use triggers to make particles disappear when they reach their respective sink. But it doesn't work because trigger reacts with other sinks along the flight path. Therefore, we developed a script written in C# (`Collision`) to solve the problem. By using the function of `OnParticleTrigger`, trigger responds only to the specified sink.

2.5 Compose music

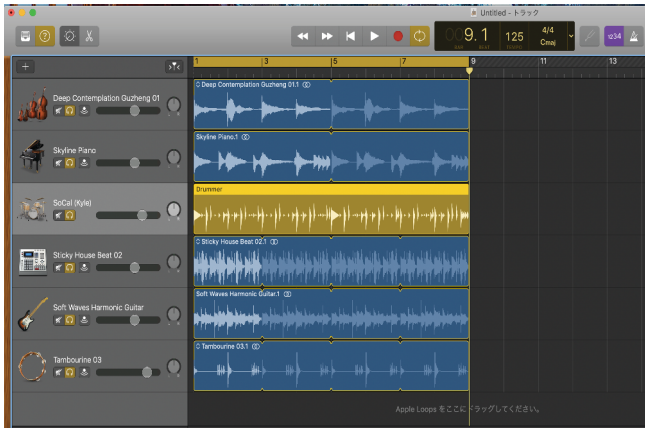

Figure 6: Demonstration music

We composed polyphonic music using GarageBand specifically for demonstrating this system, as shown in Fig. 6. The composition is intended to foster a calm mood.

3. Conclusion

We were able to visualize the state of narrowcasting communication, as shown in Fig. 7.

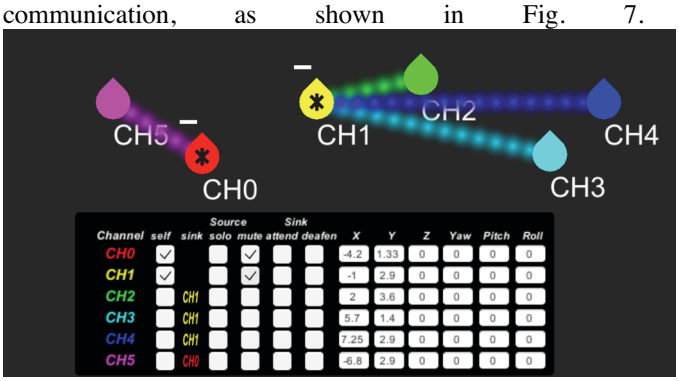

Fig. 7: Narrowcasting visualization– CH5 is auditioned by CH0, and CH2, CH3, and CH4 are auditioned by CH1.

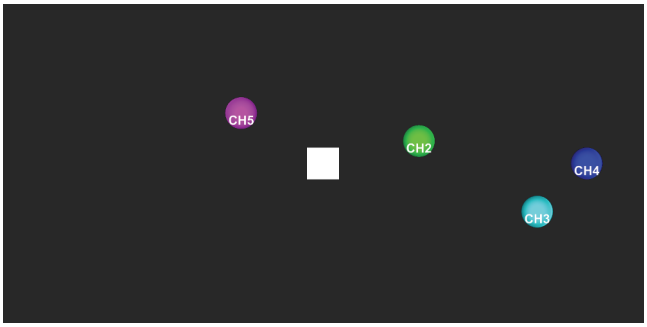

Fig. 8: Egocentric soundscape collapse

We could also deploy this program as native iOS & iPadOS application.

4. Future Work

In future work, we would like to apply technique applicable to immersive interface, such as that presented by HMD for VR or AR. We also plan to compose music specifically for demonstrating this system. Finally, the system is currently stand-alone, without networked media streams (only signaling), so it is appropriate only for offline music audition, not realtime conferencing. We hope to integrate realtime audio networking to test practical groupware applications.

5. Acknowledgements

We thank members of the Spatial Media Group. Yoshi Hoshino did the iOS & iPadOS port.

References

[1] Michael Cohen and Hiromasa Kojima. “Multipresence and Autofocus for Interpreted Narrowcasting”. In: AES: Audio Engineering Society Int. Conf. on Spatial Reproduction — Aesthetics and Science. Tokyo, Aug. 2018. ***http://www.aes.org/e-lib/browse.cfm?elib=19653***

[2] Unity https://unity.com

[3] ParticleSystem (Shuriken) https://docs.unity3d.com/jp/460/Manual/class-ParticleSystem.html

[4] Unity5.4 の Particle System の Trigger と Collision モジュールの使い方：http://hajimete-program.com/blog/2016/09/23/unity5-4 の particlesystem の trigger と collision モジュールの使い方/

[5] Unity’s Particle Trigger Technique: https://docs.unity3d.com/ja/current/ScriptReference/MonoBehaviour.OnParticleTrigger.html