UDKOSC: AN IMMERSIVE MUSICAL ENVIRONMENT

Robert Hamilton
Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University, Department of Music
rob@ccrma.stanford.edu

ABSTRACT

UDKOSC is a visually and aurally immersive rendered multi-user musical performance environment built in the Unreal Development Kit (UDK), a freely-available commercial gaming engine. Control data generated by avatar motion, gesture and location is routed through a bi-directional Open Sound Control implementation and used to drive virtual instruments within a multi-channel ambisonic sound-server. This paper describes the technical infrastructure of UDKOSC and details design and musical decisions made during the composition and creation of *Tele-harmonium*, an interactive mixed-reality musical work created in the environment.

1. INTRODUCTION

Computer-based rendered environments have played an extensive role in the production of rich audio-visual experiences within the realms of digital cinema and gaming, leveraging the power of detailed graphical rendering processes, the infinite scope of concept afforded by virtual representation systems, and the flexibility of real-time interaction models to create reactive and immersive virtual representations of existing and novel spaces alike. Sound and music have always played important roles in traditional media uses of virtual space, however significantly less explored has been the role of the virtual environment itself as an interactive generative system capable of real-time active and reactive creation and realization of musical form as well as collaboration on musical interpretation. Potential synergies between the visual and gestural control components of game-derived virtual environments and musical output from intentional or algorithmic musical systems are numerous, driving a need for coherent and comprehensive mapping schemata between potential interaction models, each one bridging modalities of gesture, motion and action in virtual space with the generation of musical sound, structure and meaning.

As high-speed networking has become increasingly ubiquitous, commodity network use has exploded with millions of users exploring online gaming and social networking in fully rendered real-time interactive environments such as SecondLife [9] and World of Warcraft [13]. The ability to visually connect in a shared "virtual" space potentially affords users of rendered environments a grounding sense of presence and connection that can increase both a user's engagement and feeling of immersion. By combining control systems designed for nuanced avatar control and motion within game-based rendered environments with powerful computer-generated musical systems capable of creating and spatializing sound and music, shared musical performance environments can be created, linking performers and audiences alike from distanced locations.

UDKOSC[1] is a musical performance system that repurposes user actions and events occurring within a fully-rendered networked gaming environment - the Unreal Development Kit (UDK) [11] - as the control systems for musical creation and spatialization. By integrating the Open Sound Control (OSC) protocol into the UDK codebase, user-controlled avatars control software-based sound servers in real-time with their motion through and interaction with the rendered environment itself. Artifacts in the environment, including projectiles and the static-mesh building blocks of wall, floor and object components have been repurposed as control-data generating reactive entities with which performers can interact and manipulate. In this manner, the virtual environment and the interactions that occur within UDKOSC can be intuitively repurposed into tools for musical composition, performance, network-aided collaboration and transmission.

2. PRIOR WORK

The use of networked/multi-user video game engines for music and sound generation has become increasingly common as generations of musicians who have grown up with readily accessible home video game systems, internet access and personal computers have sought to bring together game worlds, wide-spanning networks and interactive control paradigms with musical systems.

2.1. Virtual Performance

Networked performance ensembles existing solely in network space like SecondLife's Avatar Orchestra Metaverse [1] explore the collaborative possibilities of avatar performers in virtual space, using avatar choreography and group gesture as a complementary presentation in virtual space to streaming musical performance. Musical works are performed by the AOM through the use of Heads-Up Display (HUD) control panels rather than user's motion through the environment itself.

2.2. Game-based musical environments

q3osc [5] is a heavily modified version of the open-sourced ioquake3 gaming engine featuring an integrated OSC implementation for bi-directional communication between a game server and one or more external audio servers. Though similar in scope and concept to the UDK, q3osc's ioquake3 gaming engine lacks the UDK's powerful graphic rendering capabilities.

[1] https://github.com/robertkhamilton/udkosc

2.3. Mixed-Reality Performance

The combination of live musical performers on traditional instruments with virtual performers in rendered environments has been seen in a series of mixed-reality works by Stanford's SoundWire [3] and Music in Virtual Worlds research groups, combining high-quality and low-latency audio streaming software [2] with fully-rendered OSC enabled environments supporting ensembles of virtual performers. Musical spaces built within the Sirikata [6] engine were used to control rich spatialized sound synthesis engines in a number of musically rich mixed-reality musical performances including a series of performances at the 2009 MiTo Festival in Milano, Italy [8].

3. UNREAL DEVELOPMENT KIT (UDK)

The Unreal Development Kit (UDK) is a next-generation professional development framework for creating visually rich networked gaming and simulation environments. Built upon the Unreal Engine 3 - the engine powering current commercial gaming titles such as Gears of War, Bioshock and Unreal Tournament - the UDK offers tools for building fully-rendered complex virtual architectures within the Windows operating system (XP/7), designing AI and animation workflows, a robust multi-user networking layer and a powerful object-oriented scripting language called UnrealScript [12]. The UDK is available at no cost for educational and non-commercial use and is updated monthly with new features ranging from enhancements to the lighting and modeling tools to integration with Apple's iOS, allowing works created in the UDK to be published to iPhone, iPad and iPod touch devices.

4. UDKOSC SYSTEM OVERVIEW

UDKOSC combines an Open Sound Control implementation with a customized UDK codebase featuring enhanced user, environment and object controls and behaviors in an effort to afford users an enhanced level of control over motions and actions that can be used to drive musical systems. OSC messages are encapsulated as UDP packets and passed along their own network threads, not piggy-backing on existing network calls made by the engine itself.

4.1. Sound and Spatialization

Avatar motion and gesture in virtual space in UDKOSC is rendered and subsequently spatialized across a multi-channel sound field using a second-order ambisonics encoder and decoder built in the SuperCollider programming language [10]. When using a multi-channel sound system powered by ambisonics, all speakers are used to spatialize individual sounds, creating a stable and localized sound field surrounding a given space or audience. Sounds moved through the virtual space are correlated to movements through the physical concert halls: as an avatar traverses a rendered environment interacting with objects in the environment, sound is perceived by the audience as moving across the physical hall.

4.2. OSC Communication

UnrealScript has a provision for binding Windows Win32 .dll's to UnrealScript classes, enabling code such as the OSCPack C++ implementation of Open Sound Control to provide bi-directional communication between game engine and sound-server. This custom Windows oscpack_1_0_2.dll was compiled with specific extern methods and mirrored data structures to communicate with the game engine, both to stream specific game data out as OSC formatted messages over UDP (including Pawn XYZ coordinate positioning, XYZ coordinate tracking for in-game projectiles, state and XYZ coordinate info for Static Mesh and Interp actor classes) and to send control data from iPad controllers and from the audio-engine itself back into the game to control Global GameInfo parameters such as world Gravity and GameRate, as well as specific positioning information for projectiles and static actors.

4.3. Client/Server Architecture

The Unreal Engine makes use of a client-server architecture: there exists a "server" game instance with knowledge of position and state for every connected client actor and environmental event. As latencies between connected clients have the potential to vary widely, each client's local view of the current game state is constantly being readjusted through communication with the server, potentially offering slightly different views of a given environment across client instances at any given time. OSC communications are initialized from within the server game instance rather than within each client instance so that an authoritative and shared view is used to generate the outgoing OSC data stream.

4.4. OSC Mappings

At this point, OSC hooks have been written for various actor and event classes within the UDK framework allowing for a number of control mappings.

4.4.1. Actor motion in coordinate space

As user avatars move through three-dimensional coordinate space, each avatar's X, Y and Z location data is streamed over OSC to the ambisonic sound-server. This location data can be used to control positional ambisonic spatialization of a continuous sound-source around a multi-channel speaker setup, to trigger pre-rendered audio files, or to signal large-scale sectional divides in musical structure.

4.4.2. Projectile/Environment tracking and collision

While projectiles launched from a client avatar position are typically associated with violent action in the context of video-games, modified projectile code can be repurposed as an enactive interface for sound creation. Points of collision in coordinate space between

projectiles and environment are marked and used as spatialization coordinates in physical sound space. Projectiles can be given bouncing behaviors, creating successions of collisions capable of introducing rhythmic elements through repetition. Additionally, coordinate points on each of 3 dimensional axes can be mapped to elements of a musical note event, including pitch, timbre and duration.

4.4.3. Projectile size and amplitude

As projectiles move through and collide with their containing environment, users can scale their physical size up and down. By mapping the size of each projectile to the relative gain of the resultant sound of its collision, a simple mapping schema can be seen.

4.4.4. Projectile motion and homing behaviors

Users can be given control over the targeting systems of generated projectile, locking projectiles onto a given coordinate location in space or to an object which can move based on its own set of allowable behaviors. In this manner, users can control sets of audible parameters and the spatialization of sounding projectiles simultaneously and coherently. By keeping projectile speed of motion constant and limiting projectile turning radius, interesting grouping patterns are formed as sets of projectiles arrive at a given point only to continue past, constantly updating their direction to re-target the same location. Such behaviors offer similar behaviors as crude flocking algorithms.

4.4.5. Continuous projectile sound

Additional relationships between projectile action and musical sound are possible, including the control of continuous sound sources with the coordinate location of projectile objects. By mapping the location of multiple projectiles to similar continuous sound sources (such as a sine-wave or a continuously-sounding physical model) rich and complex musical textures can be created with little user effort. When combined with projectile homing behaviors, groups of projectiles become controllable texture swarms.

4.4.6. Touch-based homing targeting

By combining the projectile homing methodologies previously discussed with external sources for the setting of target coordinate locations, users can dynamically move groups of projectiles through coordinate space in a fluid and enactive manner. Towards this end a multi-touch iPad application has been used as a driver of homing locations for multiple simultaneous locations. The enactive quality of controlling sets of homing projectiles is greatly enhanced through the use of a physical touch-based paradigm, even one without physical touch feedback of any kind.

4.4.7. Triggers: active and location-based

For sound events or processes that require simple mode switching control or triggers similar to a key-press, users can either move through specific spatial volumes such as X,Y,Z coordinate sets (location-based) or manually interact with trigger-actor acting like a switch, button or lever (active).

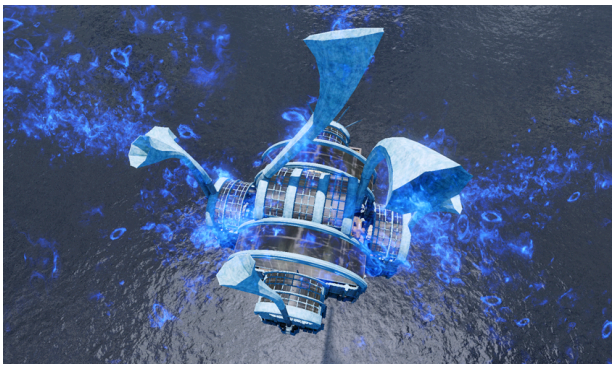

Figure 2: The *Tele-harmonium* performance space

5. SAMPLE WORK: *TELE-HARMONIUM*

Tele-harmonium for live piano and virtual performer was the first work written using UDKOSC and was designed to incorporate the architecture of the rendered environment itself as an instrument capable of performing a duet with a live pianist. Visually, the Tele-harmonium was conceived as a giant machine-building, similar in concept to an immersive music box with rotating gears and mechanisms to mechanically reproduce sound and music.

Figure 3: Tele-harmonium central organ console

The storyline of the work describes the building as an instrument capable of playing back musical memories and as such, the virtual performer is seen travelling, entering and interacting with structures within the building to create and control sound and music. At the center of the space is a rendered organ-like console, intended as the virtual manifestation of the live pianist, though perhaps more appropriately, the pianist should be seen as the live representation of the virtual instrument. Horn-shaped structures fill the environment, with each construct being capable of sending OSC messages upon avatar or projectile collision or interaction. In this manner, objects in the environment become playable “instrumental” constructs. Specific

constructs in the environment are designed as attractors for projectiles based on control modes set by performers in real-time. Around the center of the Tele-harmonium space are a series of poles and hanging bells, each of which acts as an attractor. When projectiles collide with the bell constructs, specific pitches are sounded, resulting in the formation of a series of tuned bells.

When projectiles are triggered to be attracted to the standing poles, continuous pitches are generated based on each projectile's measured distance to the center of the environment, with the coordinate point [0,0,0] resting at the center of the keyboard console construct. Continuous sounds are created using a modified version of the projectile object, mapping spatial coordinates to frequency, amplitude and specific timbral characteristics of a simple synthesized instrument. Swarming and following behaviors are extremely effective in generating rich soundfields when mapped to a simple continuous sound source, with slight fluctuations in pitch and timbre working together to create a timbrally rich musical sound.

Figure 4: Post attractors are mapped to speaker locations in physical space using ambisonics

Musically, *Tele-harmonium* is loosely built upon the harmonic and melodic structure of the *Sonata for keyboard in D minor*, K. 213 (L. 108) - "The Lover" - by Dominico Scarlatti. Excerpts from the Sonata are played both in the form of processed recording snippets as well as live materials by the pianist. Additional materials that comprise the notated piano score were composed as derivative variations on particular phrases and structures found in the original Sonata. The virtual performer controls the structure of the composition as a whole, moving though specific locations and triggering recorded excerpts of the original Scarlatti in conjunction with the live pianists playing.

To allow the virtual performer the ability to successfully duet with an instrument as melodically and harmonically flexible as a piano, both projectile collision and continuous OSC data streams were connected to sound-generating virtual instruments in SuperCollider. As projectiles collide with specific objects in the rendered environment, simple pitched sounds are generated using bursts of noise passed through tuned filter banks. Settings for filter frequency/pitch and amplitude are extracted from the UDKOSC stream: frequency was mapped to a Cartesian distance from the central rendered keyboard while amplitude was mapped to the size of the projectile itself.

6. REFERENCES

[1] Avatar Orchestra Metaverse. URL: http://www.avatarorchestra.org/, accessed 1/2011.

[2] Cáceres, J. and Chris Chafe. "JackTrip: Under the hood of an engine for network audio." In Proceedings of International Computer Music Conference, Montreal, 2009

[3] Caceres, J., C. Chafe. SoundWIRE research group at CCRMA, Stanford University, 2008. URL http://ccrma.stanford.edu/groups/soundwire/.

[4] Caceres, J., R. Hamilton, D. Iyer, C. Chafe, G. Wang, "China on the Edge: Explorations in Network-based Performance" In Proceedings of ARTECH 2008, 4th International Conference on Digital Arts, 7- 8 November, Portuguese Catholic University, Porto.

[5] Hamilton, R. "q3osc: Or How I Learned To Stop Worrying And Love The ~~Bomb~~ Game" In Proceedings of the International Computer Music Conference., Belfast, Ireland, August 24-29, 2008.

[6] Horn, D., Ewen Cheslack-Postava, Behram F.T. Mistree, Tahir Azim, Jeff Terrace, Michael J. Freedman, and Philip Levis, Infinity and Not Beyond: Scaling Communication in Virtual Worlds with Meru, Stanford University CS Tech Report, 2010 - http://hci.stanford.edu/cstr/reports/2010-01.pdf

[7] Malham, D.: 3-d sound spatialization using ambisonic techniques. Computer Music Journal 4(19), 58{70 (1995)

[8] Mito: Settembre musica (2009). URL http://www.mitosettembremusica.it/en/home.html

[9] SecondLife. URL: http://secondlife.com/, accessed 1/2011.

[10] Supercollider (2006). URL http://supercollider.sourceforge.net

[11] Unreal Development Kit (UDK). URL: http://www.udk.com, accessed 1/2011.

[12] UnrealScript Language Reference. URL: http://udn.epicgames.com/Three/UnrealScriptReference.html accessed 1/2011.

[13] World of Warcraft. URL: http://us.battle.net/wow/en/, accessed 1/2011.