# An Interactive 3D Network Music Space

Chad McKinney
University of Sussex, UK
C.Mckinney@sussex.ac.uk

Nick Collins
University of Sussex, UK
N.Collins@sussex.ac.uk

## ABSTRACT

In this paper we present Shoggoth, a 3D graphics based program for performing network music. In Shoggoth, users utilize video game style controls to navigate and manipulate a grid of malleable height maps. Sequences can be created by defining paths through the maps which trigger and modulate audio playback. With respect to a context of computer music performance, and specific problems in network music, design goals and technical challenges are outlined. The system is evaluated through established taxonomies for describing interfaces, followed by an enumeration of the merits of 3D graphics in networked performance. In discussing proposed improvements to Shoggoth, design suggestions for other developers and network musicians are drawn out.

## Keywords

3D, Generative, Network, Environment

## 1. INTRODUCTION

Shoggoth is a new network music program for real time group performance with members distributed over potentially global distances. As a reference to the strange protoplasmic beings described in H.P. Lovecraft's *At the Mountains of Madness* [16], Shoggoth allows users to reshape polymorphic terrains to create generative music in collaboration. The program is designed with a user interface that is both functional and highly visual. The interface design allows for an aesthetically pleasing presentation that serves to both enhance communication in the ensemble as well as offer a clear presentation for the audience. This is important because performances with physically separated ensembles present a unique stage presence where parts, and possibly all, of the group can only be represented through digital media. The separation in distributed ensembles amplifies several issues in traditional computer music performance, such as a lack of correlation between physical effort and sonic results. Furthermore, distributed ensembles lose fundamental components of communication such as visual cues and gestures.

These issues are not new [4, 36] and there is a growing range of techniques and technologies which seek to mitigate or embrace these features of electronic music. Controllers and interfaces are a popular solution for computer musicians to reestablish or reimagine the performance characteristics of traditional instrumentalists [25, 28, 38]. These interfaces lose value in networked performances if members are in different locations from each other or from an audience. Concerts in many forms, from experimental computer music in smaller clubs to popular music stadium shows, are now commonly performed with accompanying visuals to augment stage presence [32, 5]. While there may be useful benefits from adding the visual medium, if the presentation isn't communicative of the non-present performers, their contributions will be deemphasized or lost entirely. For this reason network performances are often realized using video and audio streaming between performance sites [6, 31, 13]. Latency and quality of connectivity are ever present concerns, and if performers aren't using traditional instruments or physical controllers then the same issues regarding computer music performance outlined earlier will still be present.

This is where virtual spaces can serve a useful role. Networked performances, distributed or not, that are performed in virtual spaces communicate performers' efforts while simultaneously increasing ensemble communication. Consideration must be given to both usability and presentation and a balance must be struck to facilitate a successful performance space. Video games are a natural source of inspiration, with their sprawling and detailed worlds, the largest of which are developed utilizing multi-million dollar budgets and over a period of several years. Music has been an important component of video games since the beginning and game music has become ingrained in our culture. Despite this, music has usually severed a secondary role, similar to it's usage in movies to set the tone of a scene or level. Game music is commonly adaptive, not interactive, because there is usually no direct connection between player actions and changes in the music [8]. Sound effects are the actual interactive components in a game, such as the triggering of a jump sound based on a button press. There is often some correlation between game state, such as the adjustment of tempo according to a game boss's life.

There is a history of utilizing games or game like worlds in music and sound art. A common approach has been to appropriate or modify an existing game for use in a work. Cory Arcangel's Nintendo cartridge hacks, including his celebrated *Super Mario Clouds*, and Tom Bett's glitch inducing quake engine modification *QQQ* are two examples of how an existing game can be appropriated to produce results never intended by their designers [8]. Both modify the source code for a game, fundamentally altering it's logic, and creating something new. Not all game appropriations are as subversive. Rob Hamilton's work *Maps and Legends* [15, 14] built using *q3apd* [26], a Quake III modification by Julian Oliver and Steven Pickles, is a network composition performed in virtual space. Player states such as position and view angle, and weapon selection, as well as certain actions such as jumping and firing are mapped using OSC to control a Pure Data patch [20]. These mappings allow Hamilton to use the core logic of the Quake engine as the framework for a networked virtual performance.

In Shoggoth, instead of using an existing game engine, a new one was written specifically for the purpose of network music performance. This allowed for the customization of a system that attempts to find the right balance between usability, musical control, and visual aesthetics; this paper serves to document those efforts. In the following section we detail the system design and philosophy, as well as some technical aspects of the implementation. Next the system is categorized using established frameworks with a subsequent examination of the role of virtual spaces in music performance. In conclusion some initial findings are reported along with useful information for other developers and musicians.

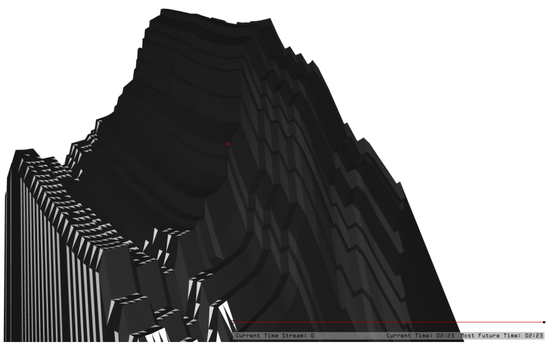

Figure 1: One possible terrain shape in Shoggoth.

## 2. SYSTEM DESIGN AND DEVELOPMENT

Shoggoth is a network music program, but video games were a large inspiration for the design. Previous forays into interfaces for network music demonstrated increasingly graphical interfaces, often accompanied by a separate visualization program. This approach has worked well, although it also meant that audiences were not presented with the same visual information as the performers. With Shoggoth we attempted to create an interface that is aesthetically rich while functioning as the interface through which the musicians collaborate.

### 2.1 The Interface

Shoggoth is written in C++ and uses the Cinder framework [7] for the graphics implementation. On startup the view comprises of a grid of flat black square islands suspended in white space. Users can fly around the space by employing controls similar to a first person shooter (FPS) game, but there is no gravity or physics. The flat grids are vertex buffer objects (VBOs) [24] comprising of a triangle mesh bound with important data such as color and id numbers. The grids can be manipulated using a selection of number keys that trigger a morphing animation into various shapes dependent on one of several generative processes. These processes are each based on a particular algorithmic model, enumerated as as (0) Blank, (1) Diamond Square, (2) Cellular Automata, (3) Strange Attractor, (4) L-System, and (5) Flocking. Each process results in a height map and a series of intermediate steps are constructed between the existing mesh and the new version. Using a queued update system the mesh is updated each frame, incrementing through a thirty step animation list, until the final version of the mesh is reached. Earlier versions of Shoggoth did not have animations between meshes and for that reason mesh transitions were jarring, which inspired the added feature. Animations not only create smooth changes and striking visual effects, but also allow for the audio sequencing to follow the interpolation as well.

A triangle can be selected, using 3D picking [34], from the grid of a terrain mesh for sequence path creation or manipulation. 3D picking is a technique that allows users to select something in 3D space using 2D coordinates, usually via a mouse controlled camera view. 3D picking was implemented in Shoggoth using a graphics technique whereby the terrain meshes are rendered at a lower resolution into a frame buffer object (FBO) [33], which is never shown to the user, and each triangle in each terrain mesh is colored according to a global identification system. When a picked triangle is requested, the color of the pixel in the exact center of the FBO is selected and then only has to be translated from an RGBA (reg, green, blue, alpha) value into an unsigned integer, resulting in the selected triangle's global identification number. This proved to be invaluable as each terrain contains over 10,000 triangles and previous attempts using ray casting were unusably slow.

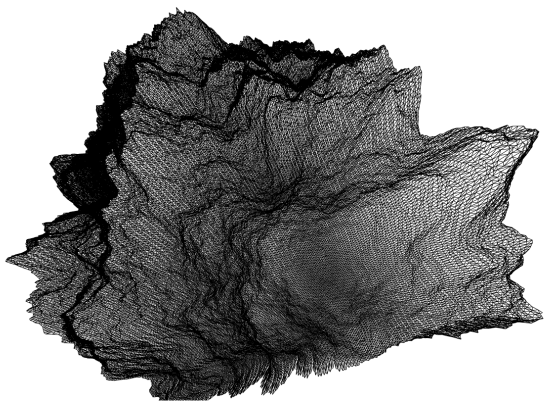

Figure 2: Wireframe render for an island terrain, demonstrating the triangle mesh and high polygon count.

A path can be created from a sequence of triangle picks, and once outlined, a read head immediately follows on the path, triggering and modulating monophonic synth instances. A triangle in the mesh of an island has two possible states: black (inactive) or white (active). If the triangle is active when a read head passes over it, then a coordinating synth is triggered, resulting in an opening of the envelope gate and an update to the parameters of the synth according to the triangle's height and location in the grid. Triangles are activated or deactivated according to a similar set of generative processes as the height map, and are triggered using the same number keys, but with the shift key pressed as well.

Player representation and communication are important in network music performance and Shoggoth has some simple, but effective, designs to facilitate them. Players are represented using minimalist tetrahedron models, which aren't complicated, but align well with the triangle based theme of the islands. Position and rotation information is mapped allowing performers to see not only where each other are, but what they're looking at, and the immediate results of their actions. This is an upgrade from the authors' previous systems where either no representation was made or only position data was represented. A chat system has been created to allow for communication, both with the other performers and the audience, and uses a multi-player game style 2D overlay.

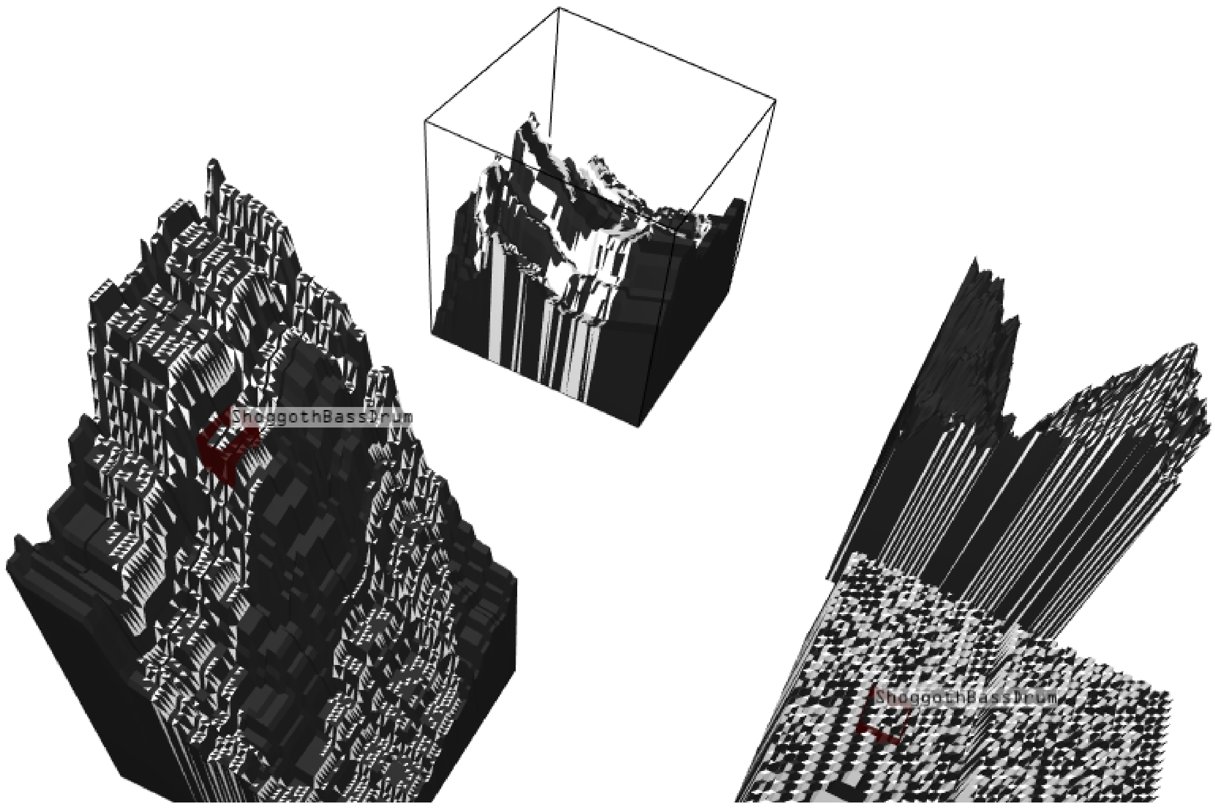

Figure 3: Multiple terrains with sequences.

## 2.2 Sound Design

Sound in Shoggoth is implemented using the SuperCollider [37] libscsynth library in conjunction with libsc++ [22] to create an internal several built natively into the C++ application. Because the server is built internally, no external messaging are necessary, and all communication with the scsynth server and Shoggoth occur through native function calls. Shoggoth will fail completely without any hanging servers in the event of a crash, where as if the server were running on the local machine this would not be the case. Maintaining independence of the sound server, language, and now the IDE is a favorable characteristic of SuperCollider as an audio language. That level of independence is not favorable when distributing a program to users who may not be knowledgeable about the subtleties involved with multiple processes.

Synth design in Shoggoth is focused on the usage of wave terrain synthesis [29]. Each synth definition utilizes at least one wave terrain oscillator that reads a buffer filled with the same 2D height map that defines the shape of the terrain that the synth's sequence resides on. This is a essential feature because it allows the terrains to effect not just the sequential triggering of synth instances or the modulation of synth parameters, but also to define the most fundamental components of the synths' timbre. Each generative process, such as the cellular automata, have a characteristic harmonic palette that forges a strong connection between the visuals and the sound. Furthermore, when the island meshes morph into new forms, the animation effects not only the visuals display, but also updates any running synths as well, creating a dramatic timbral shift.

Synths definitions must be written and edited in SuperCollider which does create dependancy for development, as the version of SuperCollider that the synth definitions are compiled in must be the same that Shoggoth has been built against. This might change given development in the libsc++ library that could allow for native or scripted synthdef compilation. Even given this dependancy, SuperCollider is an excellent choice for sound design because it has an established code base with years of active development and supplies a well defined and terse interface for synthesis. Shoggoth can be used to create a wide range of sonic output, but given the looping sequential infrastructure and the often aggressive waveforms produced by the wave terrain synthesis, rhythmic noise is the most natural end result. While this style of music may not appeal to all, generative and networked music audiences are often interested in more experimental music.

## 2.3 Networking

Open Sound Control [39] messaging is the back bone for the networking in Shoggoth. The core networking interface is implemented using OSCpack [27] to create and receive OSC packets. Additionally, Shoggoth uses the OSCthulhu [23] server and client framework because direct peer to peer networking can cause a multitude of issues stemming from packet loss. OSCthulhu uses a multi-player video game style synchronization scheme which is ideal for a program such as Shoggoth that defines networking not as a sequence of messages, but instead as a collection of states that are updated across the network. To implement networking using OSCthulhu, sync objects are created on the server, each with any number of sync arguments. When an object needs to be updated a message is sent from the local client to the OSCthulhu server, which updates itself and immediately updates all the other clients in the network. Additionally the server has a regular update cycle that updates all of the clients to the current world state. Because of this server based synchronization lost packets are quickly recovered from and the network is continuously realigned.

In Shoggoth there are eight types of information networked: Player position, player orientation, terrain height maps, terrain step grids, sequence positions, sequence sizes, synth selections for sequences, and chat. This group contains a wide variety of data from character strings to high volume meshes. The terrain meshes proved to be the most challenging to network initially. Synchronizing 10,000 triangles with 3 points each as well as the step grid was daunting, inspiring odd attempts to reduce bandwidth such as using LZMA compression [30]. The solution we found was elegant, but limiting. Instead of manually synchronizing each triangle, instead the settings and random seed used to generate a given terrain mesh and step map were synchronized, allowing for an incredibly small amount of information to guarantee state across the network. The drawback is that we had to remove manual deformation of the terrains, leaving only the generative processes able to create and manipulate the islands. This changed the nature of the performance from guided intentionality to fast experimentation, but the reduction in bandwidth was extremely beneficial.

Player position and orientation were easy enough to network, but using seven arguments per user to define them (3 for position, 4 for quaternion defined rotation) sometimes generated asynchronous updates for their individual components and unnecessary traffic. This problem led to the use of bitpacking to package updates together. Using bitpacking the X/Y/Z components of the position of a player can be packed into a single integer value, as well as the W/X/Y/Z components of their rotation, reducing traffic while simultaneously enforcing unified updates. The same technique was used with the aforementioned island states. All the settings including the process number and random seed are packed into a single integer so that there is a guaranteed success or failure of an update. This prevents scenarios where only a portion of the information needed to update an island is received, while the others might be lost, resulting in an incorrect state. Because of these efforts, Shoggoth's networking is precise and fast despite the large amount of information represented on screen and even while using low quality wireless connections.

## 3. TAXONOMY AND EVALUATION

Placing network music systems within the framework of existing taxonomies is useful for analysis and comparison. We have chosen three frameworks for this assessment, Andrew Hugill's internet music taxonomy [17], Golo Föllmer's twelves types of net music [12] and Thor Magnusson's epistemic dimension space [21]. Beginning with Hugill's Internet music taxonomy, Shoggoth sits well in the second category of music that is *Created or Performed in Virtual Environments, or Uses Virtual Instruments.* Convincing arguments could be made that Shoggoth also facilitates music that *Uses the Internet to Enable Collaborative Composition or Performance*, but the defining features of Shoggoth align more closely with most of the pieces that could fit into Hugill's second category than his fourth.

Föllmer's taxonomy is much more complicated than Hugill's and requires more consideration. There are many similarities between Shoggoth and Föllmer's description of the *Algorithmic Installations* type, but the emphasis on an installation as opposed to performance does not afford an easy fit. With that consideration, the *Performance* cluster, number 5, is the most natural, leaving a choice between *Network Performances* and *Staged Projects* types. Shoggoth makes no use of librettos or text and therefore *Staged Projects* makes little sense, leaving type K *Network Performances* in cluster V. *Performance.*

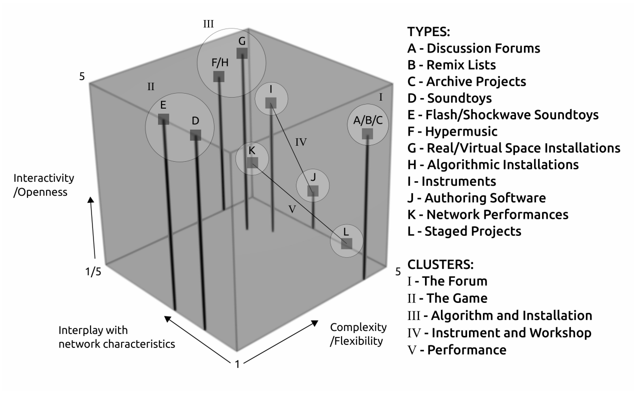

Figure 4: Föllmer's Spatial order of the twelve types of Net music (A–L) in relationship to the three dimensions "interplay with network characteristics", "interactivity/openness"and "complexity/flexibility", scaled from 1 to 5. Types are rated on the basis of averaged ratings of single projects. In mapping the types, the clusters I–V are formed.

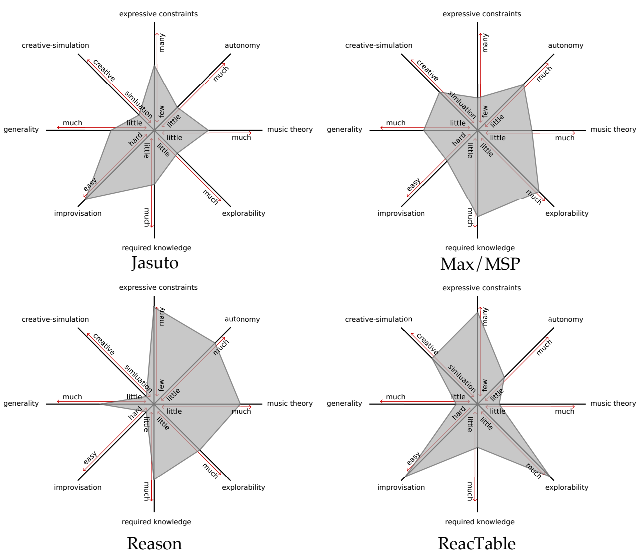

Figure 5: Jasuto, Max/MSP, Reason, and the ReacTable plotted in Magnusson's epistemic dimension space.

Magnusson's epistemic dimension space is not network music specific, but it is a useful to for analyzing and comparing musical interfaces. Eight parameters are spread across a 2D plane allowing for a mapping of the epistemic qualities of an interface. The interesting shapes produced, while somewhat arbitrary, do afford a fast comparison between multiple interfaces, and a quick glance reveals that Shoggoth is most similar to the ReacTable [19], of the list provided. While Shoggoth is marked as having more autonomy than the ReacTable (because of the extended use of generative processes), they both impose significant constraints on expression, while lacking generality and inherent music theory constructs. Instead, both the ReacTable and Shoggth emphasize improvisation in a creative and unique interface. In contrast, Reason contain more music theory infrastructure, and Max/MSP has more depth of explorability.

On February 24th, 2013, Shoggoth was premiered at the Network Music Festival in Birminghan, UK [1]. Network bandwidth over the wireless internet connection was discovered to be an issue during sound check and for the performance networking of the player avatars had to be removed. It was also discovered that the first person camera controls

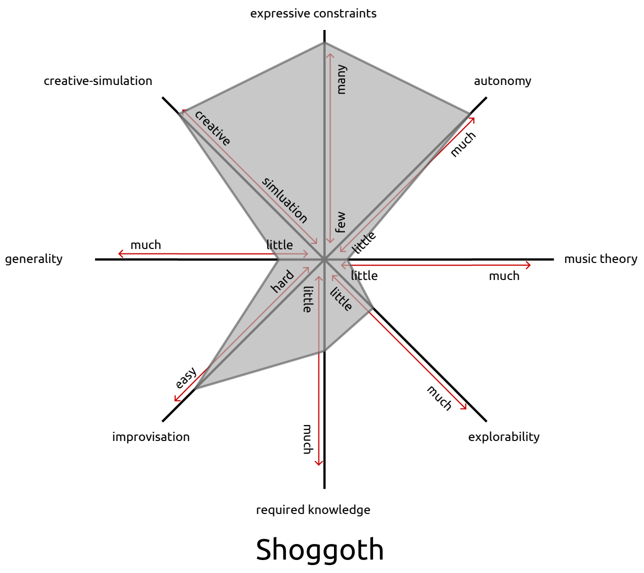

Figure 6: Shoggoth in the epistemic dimension space.

did not work correctly with a projector connected to the system, which restricted the movemen in the virtual space. Both problems have been subsequently corrected. The performance well received by the audience, with informal comments rating it highly in the festival.

## 4. PERFORMANCE IN VIRTUAL SPACE

Video game culture provides a useful reference for digital spectator events. The youngest generations have been raised in an era of video games and the internet, giving rise to online E-Sports such as *Star Craft* and *League of Legends* [35, 18]. These games are not just for bragging rights and the winners stand to win hundreds of thousands of dollars in front of thousands of fans [3]. Virtual music [11] performances have not yet reached this level of acceptance, but the concept has been proven that there is a potentially large audience for virtual performance.

There are many similarities between an online battle and an online musical performance (perhaps even an online music battle). The two are often group events and the depiction of embodiment is important to spectators. But where games have concrete goals and rules that dictate their achievement, musical performances have compositions and improvisations with a wide range of constraints and goal orientation. There are other considerations, such as the embodiment of the performers and their portrayal, or lack thereof, of physical and emotional state. These have an important role in how a performance is perceived by an audience, and by the performers themselves [9]. If virtual music performances are to attain the popularity of virtual sports, more work will need to be put into the systems and infrastructure that supports those performances.

Online games such as *Star Craft* have budgets that rival hollywood movies, but more importantly there is a depth to the software that is simply missing in network music systems. For example the players in Shoggoth are represented as simple tetrahedron with only position and rotation as defining features. Characters in a video game on the other hand can have hundreds of animations. The amount of detail in the textures, meshes, lighting, and shading in a large game dwarfs the efforts of even the most ambitious network musician. Important steps can be taken to improve the situation, such as the development of open and abstracted tools sets to reduce duplicated work and the adoption of new skills such as modeling and animation. Perhaps the most useful step is to consider how some independent developers manage to compete against even the largest games despite tiny budgets and thin development teams. Games such as *Minecraft* [10] attract massive audiences despite these issues because they use resources wisely, often employing minimalist or generative techniques, and create sophisticated and stylized game and art designs that don't require large resources.

## 5. REFLECTIONS ON DEVELOPMENT

After months of work, and many challenges along the way, Shoggoth has reached an initial release and is performance ready. The program is fairly stable and a recent feature lock down means that future development will be concerned with bug fixing and system efficiency. Other network musicians or software designers will benefit from learning about a few of the challenges throughout Shoggoth's development.

An important consideration is when to write a completely new engine from scratch or in contrast, recognizing when an existing engine is a viable option. Writing a new engine should not be considered lightly, and indeed the vast majority of the time spent developing Shoggoth was put into building basic functionality such as FPS style camera controls, mesh generation, and a chat system. The Quake III engine mentioned earlier, or something similar such as Ogre 3D [2], will already have this kind of functionality built in, and will greatly reduce your development time. Only if something requires a unique feature (in the case of Shoggoth, the polymorphic terrains) should engine development be considered. Furthermore, if the decision is made to write an engine, the creation of an abstracted framework or library will benefit subsequent development. Shoggoth is written without such abstractions and for this reason much of the code base is not easily portable to other projects. For that reason, this development cycle inspired the creation of an engine with many of the basic functions underneath a network music program like Shoggoth implemented using a clean and abstracted interface.

Another large problem facing development was a lack of focus during some periods of design. Experimentation is a useful technique in music software design, but some amount of planning will minimize lost time. For example, in Shoggoth the fundamental way in which performers used the the interface was not clearly defined until well into development, resulting in several abandoned efforts and wasted time. From a musical perspective, a lack of focus is also problematic because it creates a moving target for sound design, stunting the growth of the system's musical identity. Finally, allowing other musicians an opportunity to use and evaluate the project starting early in the process will help identifying problems not just with the code base, but also the design and vision of the project. For Shoggoth, that external assessment was not introduced early enough in the process, leading to some of the issues mentioned earlier. Moving forward the main concern is to perform with the interface, find and fix problems in the system, and to streamline the project where possible. Further features and a potential user study will follow the initial performances.

## 6. CONCLUSIONS

In this paper we have presented Shoggoth, a new interactive system for performing networked generative music within a 3D space. We began first by discussing some of the unique problems that network music performances face, especially with regards to distributed ensembles. Next, the technical implementation of graphics, audio and networking was discussed and Shoggoth examined with respect to three established taxonomies, and associated contextual considerations. Finally, we concluded with some reflections on the relationship of gaming and virtual music making, and suggestions for other developers who may be interested in some lessons for their own network music software design.

## 7. REFERENCES

[1] Network music festival, 2013. Information available at: `http://networkmusicfestival.org/` [Accessed February 8, 2013].

[2] Ogre 3d, 2013. Available from: `http://www.ogre3d.org/` [Accessed February 11 2013].

[3] W. Benedetti. Taipei assassins triumph in 'league of legends' world finals, 2012. Available from: `http://www.nbcnews.com/technology/ingame/taipei-assassins-triumph-league-legends-world-finals-1C6448579` [Accessed 8 Feb 2013].

[4] L. Berio and R. Dalmonte. *Intervista sulla musica.* Laterza, 2007.

[5] K. Brougher, J. Zilczer, C. Museum of Contemporary Art (Los Angeles, H. Museum, and S. Garden. *Visual music: synaesthesia in art and music since 1900.* Thames & Hudson, 2005.

[6] C. Chafe, S. Wilson, A. Leistikow, D. Chisholm, and G. Scavone. A simplified approach to high quality music and sound over ip. In *In Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00*, pages 159–164, 2000.

[7] Cinder Community. Cinder, 2013. Available from: `http://libcinder.org/` [Accessed 8 Feb 2013].

[8] K. Collins. *Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design.* Mit Press, 2008.

[9] D. Deutsch. *The Psychology of Music.* Elsevier Science, 2012.

[10] N. Development, 2013. Available from: `https://minecraft.net/` [Accessed 11 February 2013].

[11] W. Duckworth. *Virtual Music: How the Web Got Wired for Sound.* Taylor & Francis, 2013.

[12] G. Föllmer. Electronic, Aesthetic and Social Factors in Net music. *Org. Sound*, 10(3):185–192, Dec. 2005.

[13] S. Gresham-Lancaster. Is there no there there? video conferencing software as a performance medium. In *Music in the Global Village Conference*, 2007.

[14] R. Hamilton. Maps and legends: Designing fps-based interfaces for multi-user composition, improvisation and immersive performance. In R. Kronland-Martinet, S. Ystad, and K. Jensen, editors, *Computer Music Modeling and Retrieval. Sense of Sounds, 4th International Symposium, CMMR 2007, Copenhagen, Denmark, August 27-31, 2007. Revised Papers*, volume 4969 of *Lecture Notes in Computer Science*, pages 478–486. Springer, 2007.

[15] R. Hamilton. Maps and legends: Fps-based interfaces for composition and immersive performance. In *Proceedings of the 2012 International Computer Music Conference*, pages 344–347, 2007.

[16] H.P. Lovecraft. *At The Mountains of Madness.* Arkham House, 1931.

[17] A. Hugill. Internet music: An introduction. *Contemporary Music Review*, 24(6): 429–437, 2005.

[18] D. Jin. *Korea's Online Gaming Empire.* Mit Press, 2010.

[19] S. Jordà. The reactable: Tabletop tangible interfaces for multithreaded musical performance. *Revista KEPES*, 5(14): 201–223, 2009.

[20] M. Puckette and Pure Data Community Developers. Pure Data, 2012. Available from: `http://puredata.info/` [Accessed 12 March 2012].

[21] T. Magnusson. An Epistemic Dimension Space for Musical Devices. In *Proceedings of the 2010 conference on New interfaces for musical expression*, pages 43–46, 2010.

[22] C. McKinney. libsc++, 2013. Available from: `https://github.com/ChadMcKinney/libscpp` [Accessed 8 Feb 2013].

[23] C. McKinney and C. McKinney. Oscthulhu: Applying video game state based synchronization to network computer music, 2012.

[24] T. McReynolds and D. Blythe. *Advanced Graphics Programming Using OpenGL.* The Morgan Kaufmann Series in Computer Graphics. Elsevier Science, 2005.

[25] J. M. Morris. Structure in the Dimension of Liveness and Mediation. *Leonardo Music Journal*, pages 59–61, 2008.

[26] J. Oliver and S. Pickles. qa3pd, 2002. Available at `http://julianoliver.com/q3apd/`[Accessed 8 Feb 2013].

[27] oscpack, 2013. Available from: `http://www.audiomulch.com/~rossb/code/oscpack/` [Accessed 8 Feb 2013].

[28] P. Rebelo. Haptic sensation and instrumental transgression. *Contemporary Music Review*, 25(1-2):27–35, 2006.

[29] C. Roads. *The Computer Music Tutorial.* Mit Press, 1996.

[30] D. Salomon. *Data Compression: The Complete Reference.* Molecular biology intelligence unit. Springer-Verlag London Limited, 2006.

[31] A. A. Sawchuk, E. Chew, R. Zimmermann, C. Papadopoulos, and C. Kyriakakis. From remote media immersion to distributed immersive performance. In *Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence*, ETP '03, pages 110–120, New York, NY, USA, 2003. ACM.

[32] J. Sexton. *Music, Sound and Multimedia: From the Live to the Virtual.* Music and the Moving Image Series. Edinburgh University Press, 2007.

[33] D. Shreiner and B. Group. *The Framebuffer.* OpenGL Series. Pearson Education, 2009.

[34] D. Shreiner and B. Group. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1.* OpenGL Series. Pearson Education, 2009.

[35] T. Taylor. *Raising the Stakes: E-Sports and the Professionalization of Computer Gaming.* Mit Press, 2012.

[36] D. Wessel and M. Wright. Problems and prospects for intimate musical control of computers. *Computer Music Journal*, 26(3):11–22, Sept. 2002.

[37] S. Wilson, D. Cottle, and N. Collins, editors. *The SuperCollider Book.* MIT Press, Cambridge, MA, 2011.

[38] J. Wilson-Bokowiec and M. A. Bokowiec. Kinaesonics: The intertwining relationship of body and sound. *Contemporary Music Review*, 25(1-2):46 –57, 2006.

[39] M. Wright, 2002. Open sound control 1.0 specification. Available from: `http://opensoundcontrol.org/spec-1_0` [Accessed 2 May 2010].