Q3OSC OR: HOW I LEARNED TO STOP WORRYING AND LOVE THE ~~BOMB~~ GAME

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

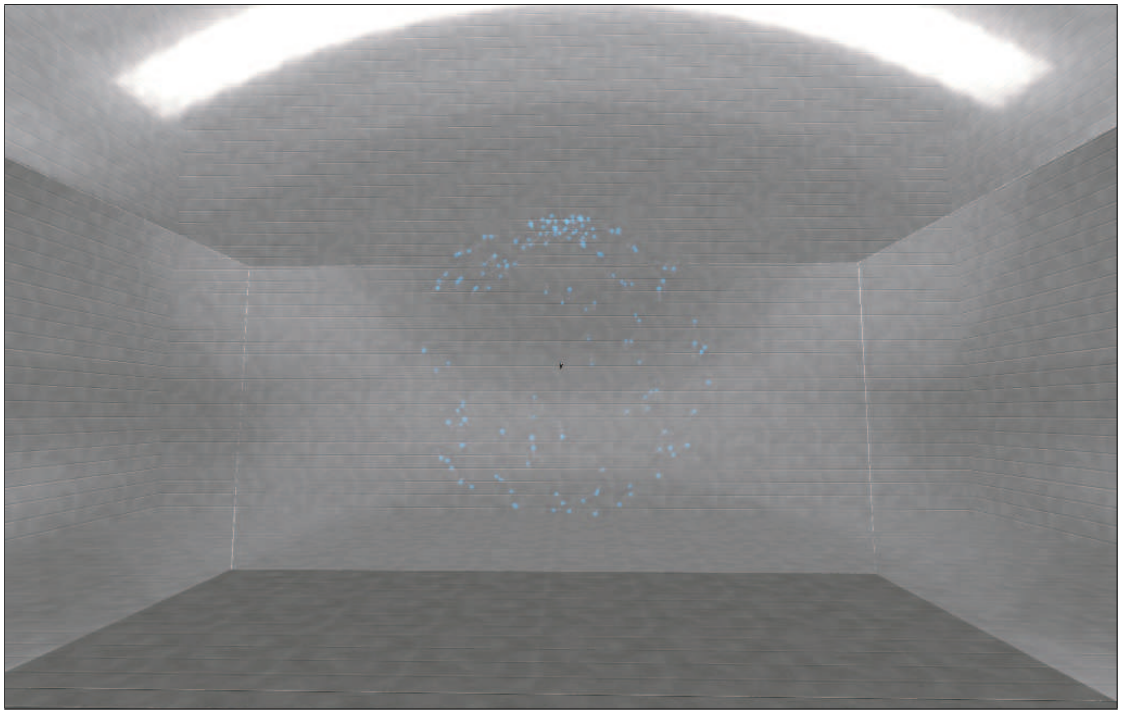

Figure 0. A single q3psc performer surrounded by a sphere of rotating particles assigned homing behaviors, each constantly reporting coordinates to an external sound-server over OSC.

ABSTRACT

q3osc is a heavily modified version of the ioquake3 gaming engine featuring an integrated Oscpack implementation of Open Sound Control for bi-directional communication between a game server and one or more external audio servers. By combining ioquake3's internal physics engine and robust multiplayer network code with a simple and full-featured OSC packet manipulation library, the virtual actions and motions of game clients and previously one-dimensional in-game weapon projectiles can be repurposed as independent and behavior-driven OSC emitting sound-objects for real-time networked performance and spatialization within a multi-channel audio environment. This paper details the technical and aesthetic decisions made in developing and implementing the q3osc game-based musical environment and introduces potential mapping and spatialization paradigms for sonification.

1. INTRODUCTION

Virtual environments such as those presented in popular three-dimensional video games can offer game-players an immersive multimedia experience within which the performative aspects of gameplay can and often do rival the most enactive and expressive gestural attributes of instrumental musical performance. Control systems for moving client avatars through fully-rendered graphic environments (commonly designed for joysticks, gamepads, computer keyboards and mice or any combination of such controllers) place a large number of essentially pre-composed in-game functions, gestures and movements at a gamer's fingertips, available for improvisatory use during different stages of gameplay. And with the advent of easily-accessible fast wired and wireless networks, the focus of computer-based game-play has shifted from predominantly

solitary modalities of play towards an emphasis on a genuinely peer-oriented and communal gaming experience.

Dynamic relationships between game-users and their peers - both virtual and physical - have shifted in kind, bolstered by massive online communities and expansive game-worlds designed and developed with multi-user collaborative or combative play in mind. Much like traditional collaborative musical performance, many networked interactive game environments allow participants to affect the experiences of other participants in (hopefully) positive or (potentially) negative manners.

q3osc is a musical and visual performance environment which makes use of these paradigms of communal networked conflict and collaboration to repurpose virtual entities and traditional game control methodologies into agents and actions capable of affecting musical response in physical auditory space. By extending the source-code of a multi-user video game engine to include Open Sound Control [20] input and output libraries, data from within a game-engine's virtual environment can be encapsulated and transmitted to one or more external sound-servers with the intent of sonifying virtual gesture and motion into a musically rich and satisfying aural experience. In turn, data generated by external performative controllers, potentially extracted from the auditory environment or generated using algorithmic processes can be routed back into the virtual environment as control data for the modification of environment parameters and the control of individual game entities.

Custom modifications made to traditional game elements such as the targeting methods of "weapon" projectiles and global world-level variables such as gravity and avatar speed allow for striking and dynamic change in the visual and performative characteristics of both client behaviors and the virtual environment itself. By empowering game-players with the ability through gesture and motion to generate complex evolving visual and musical structures, relatively simple modifications in pre-existing game code can have profound and far-reaching effects.

At this time, development of q3osc has concentrated primarily on the integration of OSC classes into the ioquake3 source code, and the modification of the behaviors of in-game projectiles to allow individual projectiles to persist and be destroyed dynamically, and to create associative "homing" relationships between projectiles and client avatars. Additional work is ongoing to create powerful and evocative mappings between game-entity motions and gestures and spatialized musical output.

2. BACKGROUND

The gaming platform modified for this project is the open-sourced version of the wildly successful *Quake 3 Arena* by id Software [6], one of the most commercially successful and widely played FPS or "First-Person Shooter" video-games of modern times. In Quake 3 Arena, game players control a futuristic gladiator-like warrior fighting with other players connected over the internet in three-dimensionally rendered graphic environments. Initially released as a commercial product in December of 1999, Quake 3 Arena used a client-server architecture with a significant amount of server-side prediction to allow players from across the world to join together in fast-moving virtual combat environments which themselves could be created and modified by members of the gaming community. Following continued research by id Software into more realistic and technically-advanced gaming engines, the Quake 3 game engine source-code was released under the GNU GPL in August 2005, providing game-developers and "modders" an invaluable tool for the creation of new gaming platforms and virtual environments of all shapes and sizes. Development of the open-source Quake 3 engine has continued under the title ioquake3 [7].

3. RELATED WORK

q3osc's use of visual environments both as representative models of physical acoustic spaces and also as virtual interfaces for the control and display of musical data has been directly influenced by works from both the gaming and computer-music communities. Muench and Furukawa's two-dimensional *small_fish* [4] makes use of bouncing projectiles and their subsequent collisions with virtual entities as midi-triggering controllers for the dynamic creation of musical patterns and forms. Oliver and Pickles' ioquake3/Pure-Data (PD) modification *q3apd* [12] made use of PD's string-based FUDI protocol to export game parameters to a Pure-Data sound-server. The real-time visualizations of ChucK processes displayed by Ge Wang and Perry Cook's *Audicle* [19], as well as game-based Audicle interfaces like the *ChucK-ChucK Rocket* interface respectively use three-dimensional graphical environments and features extracted from game-play to both represent and sonify virtual actions. In addition, this author's ioquake3-based work *maps and legends* [5] made use of q3apd and Pure Data by building virtual compositional maps within which game-players' motions would trigger sonic events and control spatialization of their own specific sound-set within an eight-channel environment.

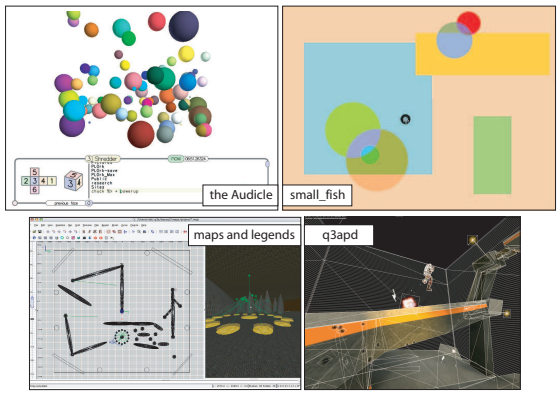

Figure 1. Interactive graphic environments used as generative systems for musical output.

Figure 2. Networked ensemble projects including performers in local as well as remote locations.

q3osc's goal of creating multi-user networked musical environments draws inspiration from a rich history of the use of WAN and LAN computer networks for collaborative performance. Early networked performances by The Hub [21] stand out as rich examples of the complex musical constructs formed through communal composition, improvisation and performance. Stanford's SoundWire project [2] and its recent networked concert performances with the ensembles Tintinnabulate at RPI (NY) and VistaMuse at UCSD utilises multiple channels of uncompressed streaming audio over the Jack-Trip software to superimpose performance ensembles and spaces alike. And the Princeton Soundlab's Princeton Laptop Orchestra (PLOrk) [16] has displayed the powerful possibilities of collaborative networked compositional form using distributed point-source spatialization.

4. SYSTEM OVERVIEW

q3osc consists of a heavily modified or "modded" version of the open-source ioquake3 project which tracks various in-game actions and attributes and exports their values to an OSC client sound-server for sonification and spatialization. As local or remote clients running the standard and freely-downloadable open-source ioquake3 software connect to a game-server via LAN or WAN network connections and move their avatars through the fully-rendered three-dimensional environment, each client's local software updates its position and any commands received from the player's control system to the game-server for coordination with other connected clients. Data extracted from the game-server is output over OSC to one or more sound-servers connected to a multi-channel speaker system, where sonification and spatialization of the game data is performed. Currently under development is an additional layer of communication in which one or multiple sound-servers output OSC messages back into the game environment to control or modify entity, client or environment behaviors.

While support for OSC in a number of interactive musical programming languages (including SuperCollider [11] , Max/MSP [10] and PD [13]) makes any of these languages strong contenders for use in sonifying game data, at this time q3osc is sonified using the ChucK language. The very nature of this project requires the dynamic instantiation of in-game entities - from players to projectiles - at any given time. Therefore audio environments to be used as q3osc sound-servers should also allow for the on-the-fly instantiation of independent audio events without performance lag or noticeable effect. In this respect, the philosophy of the ChucK language meshes perfectly with q3osc. A ChucK sound-server parses incoming OSC messages or bundles and dynamically instantiates or "sporks" independent audio-generating processes for each game-entity in the system which receive ongoing OSC broadcasts of their related entity's position and state data. As game-entities move in the virtual space, their actions generate sounds in a multi-channel audio environment which are in turn spatialized within a multi-channel speaker setup. By presenting q3osc performances in a concert space with the use of multiple video projectors and a local ensemble of q3osc performers present in the room, strong correlations can be made between actions and motions in the virtual space and audio output in the physical concert space.

5. IOQUAKE3 MODIFICATIONS

The task of transforming a game-platform such as ioquake3 from a virtual arena for indiscriminate "deathmatch"-style combat into an abstract musical playground began with elements of FPS gameplay which could be seen as corrolaries to musical or spatial gesture: in particular the movements of game-players and the projectiles they fired. In q3osc, the major efforts of game-code modification have focused on support for the output and input of OSC messages and bundles, modified behaviors and controls to weapon projectiles, and the abilities of game-clients to affect system-state and environment parameters.

5.1. Open Sound Control Integration

q3osc makes use of Ross Bencina's *Oscpack* C++ implementation of Open Sound Control by compiling the Oscpack source-code directly into the ioquake3 C-based project and calling OSC output methods from a number of game-server classes to track changes in position and state for game-clients and entities as well as changes to the game-environment itself. To facillitate linking between Oscpack's C++ classes and ioquake3's C classes, exposed methods in Oscpack were wrapped in "extern C{}" statements to allow the entire project to be compiled using a modified version of the ioquake3 Makefile calling the gcc compiler. q3osc currently features OSC output of client and entity positions and states, support for multiple OSC output destinations (both multiple IP addresses and Ports), and a limited use of OSC input to control entity and environment characteristics.

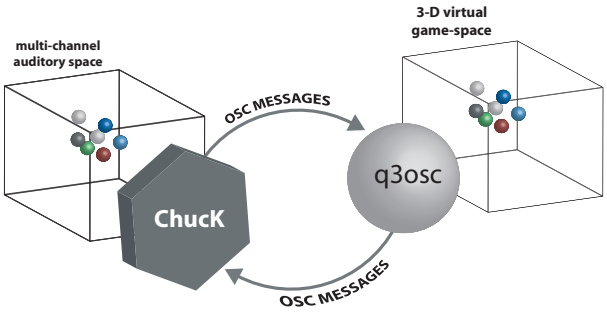

Figure 3. Bi-directional communication between game-server and ChucK sound-server allows for close correlation between elements in both the virtual/visual environment and the physical/auditory environments.

5.1.1. Osc Output

In the ioquake3 client-server model, there exist methods called for each instantiated individual game-client and moving entity called "Think" routines, triggered at extremely short intervals to calculate and adjust client or entity motion vectors given commands made by the client or received by the server. By passing data from "Think" routines to a custom OSC bundle and message formatting class, a stream of data representing constantly changing client or entity position or state can be formatted either as OSC bundles or messages and routed to an OSC client listening on a designated IP and Port address. As ioquake3 makes use of persistant data structures to store information about client-state, at this time a number of client-state parameters (such as client-view angle, three-dimensional velocity or selected weapon) are exposed and can be output using OSC as well.

To facilitate the reading of q3osc OSC output by any standard OSC client, the OSC formatting method used for output can be toggled between bundle and message formats using in-game console flags. OSC output tracking an individual plasma-bolt's current X,Y,Z position (/origin), owner or the client which fired the projectile, an entity ID for a target (/targetnum), whether this message signifies a bounce-event (/bounce), and whether this message signifies the destruction of this particular game-entity (/explode) is represented below first as an OSC bundle, and second as a single-line OSC message.

```
[ /classname "plasma"
  /ownernum 0
  /projectilenum 84
  /origin 102.030594 2550.875000 -2333.863281
  /targetnum 55
  /bounce 1
  /explode 0 ]

/projectile "plasma" 0 84 102.030594 2550.875000
 -2333.863281 55 1 0
```

5.1.2. Osc Input

One extremely powerful feature of OSC integration into the game-engine currently under development is the ability to use external OSC sending-servers to control in-game entity motions and environment states by sending control messages back into the game-engine. By allowing an external sound-server to communicate changes in the real-world auditory environment to the game engine, a more complex synergy between environments can be achieved. For example, by creating an inverse corrolation between amplitude generated by a real-world sound-server and the speed at which projectiles can move in the virtual-world, a strong sense of relation is formed between virtual and physical space. In a more complex example, algorithmic patterns of motion controlling X,Y,Z positions of in-game projectiles can be used to implement independent swarming or flocking behaviors or other strongly patterned group behaviors and causal relationships [3].

5.1.3. Multiple OSC Output Streams

To help OSC client sound-servers load-balance the potentially large number and rapid-rate of OSC streams being generated by q3osc, and the subsequent demands made of a sound-server attempting to sonify each individual stream in real-time, the ability to route OSC streams from multiple entites at any given time to multiple specified OSC-client IP addresses and Ports has been implemented. In a mapping scheme where a single projectile entity fired in virtual space results in the dynamic instantiation of a

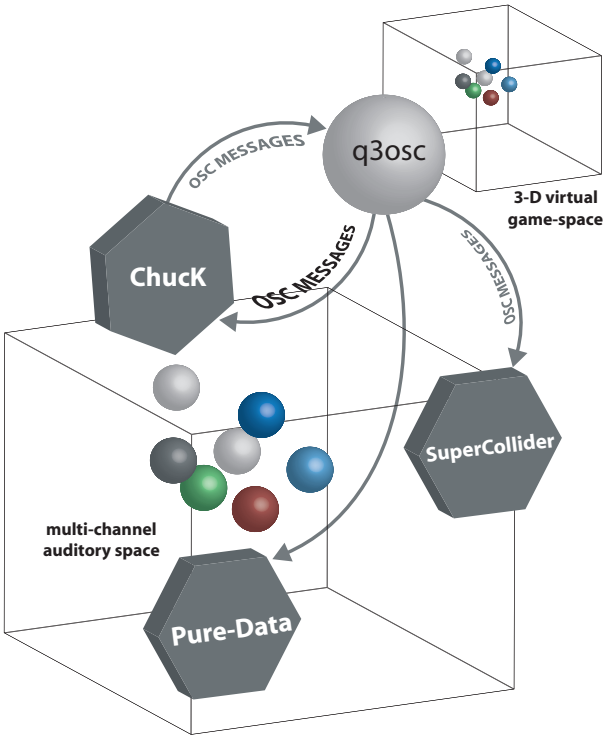

Figure 4. Multiple OSC output streams allow for the simultaneous use of multiple sound-servers utilising various software languages such as Pure Data, SuperCollider or Max/MSP. Q3osc can receive incoming OSC messages on multiple ports allowing for multi-sourced OSC input as well.

sound-object spatialized across an array of real-world speaker locations, such a load-balancing scheme greatly lessens the possibility of audio drop-outs, distortions or clicks generated by an overloaded audio-server. In this same way, multiple audio-servers running completely different audio softwares can be utilized together with little-to-no added complexity.

5.2. Projectile Behaviors and Controls

The use of in-game weapon projectiles as controllable or behaviored entities, each generating their own individual OSC output stream of data representing coordinate position, velocity and other attributes has been achieved by modifying existing code within the game-engine as well as through the creation of several new methods for effecting physics-based changes on entity behaviors. In this manner, not only does a weapon projectile like the "plasma-bolt" - a visually striking blue ball of energy - output its current X,Y, and Z coordinates as OSC messages, but such projectiles also exhibit behaviors such as the ability to bounce off walls (instead of being destroyed by contact with environment surfaces) or the ability to be attracted to individual game-clients as "homing" projectiles.

While in the standard ioquake3 code-base certain server parameters such as global gravity strength or a client's top running speed could be modified by individual game-clients during gameplay using in an in-game "console" window (similar to a Linux/Unix "Terminal" window"), modifications to the lower-level behavioral patterns of projectiles were not accessible during game-play. Weapons fired by individual clients would be assigned a direction vector (with an origin at the client's current position) and a pre-determined velocity for the selected weapon type. The game-server would be sent a message that the weapon had been fired by a particular client and would trace a vector moving at the defined velocity until either a collision occurred - where an entity such as a wall or another game-client was detected to be "hit" - or a pre-determined amount of time expired, causing the entity to destroy itself. These static behaviors and settings were hard-coded into the game engine and could not be modified on-the-fly.

In Figure 5, screen captures of q3osc display a number of modified entity behaviors. In examples I and II, large rotating spherical masses of projectile entities are created by enabling behavior flags which cause all subsequently generated projectiles to a) persist indefinitely, b) bounce from any contact with walls, floors or other parts of the environment itself, and c) constantly update their directional vectors to track any client entity within a given radius. By changing environment variables which control both this radius of attraction and the speed at which the projectiles move, projectiles are easily coaxed into orbits around their targets of attraction. Additionally, by toggling projectile homing behaviors on or off, these dynamic spheres can be made to expand, contract and collapse (as seen in example III). A linear path of homing particles can be seen in example IV, where particles fired by one game client track the position of another game client.

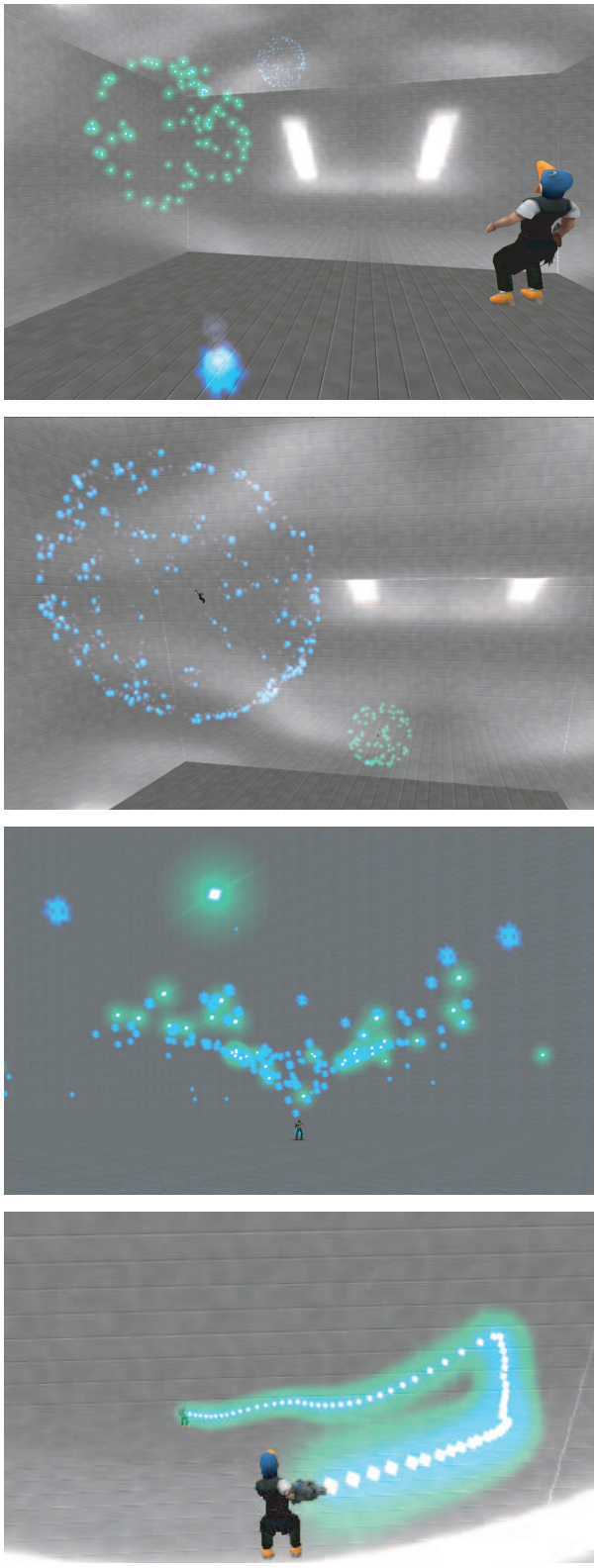

Figure 5. In q3osc, complex patterns of entites can be formed and controlled by performers within the game environment: I) game-client observes orbiting projectiles; II) clients with orbiting projectiles; III) projectiles collapsing on a stationary client; IV) client-to-client homing projectiles.

6. SPACE AND SPATIALIZATION

In standard modes of single-user or networked multi-user gameplay, the focus of both the visual and auditory representations presented to each user has traditionally been wholly user-centric. The user in his or her real-world seat is presented with an illusory visual and sonic representation of the virtual environment complete with signal processings designed to strengthen the related illusions of space, distance and motion. As game-users most commonly listen to game audio output through headphones, stereo speakers, or an industry-standardized multi-channel configuration such as 5.1 or 8.1, all audio processing done in game-engines tends to attempt to create realistic illusions of motion for one user sitting in the sound-system's centralized "sweet-spot". Such individualistic presentation by its very nature restricts the communal sensory experience fostered in the virtual environment from existing anywhere except within the game-world itself. By inverting these traditional models of sound-presentation and by focusing on a space-centric model of sound projection for game-environments, a communal listening experience can be fostered inclusive of all listeners within a shared physical space, including game-users and audience members alike.

In early presentations of q3osc-based sound works, objects within the game-environment have been spatialized across an 8-channel horizontal sound-field [5]. Eight speakers surrounding audience and performers alike were mapped to virtual speaker locations within the game-environment and entity positions in virtual space were spatialized across the sound field by correlating simple distance measures from entity to virtual speaker with speaker amplitude levels. More recent experiments make use of spatialization across three-dimensional soundfields such as the sixteen-channel spherical sound-space in the CCRMA Listening Room [8]. As the output from q3osc can be routed to multiple sound-servers, effectively load-balancing environments populated with large numbers of entities, the use of larger acoustic spaces with greater numbers of point-source speakers - such as the 48-channel multi-tiered Soundlab [15] in Belfast's Queen's University Sonic Arts Research Centre (SARC) - is being considered. Additionally, the use of Ambisonics [9] or VBAP [14] spatialization techniques is being investigated, though the lack of support for either technique in the ChucK language at this time has lowered the priority of this enhancement.

7. SONIFICATION AND MAPPING OF BEHAVIOR AND GESTURE

While the primary direction of the current discourse is aimed at the description of the q3osc interface and the technical issues faced within the project itself, the overwhelming importance of the manner in which q3osc data is and can be sonified as well as the mapping schemae available for composers and performers alike merits significant attention, surely more than can be addressed in the scope of this article. The aim of q3osc is not to define a particular mapping scheme for sonification of game-entity generated data but to facillitate the dialog between virtual action and analog sounding-gesture. That being said, a number of basic sonification mappings being currently investigated are outlined below.

7.1. Projectile Bounces

A direct and effective mapping technique currently used with q3osc is the sonification of projectile bounces, or collisions between projectile entities and environment surfaces. If an in-game flag is set - "g_plasma_bounce" - at the point of contact between a given projectile and environment surface, not only does the projectile persist (i.e. not be destroyed) and bounce off the surface, but a "bounce" OSC flag is set and transmitted. One demonstration mapping currently makes use of a simple ChucK SinOSC oscillator enveloped and fed through a JCRev Chowning-Reverb. By mapping coordinate data for a single plane - say the X axis - received in the same OSC message as the bounce flag to a desired musical scale, each bounce event can be made to trigger simple notes. As a possible example of additional non-spatial bounce mappings, speed of the projectile can be mapped to velocity or amplitude of each bounce-triggered note-event, Y-axis coordinate data can be mapped to note-duration and Z-axis coordinate data can be mapped to length of decay on the reverb (if the reverb were implemented locally to each shred). Additionally, at any such point of impact, data points such as angle-of-incidence, distance-traveled, and duration of existence all are easily extracted.

7.2. Entity-to-Entity Feedbacks

One interesting and evocative sonification technique currently being investigated tracks the relative positions of clients within the virtual-environment and sonifies the distance between them as proportionally increasing or decreasing levels of dissonant and jarring pitched feedback. As clients move within a pre-defined distance threshold of one another, this relational feedback cue not only clearly signals the proximity of another client but through the use of individual pitches uniquely assigned to each client, can be used as a non-visual method of tracking relative client positions. To encourage clients to group together in potentially large virtual environments, the cue can be inverted as well, sounding when clients move too far from one another.

7.3. Projectile Masses and Grouped Mappings

When a set of projectiles are attracted to a client in homing behaviors, each projectile essentially becomes a single unit of a collective mass each following the same behavioral rules. As such, masses or clusters of projectiles behaving in this manner can be sonified together as a larger super-entity of sorts. One proposed mapping scheme aims

to sonify such masses as a set of sound-grains of a sampled and granulated sound. With further development of the bi-directional OSC messaging mentioned previously, more complex algorithmic behaviors and group mappings are planned.

8. VISUAL PRESENTATION METHODS

The visual impact of q3osc's virtual performance environment is an integral component of its presentation. As a key goal in the project is to create methodologies for the clear and identifiable transmission of virtual performance gesture into sonified and spatialized output, it is important that audience members be able to watch actions performed within the virtual environment. Currently, q3osc environements can be projected upon a single wall of a concert hall (usually behind the performers and in front of the audience) with virtual camera angle and position controlled by a single in-game client viewpoint. As the ioquake3 engine allows for the existence of "spectator" clients, who have the ability to observe environments without being seen or being able to interact with clients or the environment itself, this visual model is the easiest to implement. From another perspective, the ioquake3 engine supports a third-person view for any particular game-client, with variable angle, camera distance or even an automated camera rotation mode. While these methods are currently being used, an ideal visual presentation of the system would make use of a fully-immersive 360-degree projection room, where the walls of a concert space are covered in projections of the virtual space. Towards this goal, the use of such spaces like the Recombinant Media Lab in San Francisco are being pursued.

9. NETWORK LATENCY AND THE EFFECT ON GESTURE

While a complete description of the workings of the Quake 3 client-server model is outside the scope of this discussion, it should be noted that due to the possibility of clients and connections respectively running at extremely different processor speeds and with varying levels of available network bandwidth, a predictive model - where the game-server coordinates clients of varying connection and processor speed by anticipating entity positions and rapidly correcting those positions to all connected clients - of synchronizing data between server and multiple clients is utilized. While this approach allows for the possibility of occasional disjointed visual motion (in cases of extreme network lag), as the game-server acts as the ultimate arbitor or event-timing, even clients with relatively high server ping times (in the 100s of ms) are constantly being realigned with the server. For current uses of q3osc, running on local wired or wireless networks, the latencies experienced are not noticed.

As the q3osc model is tested further to take into account more use of remote "off-site" connected game-clients, more detailed study of the effect of network latencies and intended coordinated virtual gesture will be needed. To determine the perceivablilty and effect on performance of such latencies, testing procedures similar to those outlined in Chris Chafe's 2004 networked ensemble performance studies [1] will be investigated.

10. CONCLUSIONS

Virtual game-environments repurposed as extensible multi-user networked performance environments offer performers and audiences alike rich platforms for musical expression. While there certainly exist a number of equally powerful solutions for the creation of virtual environments - such as Blender, Ogre3D and the open-source engines used in the earlier Quake and Quake II games - the Quake III engine's history as a successful commercial gaming platform affords users a massive pre-existant code base, a strongly user-supported and open-sourced community-driven development environment and toolset, and relatively low financial and temporal barriers to entry.

11. REFERENCES

[1] Chafe, C., Gurevich, M., "Network Time Delay and Ensemble Accuracy: Effects of Latency, Asymmetry," In *Proceedings of the AES 117th Conference*, San Francisco, 2004.

[2] Chafe, C., Wilson, S., Leistikow, R., Chisholm, D., Scavone, G., "A Simplified Approach to High Quality Music and Sound Over IP," In *Proceedings of the COSTG6 Conference on Digital Audio Effects (DAFx-00)*, Verona, 2000.

[3] Davis, T. and Karamanlis, O. "Gestural Control of Sonic Swarms: Composing with Grouped Sound Objects." In the Proceedings of the 4th Sound and Music Computing Conference.

[4] Furukawa, K., M. Fujihata, and W. Muench, http://hosting.zkm.de/wmuench/small_fish.

[5] Hamilton, R. "maps and legends: FPS-Based Interfaces for Composition and Immersive Performance" In *Proceedings of the International Computer Music Conference.*, Copenhagen, Denmark, 2007.

[6] id Software, http://www.idsoftware.com, *as viewed 2/2008.*

[7] ioquake3 Project Page, http://www.ioquake3.org, *as viewed 2/2008.*

[8] Lopez-Lezcano, F. and C. Wilkerson. "CCRMA Studio Report" In *Proceedings of the International Computer Music Conference.*, Copenhagen, Denmark, 2007.

[9] Mahlam, D. and A, Myatt, "3-D Sound Spatialization using Ambisonic Techniques" Computer Music Journal, 19;4, pp 58-70, Winter 1995.

[10] "Max/MSP", Cycling '74, http://www.cycling74.com, *as viewed 2/2008*.

[11] McCarthy, J. "SuperCollider", http://supercollider.sourceforge.net, *as viewed 2/2008*

[12] Oliver, J. and Pickles, S. *q3apd*, http://www.selectparks.net/archive/q3apd.htm, as viewed 1/2008.

[13] Puckette, M. 1996. "Pure Data." In *Proceedings of the International Computer Music Conference.* San Francisco, 1996, pp. 269-272.

[14] Pulkki.V. "Virtual sound source positioning using vector based amplitude panning." Journal of the Audio Engineering Society, 45(6), June 1997, 456-466.

[15] SARC Soundlab, http://www.sarc.qub.ac.uk/main.php?page=building&bID=1 , as viewed 2/2008.

[16] Trueman, D., P. R. Cook, S. Smallwood, and G. Wang. "PLOrk: Princeton Laptop Orchestra, Year 1" In *Proceedings of the 2006 International Computer Music Conference (ICMC)*, New Orleans, U.S., November 2006.

[17] Wang, G. A. Misra, and P.R. Cook. "Building Collaborative Graphical interFaces in the Audicle" In *Proceedings of the International Conference on New Interfaces for Musical Expression.*, Paris, France, 2006.

[18] Wang, G. and P. R. Cook. "ChucK: A Concurrant and On-the-fly Audio Programming Language" In *Proceedings of the International Computer Music Conference.*, Singapore, 2003.

[19] Wang, G. and P. R. Cook. "The Audicle: A Context-sensitive, On-the-fly Audio Programming Environ/mentality" In *Proceedings of the International Computer Music Conference.*, Miami, USA, 2004.

[20] Wright, M. and A. Freed. "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers" In *Proceedings of the International Computer Music Conference.*, Thessaloniki, Greece, 1997.

[21] Lancaster, S., "The Aesthetics and History of the Hub: The Effects of Changing Technology on Network Computer Music", Leonardo Music Journal, Vol. 8, pp. 39-44, 1998.