Akabeko Ensemble: Cultural multimodal helical computer music installation

Yoshiki Tokumitsu[1] *& Michael Cohen,*[1]

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

Abstract. In this study, we made "Aka-beko" choral ensemble. It comprises an octave of a chromatic musical scale, arranged in a helix and populated by animated oxen, instances of a regional mascot, who lift their heads and sing when triggered by events from a realtime MIDI keyboard controller. The application installation in the University of Aizu UBIC 3D Theater features stereographic display and spatialized sound.

1 Introduction

1.1 Visual music

Visual music displays music multimodally, synesthetically rendering musical information visually as well as acoustically. In this research, we developed a multimedia music performance application designed especially to flatter special features of a University of Aizu venue, the so-called UBIC 3D Theater. CAD software Blender was used to make an Aka-beko (red ox) model, which was imported into game engine Unity [3] for procedural instantiation, parametric livery (color), arrangement (3D position, including location and orientation), rigging (connection between physical controller and virtual object), animation (idling gesture of head-bobbing and occasional head-raising singing howls), and audio clip cueing.

1.2 Helical structure of scales

Musical scales can be modeled as helices, with circumplex (azimuthal) dimension corresponding to pitch class (Do, Do#, Re, etc.) and longitudinal dimension corresponding to absolute pitch, basically the fundamental frequency of a complex tone. Notes arranged in such fashion revolve around a notional listener in the tube of the helix as they climb a scale [1]. We preserve the customary left-to-right order of ascending scales (reflecting cultural preference for left-to-right literacy, perhaps based on awkwardness of writing right-to-left with a right hand). To highlight this structure, the oxen were arranged to have a somewhat redundant coding, rotated to face the center of the left-handed helix, and colored (their name notwithstanding) according to hue on a color wheel [4].

In such a convention, then, a red ox is at 'north' or '12 o'clock,' an orange one 30 degrees clockwise at a little past 'north-north-east' or '1 o'clock,' etc. For a chromatic octave (including accidentals as well as the diatonic scale), the number of notes including the doubled first note is 13.

1.3 Procedural modeling

We used programmatic specification of objects (in our case, Akabeko position, color, etc.), rather than manual arrangement. A single "prefab," with an exemplar generic template, is instantiated 13 times at runtime, all of them together performing a little dance-like ceremony as they are introduced, somewhat resembling the parade of sumo wrestlers before a tournament.

Using parametric specification of cylindrical coordinates to naturally express the location (azimuth around a ring of fixed radius), orientation (facing direction), and color of each ox, the angular bearing and rotational yaw are determined by

$$\text{bearing} = n * 360 / N,$$

$$\text{yaw} = 180 - \text{bearing},$$

where n is index of the Akabeko object and N is the total number of them in an octave, which is a dozen in our case (since the octave doubling of the tonic shouldn't be counted for the number of semitones in a single octave). Likewise, the altitude, which corresponds directly to pitch, is also proportional to note index:

$$\text{height} \propto n * 360 / N$$

2. Implementation

2.1 Blender

CAD software Blender was used to create an Akabeko model, imported into Unity as prefab (template). The Blender model comprises two parts, head and body, with the head suspended from the body through a neck hole. The model was augmented with a default "idling" head-bobbing animation and a head-raising singing gesture. The bobbing animation is a sinusoidal-like motion which starts in a neutral position, raises the head (¼), then lowers through that datum (2/4) all the way down (¾), finally raising it again (4/4) before repeating the cycle. The head-raising singing gesture, triggered by a "`note on`" MIDI event, resembles the corresponding phase of the head-bogging gesture, its "first quarter," but has greater amplitude, emphasizing the singing. The singing head is kept in raised position until the corresponding "note off" event is received, at which point the head drops back into its default bobbing, awaiting more singing cues.

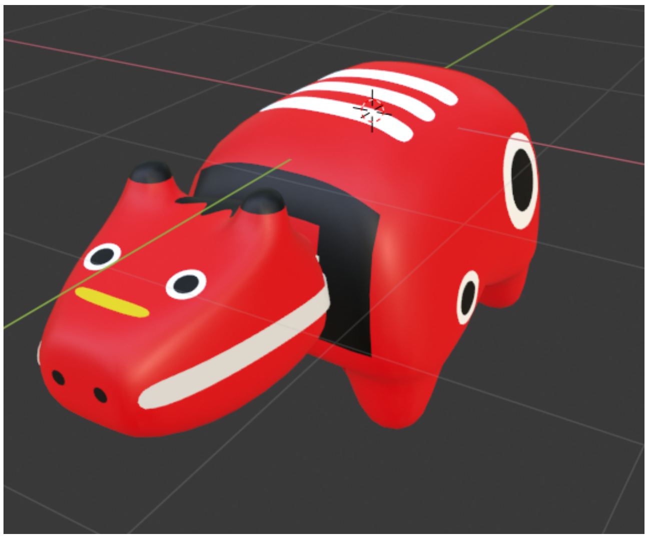


Figure 1: 3D Akabeko Model

2.2 Unity

Unity is one of the most popular real time 3D development platforms. It can be used to develop various software including 2D, 3D, VR and AR environments and experiences. It is used to arrange, animate, and project stereographic rendering.

2.3 MidiJack

MidiJack [2] is middleware for connecting a MIDI controller to Unity rigging (connection between real controller and virtual object). It is used to trigger respective choral events, mapping between the tones across one octave (chromatic scale) to corresponding Akabeko ensemble members.

Ensemble voices (instrument, timbre, or "patch") are supplied by Shakuhachi notes, prerendered by Mathematica as audio files, imported into Unity as audio clips. They are triggered at runtime by MIDI events, auditorily rendered along with singing gesture.

2.4 Animation

The cultural Akabeko has a natural "head-bobbing" or "bobble-head" default nodding, used in our application as a default "idling gesture." This gesture was originally developed in Blender by creating several key poses (Figs. 1, 2, 3) and inserting them into keyframes (Fig. 4), finally adjusted to interpolate smoothly.

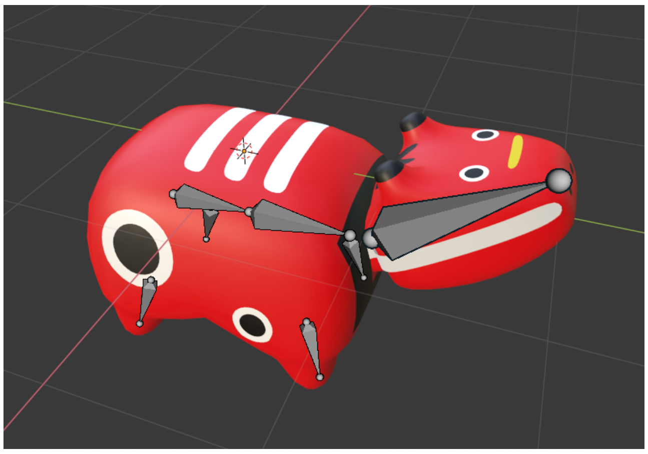


Figure 2: Head-bobbing, up

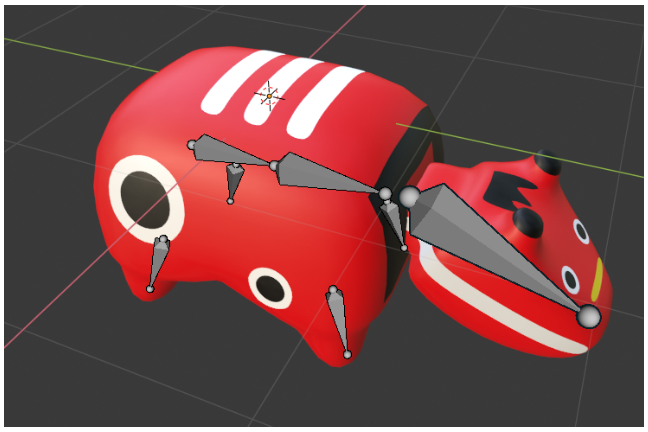


Figure 3: Head-bobbing, down

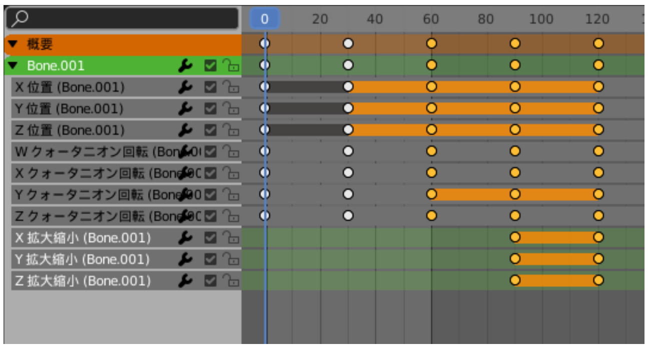
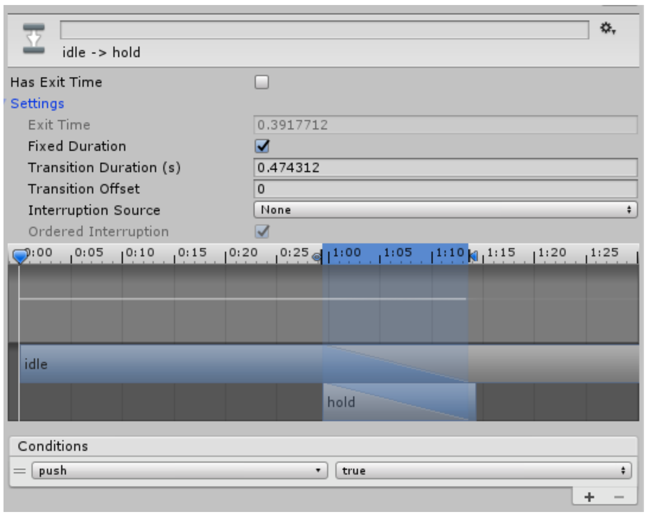


Figure 4: Animation graph (Blender)


idle -> hold

Has Exit Time ☐
Settings
Exit Time 0.3917712
Fixed Duration ✓
Transition Duration (s) 0.474312
Transition Offset 0
Interruption Source None
Ordered Interruption ✓

Conditions
push true

Figure 5: Unity animator

Unity automatically transitions between the two animations [5], adjusting the animation switching time with Fixed Duration (Fig. 5). The transition animation with the transition duration value 1 takes the same amount of time as the animation above. The animation is played below from the middle by changing the value of Transition Offset. A natural gesture when singing is to raise (not lower) head. We smoothly integrate default head-bobbing idling gesture with singing gesture when respective note is played.

2.5 Stereographic display

Having arranged separate but attached left and right cameras for stereographic display, we configured separate rendering windows, driving separate, passively polarized filtered projectors. The overlaid stereo pair is projected on a "silver screen" that preserves polarization. Circularly polarized lenses eyewear (corresponding to projector filters) worn by users separate left and right views for binocular viewing.

2.6 Displaced multimedia perspectives

Our scene is not "virtual reality," which features an intimate, egocentric, 1st-person perspective on a virtual world, but an abstract 3rd-person experience: the visual perspective, represented by the virtual stereo camera, is best placed outside the helix, typically behind 6 o'clock (or south), the better to see the entire Aka-beko ensemble. However, the virtual spatial microphone is best located in the tube of the helix, the better to appreciate the directionalized notes. By separating the virtual microphone and virtual camera, graphical rendering is from outside of helix, whereas acoustic rendering is from within the helix.

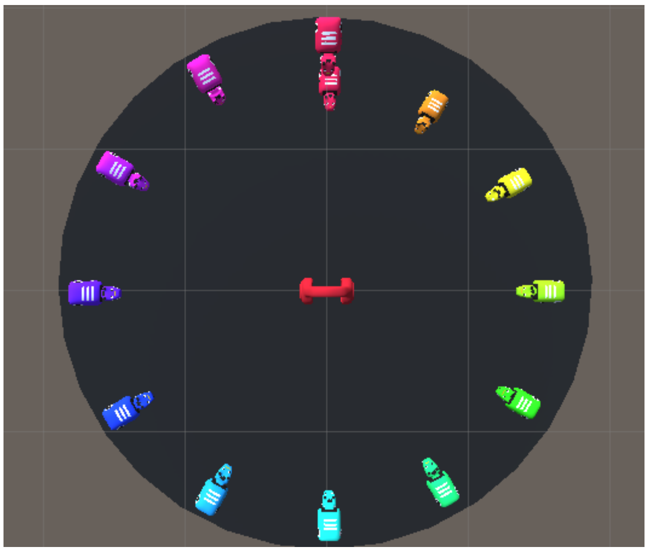


Figure 6: Modulate color and arrange as helix

2.7 Zig Sim

Zig Sim [6] is smartphone application for networked control. It uses embedded IMU sensor such as smartphone and wireless LAN module such as Wi-Fi, as shown in Fig 7. Quaternion function was used to rotate camera rotates around Akabeko helix while revolving, always facing center of scene.

2.8 Spatial Sound & Rendering

Audio clips were prepared "offline" using Mathematica's Shakuhachi instrument, one for each of the 13 notes represented from the chromatic scale. These were attached to the respective oxen, one note each. These audio sources are configured as "2D" sources, like non-diegetic scene elements, so when triggered they are not immediately directionalized or spatialized by the game engine, but instead simply sent to the workstation audio output as an unmodified mono signal. The speaker array mixer is simultaneously configured to route such display to the respective ceiling-mounted speaker or more generally panned between a pair of speakers.

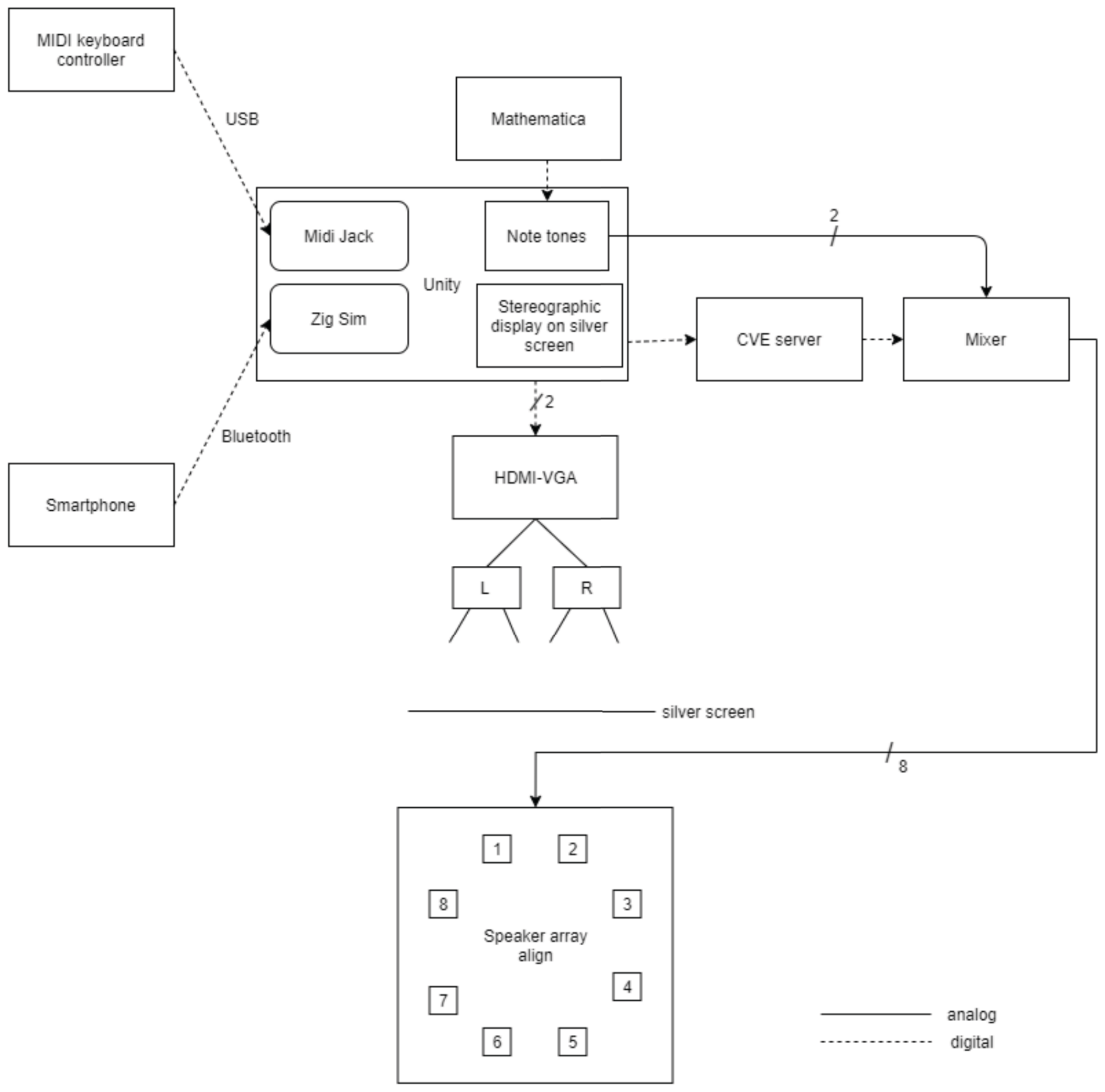


Figure 7: System schematic

2.9 Data flow for deployed application

Akabeko ensemble is hosted on a workstation (Mac Mini) running game engine. A MIDI keyboard controller is attached via USB, and a smartphone running Zig Sim connects via Wi-Fi. After a ceremonial "splash" (introductory cut scene), the virtual objects await triggers from the controllers: the oxen await musical cues to sing, and the camera awaits rotational imperatives to rotate and revolve around the ensemble helix in a phase-locked-loop "inspection gesture." Parallel game windows with runtime renderings from perspective of left and right halves of virtual stereo camera are connected to passively polarized stereo projectors, which imagery bounces off a silver screen that preserves such polarity, whence guests can see stereo imagery through complementary eyewear, a.k.a. "3D glasses."

3. Conclusion and Future Work

Our proof-of-concert has several restrictions. Because unextended analog audio output jack on workstation is stereo, only 1 or at most 2 output channels could be directly diffused by the speaker array. This means that the system is "monophonic": only 1 note can be rendered at once without confusing collision of a chord tones, at the location of the last-played note of an arpeggio. To work-around this limitation, we could deploy an audio interface, a digital "fan-out" that allows distribution of more than two audio channels. (Such interfaces had been based on Firewire or USB, and modern USB-C bandwidth can support many audio channels.) Our sound spatializer mixer has eight inputs, so that level of polyphony could be supported. We would have to cull or aggregate some of the keys to accommodate extended but still limited input receptivity, the diatonic scale (just the "white notes") being a natural selection. Only songs within a certain key (such as C or A minor) could be played properly, but polyphony would be full. Since the scale wraps around, notes in the same pitch class map to the same azimuthal direction, since our speaker array doesn't model altitude, height, or elevation.

4. Acknowledgements

We thank Ryouhei Hashimoto for his assistance.

References

[1] Jens Herder and Michael Cohen. "The Helical Keyboard: Perspectives for Spatial Auditory Displays and Visual Music". In: JNMR: J. of New Music Research 31.3 (2002). Ed. by Gunnar Johannsen and Giovanni De Poli, pp. 269–281. ISSN: 0929-8215.

DOI: 10.1076/jnmr.31.3.269.14180.

[2] MIDI Jack. https://github.com/keijiro/MidiJack

[3] Unity User Manual. https://docs.unity3d.com/Manual

[4] 【Unity】スクリプトで RGB と HSV を相互に変換する方法. http://nn-hokuson.hatenablog.com/entry/2017/04/12/194631

[5] 【Unity 開発】Animator Controller の遷移設定 (Duration 等)【ひよこエッセンス】 https://hiyotama.hatenablog.com/entry/2015/06/29/090000

[6] Zig Sim. https://zig-project.com