Metering "Black Holes": Networking Stand-Alone Applications for Distributed Multimodal Synchronization

Michael Cohen, Yousuke Nagayama, & Bektur Ryskeldiev
Spatial Media Group, University of Aizu
Aizu-Wakamatsu, Fukushima 965-8580
Japan
{mcohen, s1200180, d8171101}@u-aizu.ac.jp

ABSTRACT

We have developed a phantom GUI emulator that can read from otherwise stand-alone applications, complementing a separate parallel program that can write to such applications. In conjunction with the "Alice" desktop VR system and previously developed "Collaborative Virtual Environment," both of which are freely available, virtual scene exploration can synchronize with multimodal peers, including panoramic browsers, spatial sound renderers, and smartphone and tablet interfaces.

CCS Concepts

•**Software and its engineering** → *Software prototyping;* •**Human-centered computing** → *Smartphones; Mobile devices;* **Mixed / augmented reality;**

Keywords

Alice, Collaborative Virtual Environment (CVE)

1. INTRODUCTION

The "Alice" [4] [1] [6] 3D interactive scenario rapid prototyping environment is recognized for its inviting accessibility, including to non-expert programmers. It features rich galleries of avatars, props, scenery, vehicles, and objects, composable into dynamic scenes driven by easily configured animation, conditional and nondeterminstic triggers, and user interface events. Originally architected by the late Randy Pausch, (of "Last Lecture" fame [9]), it was not intended to have a networked interface.

A subsequent version allows export of IDE-generated Java code [7] [5] [8]. With a NetBeans plug-in, Alice v. 3 programs can be extended, allowing user-developed networked interfaces, but such augmentation is somewhat involved. Casual users are mostly unwilling or unable to confront the daunting installation of a plug-in and Java networking development. Alice v. 2 lacks such extensibility altogether, and is inherently "stand-alone."

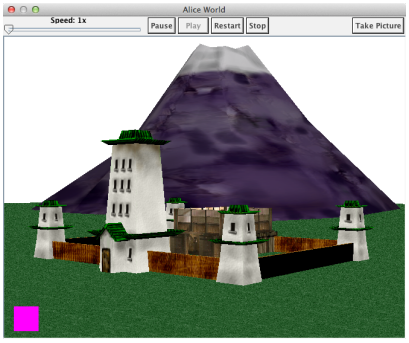

Figure 1: Stand-alone Alice scenarios can now be networked.

These "shut in" qualities notwithstanding, compelling Alice scenes could be even more rewarding if they were extended to synchronize with peered scenes, for stereography or groupware applications such as online games, or complementary control and display, such as photospherical browsers and spatial sound renderers, and alternate interfaces, such as smartphones and tablets.

Therefore, we developed a networked interface for Alice, including the ability to sense Alice state and distribute such events to parallel clients in a distributed environment. Using the resources explained here, even unsophisticated users can easily and freely design and deploy networked custom virtual worlds, rigging scenes, avatars, objects, props, and cameras to external interfaces. Combination with a complementary Alice-writing interface that reads network events [3] allows deployment of duplex scenarios that can both display and control distributed information.

2. IMPLEMENTATION

The Java `Robot` class is used to meter otherwise "mute" applications, as it can sense the colors of specific pixels in a display raster. The original purpose of `Robot` was to facilitate automated testing of Java platform implementations, but we exploit its ability to emulate GUI events and sense designated screen locations to impose sensitivity (to controlling input) and exposure (as displayed output) upon otherwise uncommunicative interfaces.

As illustrated by Figure 2, we use our own Collaborative

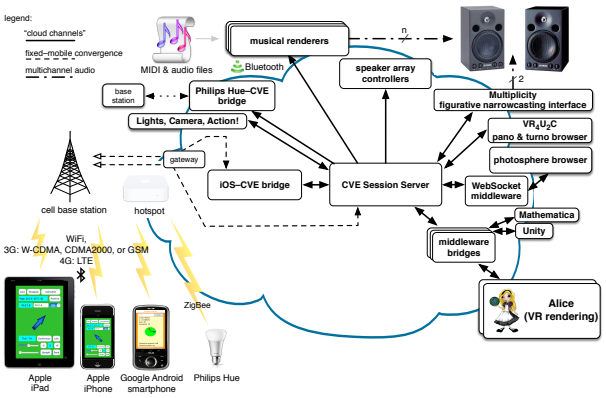

Figure 2: System Architecture: Using newly developed GUI meter and control simulator, "locked-in" applications can be networked. Space doesn't allow explanation of all compatible interfaces, but generally any CVE-conformant client can be integrated.

Virtual Environment (hereafter "CVE") protocol to synchronize distributed interfaces using a client-server architecture, coordinated by a designated session server. External controllability of Alice scenes is described by [3]. The reciprocal process described here reads Alice presented events by encoding them as colors on designated "impedance matching" panels in the Alice scene, which are sampled by an external monitoring process. The color swatches are assigned by conventions of the communication process to certain orthographic positions in the viewport, smallish inconspicuous squares in the corners of the runtime window that reflect salient scenario and scene state.

The swatches are oriented to face the virtual camera and are fixed in the viewport's frame-of-reference (Alice "`vehicle`") like a head-up display (HUD) dashboard, so constantly present reliable "signage." Since the monitoring process references absolute pixel positions, as long as the Alice window is properly placed, its swatches can be sampled.

Integrating the receiving and transmission network interfaces enables symmetric deployment for effective peer-to-peer communication (through client-server topology, in which a server redistributes session events to channel subscribers). Besides parallel instances of Alice, running on a single host or multiple computers, other CVE-conformant multimodal clients can exchange state with session-enrolled peers, including multimodal displays. For instance, a sound diffuser can directionalize audio sources, to reinforce Alice scene situation awareness or heighten immersion and presence.

3. EXPERIENCE

Since nonspecific, extrinsic control could also coordinate responsive expression, demonstration of such capability wouldn't particularly showcase the metering of Alice scenes. Because deployment of Alice event handlers to process keyboard events would be indistinguishable from deployment of a separate input processor connected to Alice, the only way to properly demonstrate Alice readability is to exercise "deep" Alice controls.

Therefore, the demonstration features Alice-specific controls that manifest externally. Clicking on objects in a scene triggers not only Alice-internal event handling (animated orbiting "inspection" of the respective object) but also dynamically spatialized sound. An aural orientation signal (short tune) comes from a notional direction (such as "North)" or landmark (such as "Mt. Fuji"), and an audio renderer infers the azimuth of the Alice-situated virtual perspective and adjusts the musical soundscape accordingly, so the user enjoys a coordinated visual and aural experience.

4. CONCLUSIONS

Alice 2 had been like a black hole, from which no information could practically escape or be detected. By expressing "side effects" at the "event horizon," as it were, we can network it, in parallel with multiple instances or other compatible interfaces. All the software is freely available, and is not difficult to set-up and configure, even for non-specialists.

Alice 2 was chosen as the first target of the networking, as it is used in a popular course [2] at our university, but the same technique could be used for any other program, including not only rapid prototyping environments such as "Scratch"[1] but any GUI-controlled program.

In the future, we might experiment with the presumably only other modality with which such a "shut-in" application might communicate with independent processes (in the absence of diagnostic messages spooled to a console logger that could be parsed by external mediating interception): audio. An Alice program can emit tones that could be interpreted by other programs, such as Chirp[2] or Google Tone,[3] which use audible tones to encode URLs.

5. REFERENCES

[1] Joel Adams. *Alice in Action: Computing Through Animation*. Delmar Learning, 2006.

[2] Michael Cohen. Multimodal Machinema at the University of Aizu. *AIS SIGHCI Newsletter (Assoc. for Information Systems, Special Interest Group on Human-Computer Interaction)*, 12(1):10–11, July 2013.

[3] Michael Cohen. Smartphone Rigging with GUI Control Emulation for Freeware Rapid Prototyping of Mixed Virtuality Scenes. In *MGIA: SIGGRAPH Asia Symp. on Mobile Graphics and Interactive Applications*, Macao, December 2016.

[4] Wanda P. Dann, Stephen Cooper, and Randy Pausch. *Learning to Program with Alice*. Prentice Hall, second edition, 2008.

[5] Wanda P. Dann, Stephen P. Cooper, and Barbara Ericson. *Exploring Wonderland: Java Programming Using Alice and Media Computation*. Prentice Hall, 2009.

[6] Tony Gaddis. *Starting Out with Alice: A Visual Introduction to Programming*. Addison-Wesley, 2007.

[7] John Lewis and Peter DePasquale. *Programming with Alice and Java*. Addison Wesley, 2008.

[8] Vanesa S. Olsen. *Alice 3 Cookbook*. Packt Publishing Ltd., Birmingham, UK, 2011.

[9] Randy Pausch and Jeffrey Zaslow. *The Last Lecture*. Hyperion, 2008.

[1] https://scratch.mit.edu
[2] http://www.chirp.io
[3] https://chrome.google.com/webstore/detail/google-tone/nnckehldicaciogcbchegobnafnjkcne