Creative Performance in Play: A Synthesis of Audio, Visual and Narrative for Maximising Expressive Potentials in Gameplay

Alexander James Thumm

B.Mus Sonic Arts (Hons) University of Adelaide, 2013

Submitted in fulfilment of the requirement for the degree of

Doctor of Philosophy

Elder Conservatorium of Music: Sonic Arts

Faculty of Arts

The University of Adelaide

April 2017

Part 1

Exegesis

Part 2

Appendices

Table of Contents

Bibliography

Part 2: Appendices

Abstract

This thesis explores expressive audio-visual performance in video games. It has been developed via the author's creative and technical background in the field of electronic music. Primary aims of this research are:

- Introduce key electronic music concepts into the field of video games in order to make gameplay as expressive as electronic music performance, by offering greater potential for original creativity in gameplay, and a more co-creative experience
- Compare expressive attributes inherent in electronic music, to audio, visual and narrative elements of gameplay. Thus find analogs in those areas where similar levels of expressiveness can be developed and integrated as co-creative gameplay
- Generate a conceptual framework, technical realization, and creative realization (playable game providing narrative framing for performative interactions) to contribute to the field of expressive and co-creative performance in games

This research is primarily intended as a contribution to the field of video games. By beginning with a small scale-focus on moment to moment gameplay this conceptual framework engages with expressive potentials already existent in the field of electronic music. By mapping dynamic gameplay parameters (e.g. avatar movement, proximities, cursor interactions) to musical parameters (e.g. DSP effects settings), a gameplay prototype was developed, rehearsed and iterated upon. This allowed for the development of performance objects (e.g. synthesizers, camera filters, etc), iterative refinement of any given object towards more expressive gameplay, and development of inter-object-relationships.

The combination of a large diversity of objects, and a small set of data-flow languages facilitates a large array of interdependent routings between audio, visual and narrative functions. Expressive potentials of audio-visual gameplay combine with narrative context to enable a co-creative experience for the player where they can situate themselves as an interdependent unit in the shifting contexts of performative play.

List of Tables

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.
I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Alexander Thumm

Signed:_____

Acknowledgements

Thanks to the University of Adelaide, the Elder Conservatorium of Music and the Sonic Arts department for supporting this research. Special thanks to my supervisor Stephen Whittington for guidance and insights throughout, and to Kimi Coaldrake for all manner of support and encouragement. Thanks to Tracy Fullerton and Chanel Summers at the University of Southern California's Game Innovation Lab for supporting my stay as a Visiting Scholar.

Technical and Creative Acknowledgements

Development Software

- Unity: Game development environment
- External code libraries used in Unity (see Appendix C for details)

Creative Contributors

- Angus Barnacle: Pre-composed background music and sound effects in Story Mode (Wwise technical implementation done by the author)
- Alvaro Rodriguez: Character concept art
- Dana Chayes: Character models and character animations
- Ben Roach: 3D modelling of architecture assets
- Nathan Grove: 3D modelling of architecture assets

Aside from the above exceptions all concept design, coding, artwork, narrative-authoring, puzzle design, and in-game implementation of music/art assets was done by the author.

Part 1

Exegesis

1. Introduction

It is highly recommended that before proceeding with this exegesis, the reader should watch Appendix D: Video 1 to get an overview of the gameplay dynamics of the framework.

This thesis is a detailed creative and technical realization of a system for expressive performance in video games. This exegesis discusses a three-year development process which involved the creation of a modular framework for performance, and a specific implementation of that framework in the game *Pallas of Vines* which constitutes the core work of the thesis.

This system allows players to create original and expressive audio-visual performances entirely within a 3D game world through controlling the motion and dynamic properties of avatars and performance objects that can be made to interact with each other to execute a diverse array of audio, visual, and narrative functionalities. These performance objects each have visually and behaviourally distinct manifestations in the game world, and can be freely instantiated and moved around a virtual 3D space where both avatar-to-object interactions and object-to-object interactions can have dynamic interdependent relationships that are mapped to the generation or alteration audio/visual/ narrative materials in real-time performance.

Hamilton has undertaken similar foundational work exploring mappings of gameplay dynamics:

> *By creating music and sound in virtual environments procedurally, that is by creating and controlling sounds through the mapping of parameters of motion, action or state to sound producing systems, the sonic experience presented to users can be tightly integrated with the visual and narrative experiences on which successful cognitive immersion in game environments is based. The same user control systems that control motion or action in space can control sound and music. By coupling user action in direct as well as abstracted fashion, rich artistic environments can be created. In this way,*

> *the virtual environment itself can become both an active and reactive participant in the sonic experience.* (Hamilton, 2014)

Pallas' gameplay system does not in any way rely on the player emulating real world musical interactions, nor emulating/recreating prescribed pieces of music, nor does it have any measure of what constitutes a successful performance. Instead it focuses on providing avenues for creative personal expression to the player, where they are invited to master a flowing dialectic of interdependent relationships between game elements in which they, as player/avatar, are an equal interdependent element.

At a certain point in the research the system divided into the two distinct streams of Performance Mode and Story Mode, each with different gameplay affordances. While Story Mode, in order to facilitate game objectives and narrative progression, does have certain measures for the player's successful execution of performative units in the form of puzzles (usually musical in nature), in these cases the system ensures that players still have a great deal expressive freedom in how to performatively execute any given puzzle solution. In both Story and Performance Modes, establishing a co-creative context for the player is essential to this research. Although this co-creative aspect is explored in various later parts of the exegesis, it is important that the reader understand it as a foundational principle that supports the work. A simple definition of the term is provided here in order to frame this understanding. Throughout this research the term "co-creative" is used to mean play/performance contexts (both virtual and real-world) that the player has abundant means to engage with, interpret and, if desired, recontextualize via their own creative input. A simple example is the player seeing and exploring the game-world in Performance Mode, and then creating music in response to their perception of it. As is discussed later in the exegesis, this becomes even more significant in Story Mode which has a greater reliance on pre-authored content (audio, visual and narrative). Thus in order to retain a high degree of expressiveness for the player, multiple in-roads to this pre-authored content must be implemented, and this is where the principle of co-creative design becomes most significant. Beyond this, there is also the abundance of options for routing and hardware mapping (e.g. MIDI controllers), ensuring that players have an equal part in designing a

performance style that works best for them, within the constraints of the game. While this last aspect is common to the field of electronic music software, it is no less important to recognise its co-creative value, and see how and where this can be applied to other areas of performance in gameplay. To be clear, the idea of co-creation as it has long existed in both music performance and traditional gameplay is a guiding light for this work, and the research makes no claim to originating the notions of co-creative performance or play.

This thesis has evolved out of a long personal history of electronic music in which I specialised in live performance, improvisation, and software/interface design. In this field I was creating software in order to achieve specific kinds of improvised performance practices that were otherwise not possible. These past works involved modular interface design combined with advanced uses of gestural (e.g. accelerometer) and camera-tracking (e.g. Kinect) technologies, and included the development and performative uses of real-time MIDI and OSC mapping systems. While it is acknowledged that accelerometer and camera-tracking technologies could be applied to this current research, in those past musical practices I found them to either be too unstable, too requiring of constant moment-to-moment adjustment (and therefore too difficult to use simultaneously with other controls), and too hampered by latency and as such have been omitted from the scope of this work.

Much of the videogame work of this current thesis has been achieved with great specificity due to my foundational work in electronic music. My knowledge of visual-programming/virtual-patching environments such as *Max* (Cycling '74, 2016) helped to establish the routing paradigm upon which this current thesis is based, and performance software that I had created in the past often acted as a blueprint by which certain musical functionalities could be translated into the game world. My experience with other routing and modular performance music software such as *AudioMulch* (Bencina, 2016) and *Reactable* (Jorda, 2003, 2009), which have illustrated the potential for real-time capabilities beyond *Max's* real-time routing limitations, has helped to further expand the conceptual model of this framework.

The field of research for this thesis includes both academic and commercial innovations in both music and game technologies. Significant academic precedents to this work include the research of Hamilton (2011, 2014) and Oliver (1999, 2003, 2009), Furukawa et al (1999), the electronic music research and commercial realisations of Carlson (2011, 2014), Jorda (2009) and McCormack (2015). Significant musical precedents in the commercial gaming world include the work of Iwai (1987, 1996, 2005, 2006), Mizuguchi (2001, 2011), Moriarty (1990), Miyamoto (1998, 2002, 2006), and Flanagan & Boom (2013, 2014).

It should be noted here that the game research and practical outcomes of Hamilton, Oliver and Furukawa are considered, in the context of this research, to be rudimentary works commensurate with the relative newness of this field. Similarly, the relevant commercial games of Iwai, Mizuguchi, Moriarty, etc. can be seen as commercial experiments (the referenced works of Miyamoto being exceptions as large-scale commercial games, but their relevant music aspects only comprise a small portion of the gameplay for each game). This is all to say that the field of music performance in games (where there is a significant narrative or at least avatar-driven component) is still in its infancy. As such, this research should be understood by the reader to be bringing the strengths of electronic music interface design, and its related performative affordances, into the field of gameplay-as-performance (including translating these structures into aspects of visual and narrative performance in gameplay). As practice-led research there is only a limited scope to describe here in writing the process by which my years of electronic music performance and related software design has translated into gameplay-as-performance interfaces. Beyond this exegesis, the bibliography should give the reader sufficient perspective as to the breadth of this process which cannot be neatly summarised as merely interaction design, game design, theories of gameplay, interactive narrative or games-as-performance, but should be seen as an aggregate of those things, for which Bogost's unit operations (described further on in this section) as well as my own elucidation of branching and nesting principles, provide unifying structure.

The notion of expressiveness is explored throughout this research from two distinct points of view, as it is, and has been throughout the work, the primary

criteria for assessing the success, or the need to reiterate and refine, any given element of the performance system. The first perspective is the technical: for any given parameter, how much moment to moment control does the player have, and how much scope do they have to create and alter relationships between that parameter and any number of others during a performance. The second perspective is the human: this is the potential for the player to engage in "personal creative expression" during performance. Clearly this is much more dependent on the individual and their propensity for engaging in personal creative expression in life/art in general, so this aspect is discussed in terms of dialectics: looking at what the elements of any given performance are, what the dynamics of their interdependent relationships are, and how the player is an element that is situated both inside and outside of that performance, and thus their consciousness being that which freely crosses the boundary between virtual and real world, and the many possible levels of creative synthesis they can achieve between the elements of performance, while such navigation is taking place.

Thus, regardless of the player's personal experience or confidence in being expressive, this performance system is designed as an environment in which creative dialogues can be initiated and are provided with the potential to flourish: an environment where a great diversity of avenues exist (or can be created by the player) through which performance unfolds as, and through, an awareness of these charged creative potentials.

To support this second perspective the following research frameworks have been employed: Bogost's (2006) concept of "unit operations", exploring the dynamic relationships of discrete units in games, Laurel's "computers as theatre" (2013) analogy, looking at the implicit narratives in the performance of interactive systems, and Murray's elucidation of a procedural narrative via systems of interrelated entities (1997). These texts will aid in providing a grounded perspective on performative dialectics while avoiding a drawn-out philosophical discussion that is beyond the scope of this exegesis.

Before entering into specifics about the performance system, an overview of its functionality is necessary. The system is large and detailed and as such it is not

possible to describe every individual functionality in this exegesis, however all this information is available in the diagrams of Appendix A, the Performance Manual of Appendix B, and gameplay videos of Appendix D. It is recommended that the reader, at minimum, explore Appendix A: Diagrams 1, 2, 5, 15 and watch Appendix D: Gameplay Videos 1, 2, 3, 7, 8, 9 before proceeding with this exegesis. Alternatively, for the most in-depth understanding of all aspects of the framework, it is recommended the reader cross-reference all Diagrams in Appendix A with the Performance Manual in Appendix B, all videos in Appendix D, and play the game itself (Appendix E).

As mentioned this system is divided into Performance and Story modes which each represent very different forms of gameplay. Except where otherwise stated, discussions in this exegesis will focus on Performance Mode, as the majority of Story Mode features have filtered down from Performance Mode.

Performance mode takes place in a freely navigable 3D environment in which players can create and destroy avatars and many different kinds of performance objects, such as audio sources, audio effects, lighting objects, camera filters and so on (see Appendix A, Diagram 2 for a full list). Most objects have parameters attached to them (either faders, dials, buttons or a combination of these things), which are visible and tangible in the 3D world, and by which the player can manipulate the internal data of those objects (for example a Pulse object has just one fader which allows the player to adjust the tempo of the pulse). Players can also route objects together where a pre-defined compatibility between object-types exists; for example a Pitch object can be routed into a Sample Player and used to change the pitch of that sound, but a Pitch object cannot be routed into a Pulse object as they have no conceivable relationship. Routing will be discussed in detail, but on the surface just consists of players using the cursor to draw a line from one game object to another (see Appendix D: Video 1).

A player can also instantiate new avatars and choose which one to control at any given moment. Avatars are needed to listen to sound sources (i.e. to render them audible in the real world via their proximity), to pick up and move objects, and to engage in proximity relationships with many other object types.

Proximity between objects is a key element in this system and it is used to communicate between many different object-types. Proximity-dependent objects are surrounded by transparent fields that show the radius of their proximity responsiveness (see Appendix D: Video 1). Since many objects can be moving in the game world at once, proximity becomes a powerful way of creating a very active musical “ecosystem” over which the player can engage in a combination of:

1) Exercising a fine grain of control over the moment to moment dynamics, or
2) Setting things in motion and observing what unfolds

Thus a combination of both “direct drive” performance (where the player has precise control of individual parameters), and generative music making is possible (see Appendix D: Videos 12.1 and 12.2).

From the start of this research *Pallas* was designed so that every part of a performance would exist inside the game world. This means that all elements of the performance interface inhabit the 3D space, no part of the graphical user interface (such as a head up display) is overlaid on top of the 3D world, and the player always has the option to perform with no explanatory text or numbers nor anything else that would interfere with the fiction of the game world.

Regarding multiplayer, while it is possible in Performance Mode to have multiple human players controlling multiple avatars simultaneously, there are several different contexts with various contingencies for doing so (as discussed in Section 4.1). Except where stated otherwise, this exegesis discusses examples in the context of single player, as all single player capabilities are also present and functionally identical in multiplayer. While online multiplayer is compatible with this framework it was beyond the scope of this research to technically implement. Appendix A: Diagrams 4 and 5 detail the relevant conceptual details of such an implementation.

The core practices employed in the realization of the practical component of this thesis – that is, in creating the framework and the game *Pallas of Vines* – involved and combination of interface design, coding, rehearsal (practice performances with the game itself), and subsequent improvements, augmentations and refinements to the system. The work was created within the *Unity* (2016) game development environment and coded in the C# programming language. However it should be noted that the focus of the research is the design and gameplay implementations of the various components of the framework, and not the code itself which I consider to be just one out of many possible ways of expressing these concepts. The ergonomic design of the performance objects was created and iterated upon inside the *Unity* editor using its in-built 3D primitives, and where needed other 3D meshes were later imported to make each object-type visually distinct. Several additional code-libraries and middleware (see Appendix C) were used to augment *Unity's* default functionalities, but only in cases where programming them from scratch would have been an immense undertaking and/or a reinventing of the wheel, and not relevant to this research.

Pallas of Vines has been presented at performances, and as a playable work at several international conferences and was enhanced by expert guidance during a 6 month period of Visiting Scholar research at the University of Southern California's Game Innovation Lab (see Appendix F).

In order to orient the reader, a brief structure of the remainder of this exegesis follows. Firstly "expressive potentials in gameplay" as it relates to this research will be defined. This definition will apply throughout the rest of the exegesis as the measure by which the framework has been developed and iterated upon. Branching and nesting will then be defined as both technical and abstract structures that contribute to and expand this definition of "expressive potentials in gameplay".

Branching and nesting will then be used as a lens through which to explore inter-object interactions within the game world, and then how those can relate to the real world, such as in hardware controls, control mapping, multiplayer and facilitating the player's fluid navigation between real and virtual world.

Next all these expressive affordances will be discussed in terms of their integration into narrative and game-objectives. This area of the discussion encompasses affordances for both players and developers, where developers can leverage their own process of performative gameplay rehearsal to design objectives that can be similarly performative for the player.

The discussion concludes with an exploration of the interrelation between players and developers as a co-creative community, and this is the culminating framing of the expressive interdependent interaction between real-world and virtual-world.

The original contributions of this research will be summarised here within the field of videogame research only. Although some specifics of these contributions cross-over and feedback into the field of electronic music, most notably around dynamic interfaces and virtual-physics-based continuous control, it is beyond the scope of this exegesis to detail their validity in that field, as it would necessitate abstracting these elements from this work's fictional context – a lengthy process creating unnecessary complexity in the face of otherwise discussing game performance as a holistic audio-visual-narrative experience. Thus beyond the aims state above, the contributions of this work to the field of videogame research are as follows:

- a definition of expressiveness in gameplay as it relates to the realisation of player intention, and a subsequent measure against which expressiveness in gameplay can be planned for (before the fact) or analysed (after the fact)

- a practice-led approach involving the design, testing and refinement of a game encompassing an expressive, co-creative performance framework (a collection of interdependent game objects and behaviours), unifying audio and visual performance with gameplay dynamics and narrative context

- a further breakdown of the above into Story and Performance modes, illustrating key contingencies needed to allow each to remain expressive and narratively contextualized. These modes serve as two extremes of a spectrum

(player-creativity/prescribed objectives) on which this research can be practically applied to future works

- a practical application of Bogost's "unit operations" and Murray's "virtual world full of interrelated entities" demonstrating the value of those theories as design principles that can enhance the expressiveness of moment to moment gameplay when both game designers and players create, observe and participate in, dynamic dialectic gameplay systems

2. Towards Maximising Expressive Gameplay Potentials for Players and Designers

2.1. Creating a Definition of Expressive Potentials in Gameplay

Throughout this research, in order to continually reflect and reiterate upon the objective of achieving maximum expressive potentials for players and designers, it was first necessary to define what constitutes "expressive" in the context of this research. Clearly "expressive" is a highly subjective term, particularly when it comes to improvisation which was the most common means of rehearsing with and refining this work.

While this section unpacks the way in which expressiveness has been *applied*, an even simpler definition will first be provided in order to frame that application. A common definition of expressiveness is "effectively conveying thought or feeling" (Oxford Press, 2018), and in this research the word "expressiveness" can indeed be used interchangeably with the phrase "to accurately realise (i.e. manifest) intention" and I will add to this an essential emphasis on dynamics, i.e. accurately realising intention from moment-to-moment. The research lays no claim to gauging a player's ability to be creative, rather its only concern is to offer them the means to accurately realise (i.e. give audio-visual-narrative form to) their intentions in the context of moment-to-moment performance. A further delineation is necessary where the reader might mistakenly infer that this work is seeking to inspire "expressiveness of thought or feeling" in the player. While "expressiveness of thought or feeling" is a valuable creative principle and a

worthy goal, it is too broad a philosophical discussion, and too intangible to succinctly quantify, to encompass in this research. It is only discussed to the extent that the game's fictional context is intended to inform the player's co-creative feedback loop during play (see Section 4.1), and, as previously mentioned in this section, where a dialectic interplay of elements can serve to further support the player to situate themselves as an interdependent element within the combined real-world and fictional performance contexts, particularly in the game's Performance Mode.

Where Bogost (2006) discusses play in terms of authorial intent (of the game designer) and interpretive freedom (of the player), the added factor of expressiveness, in terms of explicitly designing gameplay to enable the player's original creativity, thus blends these notions of authorial intent and interpretive freedom for the player. Thus in this research the player becomes co-author of their own performance, as well as both interpreting the game context (as created by the game designer), and their own dynamically unfolding performance context. This is the framing of "expressiveness" throughout this work for which Bogost's unit operations provides the theoretical underpinning: intention being realised, in both the tangible and the abstract, across dynamically shifting contexts of moment to moment gameplay in both the real and virtual worlds and their dynamic interplays. This is explored in greater depth in Section 2.2.

Defining what is meant by "performance" in this context is thus also important. Again the focus here is on the intention of the player/performer. Thus the research is not concerned with defining a specific or narrow range of performance contexts – i.e. in this sense recreation or rehearsal is as valid a performance as playing professionally to a live audience – and again is also not concerned with gauging the validity of a performance by its creative merit. To put it another way, "performance" here is framed as "the moment-to-moment process of the player/performer giving (audio-visual-narrative) form to their intention". Thus wherever this exegesis implicitly or explicitly compares performance in gameplay to musical performance, that framing applies equally to both cases. Furthermore, wherever this exegesis discusses gameplay in

reference to *Pallas of Vines*, the word “gameplay” can be considered synonymous with the word “performance”.

To clearly frame the aspect of this exegesis focused on “maximising expressive potentials”, the reader may well look at the practical results of the research and find it wanting in comparison to performance on an acoustic instrument — and perhaps more pertinently — performances in electronic music software. In this sense it is essential to keep the following framings in mind:

- This musical aspect of this research and its practical outcomes are concerned with *electronic* music, in which my creative and technical background is deeply rooted. While there are certainly branches of electronic music that are concerned with emulating acoustic instrument sounds and techniques, they have no bearing on this research. This work is only musically concerned what is unique to electronic music – those sounds and performance styles that are not possible to achieve by acoustic means. However, the *expressiveness* by which acoustic instruments can be played – that is their intimate and immediate moment-to-moment audio-visual-tactile responsiveness and feedback for the player – is held as the highest measure of expressiveness that the vast majority of electronic-music instruments/interfaces have not yet achieved in a unified sense (i.e. in some cases unified along one or two of the audio-visual-tactile axes, but rarely unified on all three axes). To put all of this simply, an acoustic instrument can’t make the sounds that electronic instruments/software can, but generally speaking electronic instruments/software have not yet achieved the level of unified audio-visual-tactile responsiveness and immediacy of player-feedback of acoustic instruments.

- This research and its practical outcomes should be viewed holistically as a game-as-audio-visual-instrument. Any attempt by the reader to separate the game’s electronic music potentialities from its visuals and its player-interface will result in a misunderstanding of its significance and contribution to the field of gameplay. What is significant about this work is that entire performances take place within a game world, informed by the visual style and atmosphere of the game, and its player-controls, responsiveness and

physics. Performance entirely within a game world thus unifies the dual concept of play in the sense of “playing an instrument” and “playing a game”, and equally significant is that the visual interface doubles as a visual performance. It should also be noted that while the game’s Story Mode makes this holistic performative-play very clear (e.g. music is performed towards a narrative end of solving puzzles to advance the plot), Performance Mode hinges on *player intention* to create this unified audio-visual-narrative-gameplay performance. Without the holistic intention, the question of “why not just perform with traditional electronic music software” is valid, but with this holistic intention, that question becomes irrelevant. However, as with the point above, the existing expressive potentialities in the field of electronic music software have been, and continue to be, valuable points of reference in developing and refining this work.

Thus this research makes no claims nor intentions to being more expressive than existing forms of acoustic or electronic music performance. Rather it situates itself as foundational work in unifying audio-visual-narrative performance with gameplay, thus contributing to future research that will undoubtedly reach those benchmarks of expressiveness already set by acoustic and electronic music performance.

In order to achieve the formulation of a clear, concise framework by which to achieve expressiveness – and most importantly one that would support the reflective reiteration necessary to refine the performative affordances of this game – an *axis of expressive potential in gameplay* was devised. While this was essentially just the very obvious and simple way in which I had, for many years prior, been refining my development of expressive musical technologies, it became clear to me that like many very simple internalised personal aesthetics, it takes some explaining to convey its simplicity. Note that for the sake of brevity the following examples are musical, but the reader should keep in mind that the axis of expressiveness being formulated here encompasses audio, visual and narrative gameplay as a unified whole.

At its most basic, I define the axis of least to most expressive with the following analogy:

- A sound plays in a continuous loop, the performer can only switch it on or off and has no other form of control. This is minimally expressive. No matter how practiced a performer or how elaborate their gestural input, they do not have expressive control of the resulting output.
- A volume slider is infinitely more expressive (assuming it is analog, otherwise if digital, it is more expressive to the degree of its data resolution). Now the performer has precise and fine-grain moment-to-moment (continuous) control. The performer now has access to expressive dynamics.
- A hypothetically infinite number of volume controls on an infinite number of sound sources is infinitely more expressive again. Now the performer has access to interdependent expressive dynamics, where from moment-to-moment they can expressively control the relationship between any number of dynamic parameters.
- However, given that in this example the sound is the same in each instance, any means to continuously control pitch or timbral characteristics creates another layer of expressiveness. Thus it is not just the amount of parameters that will contribute to expressiveness, but the availability of a diverse array of parameters.
- Then with such a diverse array, any given performance would require a means of balancing that diversity: at least one aesthetic feedback loop. This means interdependence between those diverse elements. This could be as simple as the performer continually monitoring and altering parameters to maintain the balance, or it could include one or more automated means of reflective signal processing, whereby the digital system can assess and adjust its own output (a simple example is an audio compressor).

The minimal end of our spectrum is easy to understand, however the maximal end is only hypothetical and illustrates another clear consideration necessary to include in this measure of expressiveness: physical and cognitive constraints. Simply put, with our hands, or potentially hands and feet (head, eyes, and so on), we can only control so many parameters at once, and our conscious awareness, in the dynamic moment to moment demands of performance, can only focus on so many at once.

Clearly, this does not mean that we can't control 20 different parameters over the course of a performance for example, it simply means that they must be spread out over a reasonable amount of time so that we are not tasked with manipulating all 20 parameters within the span of say 2 seconds. Thus manipulating a very large amount of different parameters is also possible, given a significant time-span over which to do it.

At this point it is important to note that in-depth discussions of the fields of performative embodiment and human-computer interaction are beyond the scope of this exegesis. The practical results of this research include in-depth player-controlled routing and hardware/software mapping systems designed to enable the player to co-author their own ideal performance interfaces to support their own physical and cognitive needs for any given performance. This co-authoring takes place both inside the game world (the dynamic creation, placement and routing of game objects) and between the real world and game world (e.g. mapping MIDI hardware to in-game performance parameters). So while factors such as dynamic/continuous controls, gestural controls, and tactile responsiveness/feedback are valuable (often essential) components of expressive performance in electronic music, the reader should keep in mind that this research does not involve the creation of hardware interfaces, nor does it prescribe the way hardware interfaces should be used in performance (beyond a necessary default QWERTY and mouse mapping of the game controls) and thus it limits its design considerations to the dynamic availability of routing (see Section 3.1) and mapping options, and continuous audio-visual feedback for mapped parameters (see Section 4.3). Effective applications of embodiment and human-computer interaction principles are thus framed here as the responsibility of the player, and can be seen as additional "units" (to use Bogost's term) in the interdependent framework, which the player can choose to dynamically navigate and incorporate into performances as needed.

Including physical and cognitive constraints, my axis of expressive potentials in gameplay would then look like this:

Table 1: Basic Axis of Expressive Potentials in Gameplay

Minimal Expressiveness	**—>**	**Maximal Expressiveness**
A single Boolean switch (on or off)		Any number of continuous control parameters • with a diversity of parameter types • with a means of interdependent communication between diverse parameter types • limited to what is possible to physically and cognitively maintain control and awareness of in a given time-span

However, “limited to as many as physically possible to control” would be clear if talking about a physical space – for example the controls would ideally need to be within arm’s reach – but it brings up another question when considering the virtual performance space, where controls can be distributed around a potentially infinitely-sized virtual 3D environment. In this case, neither space nor amount of parameters is necessarily a problem, as long as we have a suitable means of quickly navigating to the controls we need at any given moment. This “navigation” includes not just moving through the 3D space to find what we need, but also hiding and revealing visual information in a given area, as necessary for minimising visual clutter and other cognitive “noise”.

So including these constraints, my axis of expressive potentials in gameplay is found in Table 2:

Table 2: Advanced Axis of Expressive Potentials in Gameplay

Minimal Expressiveness	**—>**	**Maximal Expressiveness**
A single Boolean switch (on or off)		Any number of continuous control parameters • with a diversity of parameter types • with a means of interdependent communication between diverse parameter types • limited to what is possible to physically and cognitively maintain control and awareness of in a given time-span • with the means at any given moment to hide, reveal, and navigate-to performative parameters

The above now defines maximum expressiveness of dynamic parameters in a performance space. However, it does not yet define the relationships between those parameters. In order to achieve this, two principles were applied: branching and nesting.

The concept of branching was applied to simply explore which objects and parameters could be connected to which (by players in real-time), and if a given object could output its data to multiple other objects at once. Throughout this research, any time a new game object was developed within the framework all of its relationship potentials with other objects were maximised by applying this principle.

The concept of nesting was applied in two ways. Firstly as a means of situating one game object inside of another, allowing for a multiple different object types to be grouped together and moved via a single parent-object, thus maintaining their positions and rotations relative to others in that nested group (and thus any proximity relationships they may be currently engaged in). This kind of nesting also allows the player to dynamically hide/reveal performance parameters if and when needed in order to reduce visual/cognitive clutter. For example, a Sample

Player object has a built in amplitude envelope, however if the sample is playing in at its full length the envelope is not needed, so the player can toggle the visibility of the envelope's parameters at any time. Thus the envelope acts as a sub-object nested within the Sample Player. See Appendix D: Video 2, at 2:42 the player toggles the envelope on, thus revealing the relevant nested parameters.

The second application of the nesting principle is as follows: where a given object A passes a parameter to object B which passes that parameter to object C and so on. If any change to the parameter is made at any point, this is applied to all subsequent receivers down the chain. In this way the player can create nested chains of data transformations.

Such chains could also branch at any point, and those branches could themselves branch again, and potentially pass along other nested transformations, and so on. Thus additional expressive potential is given to the player by enabling them to create such structures.

So we thus have the following measures to assess expressiveness:

- Dynamics of manipulating multiple parameters
- Creation and alteration of branching and nesting relationships

The final measure of expressiveness breaks from the technical and adopts a philosophical perspective, or rather multiple perspectives: this is the dialectical nature of expressiveness.

Here the player is another "unit" (to use Bogost's terminology) in dynamic relationships with all other units – this includes the performance objects in the game world, the avatar that the player is controlling, the game itself as a whole, the current performance/play-session the player is engaged in, the computer that the game is being played on, the real-world the player inhabits, the universe the world exists in, and so on.

All of these are examples of radically different points-of-view to which the player can relate their conscious engagement with the game. Some of these are very

clear and literal, for example a sound source in the game world can only be heard through the virtual “ears” of an avatar. As the player can have direct control of an avatar’s movement through the game world, it is easy for them to hear from the avatar’s point of view. Taken a step further, we can have multiple avatars in the game world, and we can assign them sounds that only they will hear. When this is done, the player hears the aggregate of what all the avatars are hearing at any given moment. The player is thus, in a clear and simple sense, actively engaged in synthesising multiple points of view.

In this sense it is not just the player’s input that is expressive, but also their reception, and for the player to understand what they are receiving they must understand how it relates to their own input. Thus the performative feedback loop between player-input and game-output is the essence of this synthesis, and thus moment-to-moment audio and visual responsiveness – both of the game-to-the-player and the player-to-the-game – is paramount for enabling this process of synthesis itself to be maximally expressive (Schacher, 2015, Jorda, 2003).

In the above multiple-avatar example the avatars mediate in translating the output of the sound sources into data streams that are more coherent or manageable for the player in the physical world. In a similar way, most of the game objects in this system could be said to perform some sort of translation process in order to make data streams more easily navigable, or more easily render them aesthetically pleasing.

However, dialectics comes more significantly into play when a player begins to master certain aspects of performance. When this happens they are not just waiting for the real-world output in order to reflect and correct the course of their performance, they are seeing things from the point of view of many game-objects at once. They can create a structure of many different game objects in relation to each other, and know ahead of time the field of possibilities they can thus navigate, given this collection of objects. To put it another way, understanding the point of view of those objects, the player can anticipate the nature of the “conversation” that will take place, without necessarily knowing the specific course it will take.

The experienced player can thus synthesise these multiple differing points of view and see the situation from a higher perspective by understanding how these different points of view will behave in relation to each other. By adding their own performative input to such a situation, the player can also become aware of their own interdependent role in this unfolding conversation, thus functioning as an equal part of this virtual “ecosystem” rather than attempting to impose top-down dictatorial control.

Achieving dialectic expressiveness depends on the level of conscious engagement that the player is able to attain; however what is important here is that this system was designed to both support and nurture this kind of engagement.

While personal creativity is a factor of expressiveness, when it is viewed through the lens of such dialectic interdependence the player can view the dynamics of their relationships to the other elements of a performance system at any given moment, and assess and respond to the kind of communication that is taking place: responding to the unfolding demands of the work to become a work once under way (Peters, 2009) where the artist is a part of that work. So once again, original creativity comes from the player’s consciousness of the unfolding interactions of the performance system, including a consciousness of their own internal aesthetic goals (which is itself a discrete and dynamic element (a “unit”) in relationship to all other elements in any given moment of the performance).

As a point of reference I refer to the game *Electroplankton* (Iwai, 2005). This game contrasts with the “rhythm game” paradigm (Pichlmair, 2007), which is the most popular form of commercial music game in which players are tasked with pressing buttons or singing in time with a backing track, and are then scored on their accuracy. Rather than this prescriptive model of music-making, *Electroplankton* provides a free-form model that favours player-centric creativity and expressiveness. The player can choose from a series of different game environments, each with a different mode of interacting with a touchscreen,

which enables them to use gestures in various ways to create music. There are no explicit objectives, and the game offers no internal measure of success or failure. Thus the player is encouraged to engage in exploratory play, implicitly favouring their personal creativity. On the other hand all the different forms of musical interaction in the game are constrained to sound "pleasing" i.e. scales that contain no strong dissonances, game environments whose note-triggering is entirely tempo-quantized, and in most cases a significant portion of the musical playback is automated (e.g. the player can set the parameters of a musical pattern in motion, and the game will infinitely loop that pattern). Obviously all such factors place limits on player-expressiveness in favour of accessibility to a broader audience.

While *Pallas'* performance system makes no claims to being accessible to a wide audience, this is an implied factor of expressiveness in terms of good interface design. Simply put, the more complex a digital interface is, the more it gets in the way of the fluid execution of moment to moment performance dynamics.

A secondary factor for which *Electroplankton* provides a succinct illustration is the fictional game world context: rather than visually emulating a DAW or synthesiser interface, the game is based in a fictional context (a miniature underwater world). Here all its music-making capabilities involve the player interacting with game characters (these interactions range, in different areas of the game, between direct character control and indirect "influencing" of the characters through interactions with the game environment). *Electroplankton* could instead have been an abstract music application where game characters and environments were replaced with abstract shapes, numbers for parameter values, musical note-names, musical grids, and so on. As a contrasting example, the KorgDS10 (Cavia, 2008) (also a NintendoDS "game") is an emulation of the real world hardware interfaces of the KorgMS range of synthesisers: thus an abstract music-making tool with no fictional game-world context or characters.

Given Iwai's diverse experience as a multimedia artist in creating abstract interfaces, the intentionality of his use of a fictional/character-based interface is

clear (Mosely, 2016, 2). By using game characters and environments there is immediately a narrative context (no matter how rudimentary), and thus players can have at least the most basic level of expressive engagement with that fictional environment, which becomes an aesthetic consideration in any performance

In this sense *Electroplankton* can be seen as both an audio-visual composition with a large possibility space for its realization, and an instrument for either realizing itself as a composition, or for realizing some other score. This notion of the composition as instrument is further explored in Section 3.2. In this co-creative sense all games, even those of non-musical gameplay, are both compositions and instruments.

> *Games disturb the relation between reader and story that narratives require. In a game, "the player inhabits a twilight zone where he/she is both an empirical subject outside the game and undertakes a role inside the game." Whereas narrative creates "cognitive identification" with generally human or anthropomorphic characters, games implicate the player personally in the work.* (Bogost, 2006)

This expressive engagement with that fiction is also an important factor of this research. Music-making in a virtual-world cannot be done in a void: there is no way to create a nondescript 2D interface (Magnusson, 2005), much less a 3D environment, populated with avatars and objects, that is generic or without fictional context. There will always be some aesthetic sensibility to it and this will always influence a player's performance, whether consciously or not.

Thus my aesthetic decision in creating this performance system was to create an environment that implicitly supports and nurtures creative dialectic engagement. Thus the reason for the peaceful and contemplative natural environment of *Pallas of Vines*, evocative of a harmonious and balanced ecosystem. I acknowledge that that which constitutes an environment conducive to "creative dialectic engagement" is highly subjective, so of course I could only gauge this by my own aesthetic sensibilities. As with *Electroplankton*, what is most important is that a context is established with which players can

have expressive engagement, and to which they can choose to align or oppose their performances and aesthetic intentions.

This notion of the relationship of aesthetic intention to performative gameplay of narrative will be further explored in Section 5.2.

The digital screenshot archive of Appendix G illustrates the aesthetic evolution of *Pallas'* visual environmental.

2.2. Branching and Nesting to Enable Greater Expression While Minimising Complexity

In the creation of this performance system the application of branching and nesting principles creates the following affordances for players:

• The application of branching to performance objects wherever possible and practical gives players maximum options for moment to moment authoring of complex performance structures; that is, the real-time routing of signal flows between objects. This authoring includes creation, augmentation and destruction. Where possible, this includes the potential for one-to-many connectivity.

• The application of nesting to performance objects wherever possible and practical affords players the maximum potential for moment to moment control of the hiding/revealing of parameters. This is required to maximise the availability of parameters-per-object while minimising the complexity of on-screen information. An object that is by default nested inside of another object is essentially a component for the parent-object that can be hidden or revealed during a performance if and when needed.

• A second kind of object-within-object nesting is also possible. This pertains to a performance object that is specifically designed for nesting. Any other objects of any kind can be assigned to this object, thus inheriting the relative dynamic spatial orientation of this object. That is, if this object is moved,

these attached objects will move with it while maintaining their distance and rotation relative to it. This functionality can be used where the player wishes to keep the relative positions/rotations of a group of objects static, but whose positions/rotations relative to other objects outside of that group can remain dynamic.

• The application of nesting in the second sense is the nesting of data flows where any given “receiver” of data inherits any transformation applied to that data from any “transmitter” that precedes it in a signal chain. This concept can then be augmented with branching structures whereby complex structures of data transformation can be created in real-time. Once manually configured by the player, these nested data-transformations (whether branched or not) will thereafter be automatically applied to any incoming data in that chain, unless/until the configuration is altered by the player. This affords players the maximum potential for applying any number of transformations to any given data input. Thus even moving a single parameter can result in a cascade of responding parameters, whose responses could be anything from subtle adjustments of the incoming data, to dramatic transformations into an entirely different form of data.

These applications of the branching and nesting principles just described all pertain to mechanical functionalities: how data exchange is routed, how information is hidden and revealed, how objects are held in spatial-relationships to each other, and how data exchange is transmitted and transformed. However these are all manifestations of a more essential philosophical foundation that returns to the question of dialectic interdependence.

2.3. Branching and Nesting as a Philosophical Foundation

Branching and nesting at the philosophical level has been shown to be a means of achieving clarity and depth in play, as well as in both implicit and explicit narrative (see more on this in Section 3.2), through the creation of a language

of dynamic interdependence over which players have high levels of both pre-authored and real-time control of relationships between game-objects.

This results in a co-creative dialectic interaction between players and the game-world that allows object relationships to come to the fore as sites of expressive potential – where any level of pre-authored content can always be designed to have any level of co-created content built into its interface.

As has been discussed, this allows players to re-configure branching data-flows, as well as being able to navigate within information that is nested inside other information, and steer the flow of transformations across nested hierarchies. This ultimately reflects the player-avatar relationship beyond the virtual world, where the player can thus reflect on their physical-selves as an avatar of a higher-order awareness. Simply put, by synthesising the relationship between their real-world selves as player, and their in-game avatars, the player can witness their role in the simultaneous interdependence of both real and virtual worlds.

> *Game designers Katie Salen and Eric Zimmerman suggest that digital gameplay involves a "double consciousness" through which a player may identify with an avatar and yet remain "fully aware of the character as an artificial construct". Salen and Zimmerman propose this as a way of refuting what they call the "immersive fallacy," the misconception that, in an ideal gameplay scenario, "the player would identify completely with the character, the game's frame would drop away, and the player would lose him or herself totally within the game character"* (Cheng, 2014)

As Bogost suggests in his elucidation of his concept of “unit operations”, the “unit” is not just the discrete unfolding algorithms of the game, or even the inclusion of the player’s agency to influence that unfolding, but it includes the many and dynamically shifting contexts of moment to moment gameplay in both the real and virtual worlds: the tangible and the abstract, and all their dynamic interplays (Bogost, 2006).

> *Understanding units as objects is useful because it underscores their status as discrete, material things in the world…while I include in my understanding of units ordinary objects…I also claim that units encompass the material manifestations of complex, abstract, or conceptual structures.* (Bogost, 2006)

> *Videogames ask the critic to ponder the unit operations of procedural systems. It is only appropriate that we also begin thinking of such criticism as…a set of relations between parts, not just in text, but in the world as well.* (Bogost, 2006)

Through this "double consciousness" players can thus recognise the nested contexts in which performance takes place: the game world nested inside the real world, the discrete motifs and gestures nested inside any given performance, and so on. This allows us to view the vast magnitude of perspective available to us when reflecting upon the player-avatar relationship in performance: generally speaking performance of any kind is already self-reflective for the skilful performer, so when we add this powerful metaphor, the opportunity for freely navigating back and forth through higher-order states of self-reflectivity becomes much more direct and palpable.

Simply put, we have nested feedback loops:

- the player has a thought and so executes a real-world input on the game-hardware in order to move an avatar in the virtual world
- the avatar responds to the gestural input, thus effective the game's audio-visual output
- the player sees/hears the avatar's response and adjusts their input according to whether that response has satisfied their intention

And from a higher perspective:

- at any given moment the player has a conscious intention (e.g. an idea or aesthetic intention)

- they physically execute on this communication through gesture on the game-hardware, thus moving the avatar
- the avatar responds to the gestural input, thus affecting the game's audio-visual output
- the player's physical perception sees/hears the avatar's response (and its result on the game's audio-visual output) and adjusts their input according to whether it has satisfied their intention
- the player, through the lens of their original intention, witnesses both the mind/body's response and the avatar's response and adjusts their communication according to whether those responses have satisfied that intention

However in itself this is not a linear process, as this is only one example of one kind of relationship – the player-avatar relationship. In addition, different levels of these processes could be happening across different time-frames: e.g. for a given player, perhaps their aesthetic intention could change faster than their physical ability to execute on it. Thus feedback loops at different levels may be looping out of phase. The above example is used as it is human, and thus easier to relate to. However, as this research is positing a context in which all elements of a performance are interdependent, there are many such different kinds of relationship.

At this point in the discussion the standard rational assumption would be that we cannot apply the same analogy to all other kinds of unit relationships since, for example, a sound source object in the game world does not have a consciousness. However, for the purpose of this research framework we can view an object in the game world as part of a performance context, and this performance is itself infused with conscious intention that is informing the possibility-space of the performance. Or to use a different lens, perhaps the context in a given moment is a particular movement of a performance, or the execution of a particular technique, gesture or motif. Whatever the case, it returns to the notion that the player is not in a top-down dictatorial role, they are navigating in and communicating with a space inhabited by other performance entities, both abstract and tangible (in both the virtual and real worlds). Thus when we look from within the performance itself, gauging who the performance,

or any abstract part of the performance (a movement, a motif, a riff, a gesture, and so on) “belongs” to cannot be answered simply by saying that it belongs solely to the player. Any argument assigning ownership solely to the player reduces or dismisses the dialectic context of the feedback loop of player/performance intention: that synthesis for which this performance system’s shared data-flows were designed.

Branching and nesting throughout the whole performance system of this research is a means by which conscious intention of the player can be infused into the creation of a possibility space for performance and shared between the constituent elements (the game objects) whose dynamic relationships create that possibility space.

Looking at branching in this sense, the diversity of dynamic relationships is of great significance, and as this necessitates the continuous translation of data between different types of game objects (as well as between abstract entities formed and dissolved in the shifting moment to moment demands of the performance), we thus have an additional powerful perspective on dialectic synthesis: the different “languages” that the various types of game objects speak all able to be dynamically “translated” whereby the translation process can be consciously and continuously mediated in order to align it with the higher aesthetic goals or intentions of the unfolding performance.

In order to understand what this dynamic process of translation is and how it is facilitated we must return to the somewhat more mechanical aspects of this performance framework.

3. Branching and Nesting Structures in the Game World

3.1. Real-time Creation, Destruction and Communication Between Performance Objects

At this point it is useful to reiterate that this work is focused on “maximising expressive potential" for players, and to define what that means in terms of

player affordances. Affordances here should be viewed in the unified sense of audio-visual-narrative performances in gameplay. In this way, this section details the interactions and relationships that bring such performances into being via player intention. Creating these objects and dynamically defining their relationships is one level of affordance – such functions are the basic units of gameplay – on another level the goal is then for the player to execute a successful performance (i.e. one that satisfies their intentions) using those objects and their relationships. For the latter goal, the affordances are extremely broad and require that the reader synthesise several areas of this framework: This Section, as well as Section 3.3 provide the grounding in general objects and their dynamic relationships and data flows, and Appendices A, B, and D detail, and demonstrate examples of, all performative gameplay functions. Simply put, what constitutes the affordances of this system cannot be separated from player intention, where the interrelation between available functionalities (the player's ability to create new units of functionality by synthesising different kinds of pre-existing ones) is as important as the pre-existing functionalities themselves. That is to say, this work is presenting functionalities in a branched and nested sense, where affordances are used by the player to create new affordances for themselves.

When the player begins a performance in this game they are greeted with a clean slate of functionality. Although an implied narrative context is communicated – through the nature of the game environment, the architecture, the appearance of the avatar and its animations, the responsiveness of the game controls, and so on – nothing is yet happening in the game world that constitutes a performance, and no "instrument" as yet exists (at least in the musical sense).

The player begins adding objects to the virtual performance space as needed, in order to satisfy (or at least begin to explore) whatever aesthetic intentions they have set for themselves. This intention could be anything from free improvisation to faithfully following a score. The decisions of setting up the virtual space involve not just what objects to instantiate but also where in the 3D environment to create them, since proximity (and in some cases, also relative rotation) is essential to the majority of interactions. However position and rotation are not locked in once the object is created, and performances typically

involve a great deal of repositioning objects relative to each other to effect these proximity relationships.

It is important to note also that this instantiation of objects does not necessarily only happen before a performance begins, but can continue throughout. In essence the player can be creating the instrument as they are performing with it (see Appendix D: Video 12.1). Further to this, in doing so they are reconfiguring the virtual performance space: thus both creating the instrument and the performance space, in response to the unfolding demands of the performance, “the demands of the work to be a work, to become itself once under way” (Peters, 2009).

Once created, objects can be freely destroyed by the player at any time. This is something that is very familiar to the field of electronic music in software such as *Live* (Ableton, 2016), where during a performance the creation, manipulation and destruction of modular elements such as sound sources and signal processing effects is commonplace. In this research framework, what is significantly different to that electronic-music-software paradigm is that when we instantiate game objects, at any given moment they inhabit a position in the 3D world, thus affecting the way the player will perceive and navigate that space. Thus destroying objects doesn’t just remove their functionality or expressive potential from a given performance, it also changes the dynamics of the navigable space: both literally removing an obstacle, and perceptually reconfiguring the spatial flow for the player.

Communication between game objects is one of the most detailed aspects of this research framework, and it is the aspect which has taken the most amount work to successfully execute in all its diverse game world manifestations. This is due to the fact that the potential for different kinds of connectivity between different object types is vast. Added to this was the iterative development process of ensuring that the player can rely on a consistent mode of interaction in creating and destroying all these different connection types.

Looking just at the interface, in most cases connectivity simply involves the player drawing a line with the cursor from the sending object to the receiving

object (some objects can both send and receive, so the direction in which the line is drawn is significant). If the player attempts a connection between two object types that is not valid, the line will have no effect and will be immediately destroyed.

Appendix A, Diagram 2 gives an overview of all the different object types available to the player; the white boxes in that diagram show the objects that have a physical form in the game world. In most cases the player can instantiate and manipulate any number of these objects prior to or during a performance. Objects that have “module” at the end of their name are scripts only; modules of code that be added to other objects to augment their capabilities, but which do not require a specific game-object manifestation in and of themselves. Game objects can be further delineated by the kind of data they are able to share, meaning the kind of performance information they can exchange with the same or different object types. The Categories of Data Flow below broadly defines the different types of information that can be shared between connected game objects:

Table 3: Categories of Data Flow

Name	Data Being Output
Default Parameters	A continuous stream of floating-point values that updates at the game's frame-rate
Proximity	Distance dynamically calculated between 2 given objects. In some cases also combined with relative rotation
Default Switching	Trigger or Boolean only
Audio Signal	Updates at Audio/Sample rate (for Pallas, locked to 44100Hz)
Pulse	Used to drive sample envelopes, sample playhead positions, synth arpeggiation, and node sequences
Note Data	Pitch and velocity
DSP Effects	…and their parameters, specific to each effect type
Camera Filter Effects	…and their parameters, specific to each effect type
Object Colouration	Sends/receives a composite colour determined by dynamic RGB values
Size, Speed of Movement, Angle of Rotation	A continuous stream of floating-point values that updates at the game's frame-rate. Based on either oscillating values (e.g. size scaling up and down) or player-driven (e.g. rotation speed only changes via player input)
Physics Properties	Dynamic relative position, velocity on each axis, magnitude of velocity
Mappable External Hardware / Software Data	Incoming MIDI / OSC

This table shows the different kinds of information that can be exchanged in a performance. In the scope of this exegesis it would be an excess of information to match all the above Categories of Data Flow with all their compatible object types, so an overview of what is possible is much more useful. For more exhaustive details see the Performance Manual in Appendix B.

Throughout this framework both default parameters (almost always manifested in game as dials and faders) and proximity interactions are ubiquitously available as a means of manipulating data between different object types. This makes it easy, for example, for the player to connect many default parameters together and use one to control many others (including for parameters connected to different object types), or to create a chain of parameters where the data is transformed at any given point, or for the player to connect an avatar

to a default parameter in order to control that parameter via the avatar's proximity (see Appendix A, Diagram 16). Since most performance objects can be readily controlled via their in-built dials and/or faders, the kind of routings described in this paragraph are the most commonly used in performances to create and destroy dynamic relationships between game objects.

Proximity as described above relates to the case where an avatar is connected to a default parameter. Proximity also has specific effects for different object types, and this is the kind of proximity that is based on fields (see Appendix A, Diagrams 7 & 8). Fields are semi-transparent spheres surrounding most object types that the player can dynamically resize at any time. The size of the field determines the radius of the effect of the connected listener object's (usually an avatar) proximity to that object (note that the player can also determine a specific avatar for that object to have a proximity relationship with). For example, for a sound source the size of the field determines the radius in which that sound is audible: if the connected avatar is within that radius they can "hear" the sound, and the closer they are to the centre of the field, the louder the sound. To use another example, the same analogy applies to a localised light-source: if the avatar is outside the field the light is completely dimmed, if the avatar is at the centre of the field, the light is at its brightest. Note that it is at a developer's discretion whether the proximity field on any given object is used. While fields can be applied to most object-types, in practice it is not always practical to do so, and it is often the case that choosing not to include a field (and thus a proximity dependence) for a given object means sacrificing some expressiveness for the benefit of stability. For example a Pulse object can be given a field, such that an avatar's proximity will determine its pulse-rate, but generally speaking it is more useful to maintain a stable tempo, rather than to expressively fluctuate it.

Switching is also widely implemented throughout the framework (see Appendix D: Video 8), and the vast majority of performance objects are compatible with default switches. Switches can be configured to respond to either cursor clicks (thus responsive only to manual triggering by the player) or avatar-proximity (in which case they have a visible field delineating their proximity threshold, within which they will be "on" and outside of which they will be "off"). As there are

various ways to automate the movement of avatars in the game world (such as placing them on a rotating platform), proximity switches can be useful for creating generative performance structures (see Section 4.1). Players also can assign specific avatars to specific switches, which again is more significant in generative structures or multiplayer contexts where we would have more than one avatar moving at once.

What constitutes an “on” or “off” state can vary greatly from one object type to the next, so what is important here is that a given switch simply sends out a signal to all connected objects telling them to toggle their state. Then locally, i.e. at the receiving object itself, this toggle is translated and executed in the way relevant to that object (e.g. switching on a sound, switching on a light, bypassing an effect, taking into account whether that response is immediate or easing, and so on).

It should be noted also that all object types can be toggled without the need for a switch object. Switches would only be used either when multiple objects need to be toggled at once, or in the case of a generative context where an automated movement (such as an avatar standing on a rotating platform) was being used to toggle an object (or a group of objects) via a switch. Aggregate switches (see Appendix A, Diagram 17) are a much less commonly used kind of interdependent switching, suited to more complex and/or generative performance setups.

Beyond the categories of Default Parameters, Proximity, and Switching, the remaining Categories of Data Flow in Table 3 are each specific to just a few types of connectivity, and while they may still be vastly important to almost any performance – such as note and pulse data certainly are – in the context of this framework it is only relatively few object types that need to “speak” any of these other languages.

Thus communication between performance objects throughout this framework can be seen to carefully negotiate a balance of generic and specific data sharing. This allows on the one hand for very rapid exploratory routing together and continuous exchange between (potentially vastly different) functions

sharing a common language, and on the other hand for discrete control of precisely timed and executed functions sharing a highly specific language.

3.2. Gameplay That Crosses the Boundaries of Instrument, Performance and Composition

In the context of gameplay, the distinction between composition and performance is not as significant as it is in the field of music. Gameplay always entails performative co-creation (in both the player's input and their reception/ interpretation).

> *Videogames participate in the struggle between authorial intent and interpretive freedom. Video games require players to create a subjective understanding of the synthesis of one or more unit operations. Games demand that players be capable of making this synthesis palpable in their own experience.* (Bogost, 2006)

Indeed as the works of both Bogost, Murray and Laurel attest to, any form of digital interactivity entails some level of performance/authorship on the part of the user, in recombining units of meaning and/or function.

> *Janet Murray offers another way to look at the relationship between narrative and technology, what she calls procedural authority: "The most important element the new medium adds to our repertoire of representational powers," says Murray, "is its procedural nature, its ability to capture experience as systems of interrelated actions."* (Bogost, 2006)

However, since this research was born out of musical performance practices, and more specifically out of electronic improvisation, acknowledging this composition/performance distinction is important. It is most significant where it has been useful in creating a space conducive to players freely crossing these boundaries between composition, performance, instrument and game, through encouraging conscious engagement with the potential for their gameplay to encompass all of these contexts.

Taking this a step further, looking broadly at gameplay as performance renders these questions of whether a piece of music/art etc. is a composition, performance or instrument somewhat outmoded. Procedural co-authorship is now so natural to multiple generations that have grown up with video games that there is nothing revolutionary about acting as co-creator of a work at the same time as one is performing within it. One could say that any kind of game in the long human history of games, i.e. including pre-video games, is a form of procedural co-authorship. However what is significant about games and digital interactivity in general is the increasingly fine-grain nature of those procedural units that continues to give players more options for more precise and detailed control as digital processing power increases. This “fine-grain nature” is the data resolution of procedural authorship, and in order to keep abreast of its expressive potential, it is necessary that frameworks such as this research are created, refined and evolved into new co-creative mediums. For this field of research this entails continual refinement in moment to moment gameplay in order to facilitate games that are more precise instruments, more nuanced performances, and more detailed and/or refined compositions. It also means a more fluidly responsive lens for navigating between these points of view.

It is in this sense that the title of this section is quite intentionally given as *gameplay* that crosses those boundaries. This is to highlight the dual nature of “play” – the difference between playing a game and playing a piece of music (i.e. performing, even if there is no audience).

> *As freeing as gaming can be, it seldom entails the straightforward possession of agency or some boundless capacity for action. In the same way that musicians – even (or especially) during their most virtuosic exhibitions – might feel as though they are getting lost in, giving over to, or being swept up by the performance and instrument at hand, so players of games oscillate between being in and out of control, playing and being played, and acting and being acted upon…Working out these fundamental tensions is what makes gameplay a dynamic, interactive experience.*
> (Cheng, 2014)

A game itself is play, it is a performance context. Play, as in performing music, is a game in the sense that it immediately creates a fictional context, a magic circle in which certain specific rules and conventions apply which dissolve as soon the performance is complete (Huizinga, 1955). Much has been said in game research about the boundaries of play and the “magic circle”, and it is not necessary to reiterate such discussions here. Suffice to say that a consciousness of the boundary of a given performance, the divide between the real and the virtual world, allows the player to “play at playing”: that is, they can be both playing a game and performing a piece of music at the same time. Thus, entirely within the magic circle of the game world they could be playing a game that is only perceived by them in the physical world in the most minimal way possible (e.g. through a screen, speakers, and the game controls in their hands). Now imagine that at the same time they are on stage in front of an audience who are also seeing and hearing the performance. The player thus has the freedom to shift their consciousness back and forth between playing a game for themselves and giving a performance for – and in collaboration with the responsiveness of – the audience, and they can and should continue to shift this conscious perspective in whatever way is most conducive to the flow of their moment to moment expressiveness. The *Pallas* performance framework has been designed to facilitate such an attitude, and this attitude is what is meant by *gameplay* of a higher-order – where the performer, by consciously authoring and navigating the relationships between units of a given performance, can thus “play” back and forth between game world play and real world.

> *Whether actively role-playing or not, players of games are tasked with straddling and arbitrating between multiple frames of mind. As Ken Hillis puts it, virtual environments in general offer "a space of performance, a multipurpose theater-in-the-round for the many components of the self”* (Cheng, 2014)

As a composition the game provides a context that makes any given performance distinctly a performance of *Pallas*. That is to say it is not a blank slate or an attempt to make a piece of abstract music-creation software that just happens to use gameplay to drive parameters. Instead, much like

Electroplankton, it provides a game world that has a unique atmosphere and an implied narrative in its distinct visual nature and in the consistency of its "feel" – i.e. its responsiveness to player-input, its animation, virtual-physics and so on. Clearly it is not a composition in the sense of a fixed linear piece of music, but rather it is a field of possibilities where visuals can be augmented and altered to a certain degree, and the audio can be created and altered to a much higher degree. For any given audio-visual performance the "piece" being performed would always be recognisable as *Pallas*. In a purely audio performance – without the accompanying visuals – the audience may not be able to tell whether or not it was *Pallas* being performed, but nonetheless it would be comprised of distinct audio qualities, behaviours and idiosyncrasies that would undeniably make it a performance of the "composition" *Pallas*.

Considering the game as a performance in its own right depends entirely upon the intention of the player giving any particular performance. For example, it is conceivable that scores could be composed specifically for *Pallas*, which could be recreated in performances with a high degree of consistency and precision (depending of course on the detail of the score and the skill of the performer). Thus any performance of such a work could absolutely be identifiable as that work. However, given the previous consideration of *Pallas* as a composition in and of itself, this would mean that we have a composition executed within a composition – so once again the nesting principle is at play.

As an instrument, the player can be creating, augmenting, deconstructing, or indeed destroying it either before, during, or after they have performed with it. Thus it is more than what would typically considered an instrument in a fixed sense. However *Pallas* can also function in fixed configurations: for example a player could quite easily set up a synthesiser in the game, plug in a MIDI keyboard and execute a regular linear score of keyboard music. This stability and consistency is as important as its potential for being dynamically altered and reconfigured. Returning to the idea of a score specifically for *Pallas,* such a piece could describe both musical content as well as timbral transformations in great detail (both of which require moment to moment stability of configuration), as well as prescribing gameplay as a means by which the "instrument" (i.e. the apparatus upon which the music is being performed) is reconfigured: that is its

constituent parts are created, augmented and destroyed. Indeed such a score would perfectly illustrate the synthesis of all the “boundary crossing” concepts elucidated in this chapter: the composition would not only include performance instructions, but also gameplay instructions for “how to build/alter your instrument”.

3.3. Flowing Data: Navigating the Shared and Translatable

Section 3.1 has already discussed communication between performance objects in some detail. In the context of “Branching and Nesting Structures in the Game World” it is important to now explore the data flows of those communications.

In order to achieve maximum expressiveness, two aspects of the communication between performance objects are important:

- How easily connections can be created, bypassed, navigated and destroyed
- How easily the data-flow can be dynamically controlled or altered

In both cases it is essential that play/performance is uninterrupted. In the field of live audio this is a precise operation. For example, even in the commercial patching environment of *Max*, because the system is not designed for live-patching, if one attempts to use it live it has a critical flaw that impedes the majority of patching functions: that is, connections between audio objects cannot be created or broken with causing at least a momentary glitch-artifact. As *Pallas* is designed for live performances that do include real-time patching, such limitations would not have been acceptable, and it fortunately came to pass that uninterrupted live patching was achievable within very few design iterations. However for any serious performer some understanding of audio processing is still necessary to avoid other potential erroneous audio artifacting. For example, removing a DSP effect from a chain takes it out immediately which in many instances (depending on the effect) can cause such artifacting. In this case it is up to the player to reduce the wet/dry level of the effect first

(which, given a practiced hand, could be executed in less than 1 second) before removing the effect. For some operations a fast automatic transition is executed, such as when bypassing an effect a fade-in/fade-out of the wet/dry setting. These are limited to contexts where it would be of no benefit to the player for having any other options for doing otherwise e.g. in the above example, there are several other means of controlling the wet/dry setting of an effect, so the player would only ever use the bypass function if they wanted to execute an instant transition, thus it is handled automatically. For all other object types where it is relevant, bypassing is readily available and can be executed instantaneously with no interruption to the flow of audio or visuals.

Regarding destroying connections, one of the key considerations is the ability for the player to easily navigate a potentially large array of connections during a performance. Early in the research the connectivity lines that the player draws from one performance object to another would, once drawn, remain constantly visible until disconnected. Once this research framework reached a certain level of complexity a typical performance involved instantiating and connecting a significant amount of objects. This resulted in a visual clutter of lines all over the game space.

The solution was thus for a line to be visible while being drawn, but then to fade quickly to complete transparency as soon as it is connected. A global list keeps track of every object's current connection-lines, and the player can thus move the cursor over a given object and press a key to either "flash" all connected lines, making them visible for a brief moment before fading again, or to toggle the visibility of all connected lines as needed (when wishing to make them remain visible) (see Appendix D: Video 9). This means that for any given object in the performance space at any given moment, the player can quickly view all its current connections.

As it is typical for many game objects to remain dormant for extended periods during a performance (while the player is attending to some other area), this ability to hide and reveal connection lines is essential for achieving a moment to moment balance between detailed views of the current configuration of specific areas of a given performance – i.e. the ability to perceive in an instant what

relationships exists between specific objects – and visual/cognitive clarity on the performance space as a whole – i.e. not cluttering it with connection lines. This once again comes back to the hide-and-reveal nesting principle, and in this sense since connection-lines are owned by both objects that they are connected to, a global game-context is responsible for managing this particular nesting function.

Destroying connections is the one significant function of the interface that sacrifices consistency for contextual suitability. While a highly refined interface such as the *Reactable* offers a powerfully simple solution to this problem, in practice, that did not suit the 3D game context. In the *Reactable*, two modes of breaking connections are possible depending on the connection in question. The first is simply moving the objects out of proximity of each other, which is not an option for this framework since inter-object proximity has too many other dependencies. The second is for the performer to use their finger to swipe a line perpendicular to any connection-line on the screen (Jorda, 2009). Since *Pallas* supports one-to-many connections, and thus a single object may have many outgoing connections to other objects, the risk of using the cursor to cut the wrong connection is the first problem. The second problem is the camera perspective. Even though the *Reactable* and its constituent components are tangible 3D forms, the interface on which these forms interact is limited to a 2D plane. However, in *Pallas*' full 3D space, where the camera typically looks down at an angle on the scene, it is not always so easy to distinguish which line a cursor is passing through, and so again the player would risk cutting the wrong connection, or accidentally cutting multiple connections.

This issue proved to be the one that has required the most design reiterations and contingencies to solve. In the end a balance was achieved by distinguishing between objects that could only receive a single instance of a particular connection type – e.g. a sample player can have only one pulse object connected to it at a time – and objects that could have any number of a particular connection type – e.g. a switch can have any number of outgoing connections.

Where only a single instance of a particular connection type is possible, a node is created (when that connection is made) on the receiving object which the player can use to break the connection. For example, since any object that can connect with a pulse object can only have one incoming pulse at a time, those objects themselves will create the disconnection node.

Where any number of connection types are possible, disconnect nodes are handled by the sending object. For example a switch may have many outgoing connections to objects that it will trigger, and those objects may be connected to any number of other switches. Thus, rather than accumulating disconnect nodes (for each switch they are connected to) on those receiving objects, the disconnect nodes can be “flashed” in a similar way to that which line-flashing (described earlier in this section) takes place. In such cases when the player executes this function on the switch object (or whichever object in question), disconnect nodes for all connected objects will appear halfway between the position of the switch object and each receiving object, which the player can use to break the respective connection.

While implementing both of these above cases was an involved solution, the result is an interface that supports the destruction of connections with no or negligible interference to gameplay flow.

In addressing the question of how easily the data-flow can be dynamically controlled or altered, I will address the Categories of Data Flow (Table 3) and briefly return to the subjects of default parameters and proximity.

In Section 3.1 the basic means of manipulating default parameters and proximity was discussed in detail. These means are the essential foundation for controlling and altering data flow throughout this framework. But there are two additional considerations further to this:

- automated parameters
- specific behavioural characteristics of the other Categories of Data-Flow

When considering the data-flow of default parameters and proximity, automated parameters are a powerful means of controlling and altering that flow. Since this research is focused on expressive gameplay for players, the notion of automating parameters would not seem congruent with this philosophy. However the history of electronic music performance has shown that, in the right balance, automated parameters can enhance creative expression, and allow even large and complex performance setups to be more easily navigated by one or a small number of performers. Take for example the LFO – by its nature an automated parameter that will continue indefinitely to run its course until altered by the user. The LFO has expressive potential because the user can alter its parameters: it is an automated parameter, but it is one whose own parameters can be dynamically altered (or indeed even themselves automated by another LFO), and in this way it becomes an instrument within the instrument: a discrete piece of functionality that can be performed *with*.

So it is with *Pallas* where we have various objects that can perform different types of automated oscillation – size, rotation, relative position, orbital-rotation position, parameters LFOs, and so on, for which we can dynamically set the rate of oscillation, and the minimum and maximum values. Various values of these objects can either be mapped directly to default parameters (e.g. the changing dynamic size of a scale-oscillating object), or they can be used to either directly or indirectly effect proximity changes: again I return to the example of a rotating platform with an avatar standing on its outer-edge. If the avatar is listening to a sound source that is next to (but not on) this platform, then the sound source's volume and panning will be oscillating as the platform causes the avatar to rotate periodically closer to and further away from the sound (see Appendix D: Video 9).

Regarding the other Categories of Data Flow, each has specific behavioural characteristics that must be considered in regards to the dynamic control and alteration of performance data:

Audio signal (including DSP effects):

Care must be taken to manage connections and disconnections of DSP effects to avoid audio glitch-artifacts or other undesirably sudden drastic changes to the audio stream. Such issues have intentionally not been smoothed over in order to retain the maximum amount of flexibility for the player to control the dynamics of those transitions.

Pulse:
(see Appendix D: Videos 3, 4, 12)

Pulses, like audio signals, operate at audio-rate. While audio-rate is used to maintain the consistency of the pulse (since a game's update rate constantly fluctuates and thus cannot provide a stable tempo) the pulse itself obviously does not output as a continuous audio signal but rather as discrete singular bursts. These can be used to directly trigger samples – either to re-trigger from their loop-start position or to trigger an instance of their currently set envelope if they have one (in which case any number of such instances can play concurrently). They can also be used to directly iterate through a synthesiser's arpeggiator. In order to use incoming pulse-data, a given arpeggiator requires incoming note data (e.g. the player holding notes on a MIDI keyboard) or the player needs to have previously locked-in some note-data. Pulses can also be used to drive node-sequences (see Appendix A, Diagram 18) – these are branches of nodes that the player can create and route to audio-objects to trigger them to play sample-loops, envelopes, or in the case of a synthesiser object, individual-notes. Since node-sequences can be of any size and branching-complexity, and the player can at any time restart, set loop positions, change sequence direction, loop both backwards and forwards, and mute any given node, the flow of a node-sequence can quickly transition from a "hands-off" automated process to a dynamically authored one.

So while the pulse in itself is one of the simplest objects in the framework, controlling its data-flow becomes incredibly nuanced when the player considers how that data is to be received, and potentially received *differently* at many different sources simultaneously.

Note Data:
(see Appendix D: Video 5 from 7:13)

While note data in the context of *Pallas* is ostensibly the same as a standard MIDI note-on note-off operation, the framework intentionally avoids using MIDI for all note-operations, for two reasons. Firstly, not using MIDI means that play speed/frequency (for both sample players and synthesizers) can be set as a continuous floating-point value, allowing for high-resolution glissandi of any length, as well as theremin-like implementations of a monophonic synthesizer. Secondly, the formatting of a MIDI message would require additional note information that is never used in this framework and would just require an additional translation step back and forth to add or filter out this extraneous information every time a note message is processed. However, care has been taken to ensure that MIDI note input and output are possible, so in these cases the translation (from MIDI data to *Pallas*' own note data format) only takes place as needed when MIDI note data is incoming (from MIDI hardware or software external to *Pallas*) or translated in the opposite way for outgoing data.

In performance most serious (i.e. technically/aesthetically skilful) uses of note data would require external MIDI hardware or software, otherwise a player would be relying on the extremely limiting QWERTY keyboard for note input (thereby also losing the potential for note velocity). Thus any further considerations of data-flow in this discussion are dependent on such external sources, as well as the real-time mapping of such sources, all of which are discussed in greater detail in Section 4.3.

Camera filter effects:
(see Appendix D: Video 7)

In order to provide consistency of interaction for players the design of camera filter effects was initially conceived to have the same connectivity structure as audio DSP effects. So for example where an audio source could have a low

pass filter, a phaser and a flanger attached, each of which are separate objects in the game world, a camera object could have a saturation, contrast and vignette filter attached, each of which would be separate objects in the game world. However there is one obvious problem with this model, that being that with sound it is possible to hear many different audio sources simultaneously. With visuals this is not possible, or if technically achievable, is not at all practical. Simply put, the player only needs to see the output of one camera-on-the-game-world at any given moment. Thus the connectivity structure of camera filters was redesigned quite dramatically so that a chain of filters can be stored and manipulated on any given camera filter object, but behind the scenes each filter (i.e. each individual unit of signal processing) is automatically processed on the game's main camera. While it would be technically possible and quite easily achievable to allow the player to instantiate any number of cameras during a performance then apply filter chains to specific cameras, thus making the interface an exact analog of the DSP-effects-to-audio-source connectivity, in practice this is quite redundant. The reason being that effectively navigating the 3D space is dependent on the way that the camera moves with the avatar, in either a first or third person view, and this is of vital importance concerning moment to moment accuracy of control – not just of the avatars themselves, but also concerning the player's ability to see and manipulate parameters. Even if the player wanted to use multiple pre-set camera angles each with its own chain of filter effects, for example to create a machinima-style narrative performance, in the virtual space this is still easier to achieve with just one camera, since the camera can instantaneously move to a new position and angle, with a new set of filter effects, as quickly as a hard cut would take place. If however to transition, e.g. crossfade, from one camera to another, and thus also one camera filter chain to another on multiple cameras, is not currently possible in this framework.

Regarding data-flow, all of this means that any given camera filter object also holds the controlling objects (and the parameters for each controlling object) for however many filters the player has instantiated in its chain. Thus a great deal of information is nested in every camera filter object, making it essentially impossible for the player to see, at a glance, how each effect in a

filter chain has been set. Again, this can be contrasted with an audio source where any connected effects and their parameters would be separate objects that are (along with their parameters) all simultaneously visible. So while navigating and adjusting parameters on a camera filter object is relatively fast and simple, it lacks the constant visual feedback of most other object types, and in this respect it substitutes the constant availability of parameters for the condensing of information.

Object colouration:
(see Appendix D: Video 6 from 5:05)

The data flow of object colouration relies of a very simple implementation of default parameters. Using dials and faders, the colouration object allows players to dynamically define values for red, green, and blue (RGB), thus creating a single colour that is a composite of those values. Then any compatible object connected to this colouration object will take on that colour (and any given colouration object can output to any number of receiving objects). This is most useful in performance for dynamically controlling the colour of light sources, where it is possible to have any number of light sources being controlled by any number of colouration objects.

Size, speed of movement, angle of rotation:

These have been covered earlier in this Section in the discussion of automated parameters.

Physics properties:
(relative position, velocity on each axis, magnitude of velocity)
(see Appendix D: Video 9 from 6:34)

Almost all game objects in this framework use a vastly simplified physics that essentially makes sure that they are obstacles (i.e. they can’t pass through

each other), and that they have basic gravity to keep them grounded. Aside from those functions, any other motion-based relationships they have are faked in code. This is because physics simulations are highly CPU intensive, and in a framework such as this where players have the freedom to instantiate any number of objects at will, the burden of physics simulations on all those objects, combined with of intensive audio and visual processing, is too great for the average computer given the current state of computing power.

Thus in this framework a specific object pair has been created with which the player can explore physics-simulation interactions. This pair consists of an anchor-object and a motion-object. The anchor is needed so that the player can easily interact with the parameters of the motion object, freeze its motion when needed, and as a point from which proximity from the motion object can be measured and output. If there was no anchor object all of those interactions would be very difficult to fluidly execute on a constantly moving object.

In regard to data flow, the anchor object outputs three distinct sets of data: its relative distance to the motion object (output as three separate parameters: X, Y and Z distance), the velocity of the motion object (output as three separate parameters: X, Y and Z velocity) and the magnitude of the velocity. However, what the data flow really pertains to is the dynamic movement of the motion object. In this sense, controlling those parameters means skilfully applying forces to the motion object, and this is where some of the most dynamic interactions between player, avatar and environment can be seen in this framework. Using either the cursor-clicks, cursor-gestures, or avatar movement, the player can knock the motion object around (the motion object is by default a sphere, but can be substituted for other shapes). If this is done on a flat plane, the player's control will be very direct and predictable, however if on an undulating terrain, then clearly a much more nuanced performative dialogue can take place between player, avatar and environment, where the player/avatar has a richer feedback loop of responding to the nuanced motion of the object bouncing and rolling through such a terrain. Extend this then to a multiplayer context where, for example,

multiple player-avatars could engage in a kind of soccer match with this one object type (which is outputting up to seven different parameters (per physics object, remembering that we could have more than one) that could be used to control any number of audio or visual parameters), and it is clear to see the broad expressive potential of such a physics-driven data flow.

In addition, other object-types can also be made to move with the motion-object, thus matching its position (though not its rotation). This allows the player to set a proximity interaction in motion, for example between an avatar and a sound source, and let the physics control the dynamics of that interaction (e.g. the sound source rolls around the avatar through the terrain), mediating only if and when they choose to.

Mappable external hardware/software data (incoming MIDI and OSC):
(see Appendix D: Videos 10.2, 10.3)
(see Section 4.3)

3.4. In-Game Interfaces in Narrative Play

> *The most ambitious promise of the new narrative medium is its potential for telling stories about whole systems. The format that most fully exploits the properties of digital environments is…the simulation: the virtual world full of interrelated entities, a world we can enter, manipulate, and observe in process.* (Murray, 1997)

For the first year of its development this framework was focused solely on becoming a means of allowing players to create original audio visual performances. As such it did not prescribe any goals or game objectives for the player.

At a certain point in its development, the real time authoring of text-based conversations became a part of the framework, initially intended as a way for one or many players to include narrative in their performance where they could

potentially create machinima-style films within *Pallas* where music, visuals and text-based conversations could all be performed in real time (and thus potentially improvised) (Cameron, 2009).

While this implementation of a “performed-conversation object” was technically successful, pursuing a machinima angle did not seem the best use of the core strengths of this research, as it leaned more heavily on the linear end-product rather than the branching potentials of the performative event. So the next unfoldment became to explore the real-time creation of narrative objects, with a mind to moving towards the real-time creation of narrative puzzles. To this end a point-and-click adventure paradigm was employed, which eventually led to forking this research into the two distinct streams of Performance and Story Mode.

The point-and-click adventure, championed by Ron Gilbert’s *Maniac Mansion* (1987) and its underlying scripting system known as *SCUMM* (Script Creation Utility for Maniac Mansion) (Gilbert, 1987), is a genre of interactive narrative gameplay that relies on cursor-based object interactions. It typically includes gameplay systems for verbs (or at minimum a single “interact-with-object-x” verb), inventory, and branching dialogue trees. *SCUMM* would subsequently serve as technical foundation of a generation of highly influential narrative adventure games (Bevan, 2013) from Lucasfilm Games (later to become LucasArts) including *Loom*, for which the *SCUMM* interface was adapted to allow for a focus on musical functionality (Moriarty, 2015, Maher, 2017). The point-and-click adventure genre typifies the first large-scale cultural adoption of Murray's “new narrative medium…the virtual world full of interrelated entities, a world we can enter, manipulate, and observe in process” (Murray, 1997). While the SCUMM system didn't aspire to great heights of sophisticated storytelling, it had a significant historical impact (though indirectly, as LucasArts’ competitor Sierra was much more successful in popularising the point-and-click genre) through its crystallising of a tightly constrained set of *types* of interrelated entities. This made the navigation of such narratives much more readily accessible to players, and cleared many interface roadblocks, to thus enable players to focus on “playing the narrative” without the added complexity of figuring out *how* to play the narrative.

Applying a point-and-click adventure interface to *Pallas* began very simply: the player could place a default 3D cube into the scene. This could then be assigned a name for whatever object it was meant to represent, and then it could be routed to verbs or other objects, whereby text-based responses could be authored for each of those routings. For example if the object was an apple, and the player routed the apple to the “look at” verb, then they could author a response to the command “look at apple” such as “it’s red and shiny”. This could all be done, for any number of verbs or other objects, in real-time in the game and exported as a text-file for each object (this exporting could even be done invisibly to not interrupt gameplay). These text-files would contain the object’s name and a list of all the verbs and other objects that it had responses to, and what those responses were.

Aside from the obvious problem of how, in real-time, to make an object look like what it was intended to be (how to access and assign an appropriate 3D mesh to replace the default cube), the system relied too heavily on a convoluted interface. Thus not only would it have been unsuitable for performance in front of a live audience, but even as a means for developers to performatively improvise the authoring of narrative objects and puzzles it proved to be more cumbersome than simply doing so in non-real-time.

Nonetheless, realizing that such a system was more useful for developers to perform with than for players led to an important augmentation to this research: that was considering not just player-centric, but also developer-centric performative expressiveness, which will be discussed in depth in Section 5.1.

Despite its shortcomings, the act of manifesting this real-time narrative object system had several significant benefits that have remained relevant throughout the course of this research. The first being that non-real-time authoring of responses between verbs-and-object or objects-and-other-objects in this framework is incredibly rapid, since at first it had been designed to function in real-time. The second being that, in programming a system whereby a given object would require only a string of text (e.g. the name of another object or verb) in order to know what response to give, it became apparent that in this

context verbs are equivalent to objects and thus verbs could be defined contextually for each object. This means that, instead of choosing from a global list of verbs that would apply to all objects in the game, the developer can, for any given object, define any number of unique verbs that apply only to a given object.

This realization thus had a dramatic impact on the narrative interface for the game's Story Mode and its potential for both communicating expressiveness and for engaging the imaginative expressiveness of the player. Simply put, this meant that what is possible depends upon the object being interacted with. This also carries a clear philosophical implication: it is a bottom-up approach where each object (which includes interactions with game-characters) is valued for what it uniquely contributes to the field of interaction and the narrative context, rather than a top-down approach where the developer/narrative applies an unchanging set of verbs to any situation. This also resulted in nesting the verbs as nodes within each object, to be hidden and revealed by the player as needed: a game-world manifestation of the notion that the actions to be performed on a given object belong to that object.

The narrative adventure game *A House in California* (Elliot, 2010) also played an important role in the development of this particular concept. Although manifesting a very different solution, it illustrates a unique example of the expressive efficacy of changing the availability of verbs to suit (and to effect) changes in the narrative.

The final and far-reaching factor in creating in-game interfaces for narrative play, was in looking at the entire performance system and, step by step, applying it to the design of puzzles. Since the performance system has such a broad range of potentials for many styles of audio-visual interaction, this is an ongoing undertaking that will extend far beyond the scope of this research.

Essentially this process entails either pre-configuring single performance objects, or pre-routing combinations of performance objects, hiding away any parameter controls that are extraneous to the context of the particular narrative/puzzle element being created, then configuring an interface that is consistent

with the visual/architectural/environmental/narrative atmosphere of the area of the game world in which the puzzle in question is situated. The final step is then to create the means to evaluate the “success-state” of the interaction, by which the game recognizes the puzzle is complete and unlocks the reward (or advances the narrative). This whole process is discussed in detail in Section 5.1.

See Appendix D: Story Mode Video for demonstrations of all concepts explained in this section.

4. Branching and Nesting Structures in the Real World

4.1. Co-Creation of Place, Generative Structures and the Artist-Artwork Feedback Loop

From the dialectic perspective of this research, where players are part of the interdependent interplay of units of which *Pallas*’ framework is comprised, co-creation of place in an inseparable part of gameplay, or more specifically, the whole experience of playing the game.

Co-creation of place in this context refers to the performative feedback loop of artwork and artist creating each other (Heidegger, 1960). This applies equally to *Pallas*’ Performance Mode and Story Mode. In both cases gameplay has been designed with the intention of encouraging the player’s awareness of themselves in the real world as player, and of their power to manifest their personal creative intentions in playing the game well (De Koven, 1978). By encouraging an awareness of gameplay as performance, regardless of whether there is an audience, this process of manifestation means that the “site” at which the performance happens is always shared between the real and virtual worlds. The better a player understands this, and the more fluidly they can navigate their awareness between the two, the more effectively they can achieve mastery of both technique and aesthetic intention. This is where I would make the distinction between gameplay and performance. Any given play session could potentially be both of these things at once, but without an

awareness and conscious engagement with the atmosphere being created in the real world, then gameplay can remain gameplay but never truly manifest as a performance.

In this sense, the feedback loop for the player/performer requires a new kind of discipline from that of the non-game musician/performer. This is due to the fact that a game-as-instrument, such as *Pallas*' performance system, comes with a context, an implied narrative, that is compelling in its own right. Thus the discipline of the player/performer is to not abandon themselves to the personal satisfaction of creativity in compelling creative (virtual) space, but to at once hold that space in their awareness and bring it out to the physical space where the performance becomes manifest.

This kind of co-creation is an ongoing creative unfoldment and, as has been discussed, the player of *Pallas* is a constant dialogue with all the elements of a performance, all the game objects and audio-visual gestures and movements, and the relationships between all those elements. But further to this, it is a co-creation between the player and the game designer, between the player and any other players (in the case of multiplayer), between the player(s) and audience (whether live or asynchronous in the case of a recorded performance), between player and score (if performing a pre-composed piece), and between a community of player-performers who are mastering, evolving and sharing their techniques an aesthetic sensibilities. All such relationships also rely on this dynamic interplay of real and virtual engagement.

Creating generative structures in the virtual space is a strategy that can help to facilitate this co-creative interplay. This is precisely due to the fact that executing a generative structure entails, or at least grants the potential, for setting a process in motion and allowing it to unfold: to sit back in the real world and allow and observe the virtual running a particular course, thus gaining perspective on one's agency when one does choose to interact with it. *Pallas*' performance framework has no means nor need for locking out a player's control, so even while executing a generative structure the player can at any time intervene to adjust or adapt any given part of that process. While generative systems such as *Nodal* (McCormack, 2015) have been explored in

this research, creating a dedicated generative system was never the intention. However through the course of creating the performance framework it gradually became apparent that a great deal of scope for interdependent automation already existed.

This is an inherent benefit of realizing the performance capabilities of video games: that many systems throughout the history of gaming already exhibit generative behaviours and many more have the inherent potential to be generative if, for example, certain gameplay constraints were lifted. Even the simplest video games exhibit such a fine grain of dynamic procedurality that it takes only a few well-crafted rules of behaviour between discrete entities, as evidenced by digital realizations of the elegantly simple *Game of Life* (Conway, 1970), to create a system that can output a dynamic array of precisely crafted responses that, like *Nodal*, they can be well suited for mapping to performance parameters such as audio (Oliver, 2003).

Put simply, game world entities are commonly programmed to explore, to seek out other entities, procreate, multiply, die, and so on: all such behaviours and the resulting movements and interactions between characters in the game world are rich sources of generative information that can easily be mapped to performance parameters. While *Pallas* does not employ any of these video game tropes, favouring instead direct-player control and placing NPC behaviours outside the scope of this research, the research does acknowledge this vast untapped potential in the field of performative gameplay, and in particular what it would contribute to expressive performance in gameplay.

Pallas employs a dynamic interdependence between many object types that can be made responsible for triggering audio/visual functions of objects, and a few specific object types that create endless dynamic motion. By setting in motion processes at different phases (such as different rotation speeds) and creating networks of interdependent switching (toggling various audio and visual objects based on the position of automatically moving objects) systems can be skilfully crafted and set in motion by a player which, while not emergent in the true sense, can exhibit an aesthetic outcome greater than the mechanical means by which the system executed it (see Appendix D: Videos 12.1, 12.2).

Returning to the concept of expressiveness, it could be said that this represents a form of asynchronous expressive performance: more akin to composition in the sense that the player executes their creative faculties before the performance takes place. However, once set in motion the player can always make the decision to intervene or rejoin the performance, thus this question of expressive engagement once again depends on the intention of the player in any given moment.

4.2. Local and Non-Local Multiplayer Affordances

The addition of other human players can dramatically alter the landscape of what is possible in performance. At the time of this writing, only local multiplayer on a shared screen has been implemented as a playable feature in the framework. However online multiplayer has been both designed on paper and prototyped sufficiently to show that it is not only possible but easily achievable.

The great benefit of multiplayer is that it alleviates the limitation of only having one avatar in motion at any given moment, or of only directly moving one non-avatar object at a time. In single player performance, particularly when also not using an external (such as a MIDI) controller, these are both a continually challenging aspects, especially avatar movement as it is the primary means of controlling volume and panning of all the audio sources in a scene. It is not uncommon to have three to five audio sources playing simultaneously, each with its own avatar-listener. In such an example a single player is typically adjusting the parameters on the sound source, and often the parameters on attached effects, while attempting to move the avatars (one at a time) in order to maintain the volume and panning balance between all the sounds in order to create an aesthetically pleasing performance.

Local multiplayer (see Appendix A, Diagrams 3 and 5) as it currently exists in the framework, has several distinct manifestations, each of which has different affordances for players and audiences (in cases where an audience is present):

• Shared-Computer: all players share one computer and thus one screen. The audience sees the same visual output that the players are seeing (typically a duplicate, shown on a projector screen). The same audio output is shared by all players and audience (usually speakers, potentially a combination of speakers for the audience and headphones for the players). This can either take the form of each player controlling and switching between any available avatars, or it could involve one or more of the players dedicated to external hardware control (such as playing a MIDI keyboard, see Section 4.3)

• Audio-Hardware Network: each player has a separate computer and thus their own screen (and headphones if desired). Each of their virtual game-worlds is discrete and not digitally connected to each other (i.e. they are each running a completely separate instance of the game), but the audio from all instances of the game is routed from each computer to a single real-world audio-mixer and output through a stereo system for all players and audience to hear. In this case two visual options exist: to display one instance of the game on a projector screen for the audience, or otherwise to switch between the different instances on the projector screen.

Non-Local Multiplayer (see Appendix A, Diagrams 4 and 5) refers specifically to online play. Here each player has a separate computer and thus separate audio-visual output. One audio-visual output can be displayed to the audience at every different physical location of each player in the multiplayer network. Given the physically separated nature of each site of performance, players have more freedom to be selective about which audio aspects are part of their local instance of the performance. For example, any given player can mute any given sound on their instance only, without that muting affecting what anyone else is hearing. This creates the unusual potential for players to synchronously be engaged in the same performance while each experiencing a potentially dramatically different version of it.

Here also we have the most literal realization of the real-world/virtual-world interplay, where the real-world performance is distributed among many discrete physical locations, but the entire process of communication bringing those instances of the physical performance into being is taking place through gameplay in the shared virtual space. In fact this sharing of the virtual space is

only an illusion (as it is in most online games), as a different instance of the game is running locally on each player's computer. Leveraging this illusion – i.e. not simply assuming that all information needs to be synchronised across all instances – is what allows *Pallas*' online performance system the additional flexibility for players to make those selective decisions about how they choose to locally experience the collective performance.

Across all forms of multiplayer, it is clear that by introducing even one additional human player, the expressive potentials of the performance are greatly increased. Ideally the players' aesthetic intentions for the given performance are congruent, and they can get on with letting the piece unfold as it needs to, allowing for an interdependent awareness of each other's needs in that unfolding. Thus not only can the technical challenges of fluidly navigating and manipulating complex performances structures be distributed between players and thus overcome (or at least eased), but also the cognitive challenges of keeping track of everything that is happening in such structures.

4.3. Hardware/Software Interfaces and Real-time Mapping Strategies

Much has already been said about default parameters, connectivity/routing and avatar control. To recap what is relevant here: dials, faders, toggling objects on/off and inter-object connectivity all rely on cursor interactions, avatar movement relies on QWERTY keyboard controls, and quick instantiation of objects during performance is done with QWERTY hotkeys (see the Performance Manual in Appendix B for exhaustive details).

In Story Mode the cursor is also used to handle verb-object interactions and inventory, and the avatar always carries a synthesiser, used for puzzle solving, that is played using the QWERTY keyboard to input an octave of notes, (this is also an option for note-input in Performance Mode, but as previously discussed is best avoided).

These are essentially all the default modes of interaction in the game, and all are pre-mapped and unchangeable for any given object.

However, real-time mapping of both MIDI and OSC is implemented in *Pallas* such that its simplicity makes it a powerful means of transforming the nature of a performance. First because it opens up performance to a whole new range of gestures, and in the fields of MIDI and OSC the vast range of different hardware/software means that a huge amount of different modes of gestural performance are possible. Secondly as it allows for not just the creation but also the alteration of mapping, so once created mappings can also be shut off, or the incoming data can be scaled. This means that the mapping process itself can be an expressive part of a performance. Thirdly it allows an alternative means of distributing the demands of a performance between multiple human players: where one player can focus on managing multiple avatars, and another (or several others) can be controlling any other performance parameters via MIDI or OSC, or playing notes into performance objects, e.g. playing an in-game synthesiser object via a hardware MIDI keyboard. Thus clearly amplifying the expressive potential beyond what a single player could achieve.

This mapping system is a descendent of *Ableton Live's* MIDI mapping paradigm, and has been refined through my own past design and implementation of real-time MIDI and OSC mapping systems in *Max*. For any given dial, fader or MIDI-note compatible game object the player can cursor over and hold a specific QWERTY key, making that game parameter ready to receive a MIDI mapping (see Appendix D: Videos 10.2, 10.3). Then by simply moving a continuous-control or playing a note on the MIDI device, that in game parameter is mapped to the MIDI control. In the case of a note receiving game object, the player can use a modifier key to determine whether all notes of the MIDI note device will be accepted by the game object, or just the specific note that the player has pressed to make the mapping. This option to map just a single note exists for the purpose of "trigger pad" style sample performance, where the MIDI notes of a hardware device can be used as a bank of sample-triggering buttons, so any given note can be assigned to a specific sample (thus a specific game object). The player can use the cursor to toggle any mapping

on or off at any time. Of course, as is usual for such a system, the mapping potential is for one-to-many where, for example, any given MIDI hardware control (e.g. a single dial) can be mapped to a limitless number of in-game parameters to control them all at once.

The OSC implementation is more challenging to manage than MIDI since in many cases OSC outputs a continuous stream of data. Thus when there are multiple separate OSC streams incoming at once, the game would have no means of knowing which one, at any given moment, it should be mapping a given parameter to. While there are many possible strategies for dealing with this issue, a unique solution was designed for *Pallas*. The player can pre-input up to 10 OSC addresses that they wish to use into a numbered list before the performance. Then during the performance they can input numbers 0-1 on the QWERTY keyboard to assign the corresponding entry in the list to a cursor-selected dial or fader in game. Then, as with MIDI, they are also free to toggle the mapping on and off at any time. While this system limits the available mappings to a set of 10, it facilitates uninterrupted performance with no overlaid interface obstructing the view of the game world (note this OSC mapping system has not been implemented in the current build).

A custom OSC application for controlling *Pallas* via iOS and Android was also created for this framework. However, due to the technical challenges of sharing iOS/Android software, it has not been included in the Appendix data. This application has a touchscreen and accelerometer interface for avatar movement, text-based dialogue input, and the carrying of virtual objects, allowing for a basic implementation of local multiplayer (note that a more advanced form of local multiplayer is possible via a PlayStation DS4 controller, for which the game is also pre-mapped. See Appendix B, and Appendix D: Video 10.1 for details).

Applying those design principles whereby interface overlays are avoided and performance is not interrupted, a further mapping system, although not currently implemented in this framework, could be created for QWERTY controls: both to map QWERTY keys to trigger any switch-compatible object (e.g. toggling sound

sources, lights, camera filters, and so on), and to allow the player to re-map the hotkeys for object-instantiation. Such a system would function in the same way as the MIDI mapping system already described.

As discussed in the introduction to this research, great care has been taken to give the experienced player the option to perform with no extraneous text, numbers, or other kinds of alpha-numeric or similar abstract symbolic data that is not a part of the fictional context of *Pallas* as a game world. These MIDI and OSC mapping systems have also been designed with this intention, and thus the mapping process is all but invisible to an audience, relying only on the slightest visual feedback to let the player know when an operation has taken place.

Both MIDI and OSC are not simply hardware-to-software protocols, but also allow for software-to-software communication. This means that other music/controller software can also be used to control any aspect of *Pallas*. Taking this a step further and applying the branching and nesting principles expands these concepts into a rich field of future research: interdependent networks of different games/applications sharing real-time performance data.

Developers of such inter-game performance communication would not be limited to MIDI and OSC communication, but could easily design their own protocol. However, I limit this discussion to MIDI and OSC since this retains the potential for also communicating with existing music/controller software.

5. Performative Narrative, Puzzles and Game Objectives

5.1. Performative Developer-Authoring of Narrative and Puzzles

At a certain point in this research the development of narrative and puzzle elements could be seen as a valuable augmentation that was contributing new material to the expressive intention of the work.

It soon became clear that this expressiveness no longer just concerned the fluidity of gameplay in Story Mode, but included those instances where a developer could use *Pallas*' performance mode to improvise and export content that could thus be imported into narrative and puzzles for Story Mode.

The principle that thus became apparent was: the more performatively expressive the design process is for the developer, the more effortlessly they can translate this to expressive gameplay for the player.

While this did at first mean literal improvisatory authoring and then exporting content from Performance Mode to Story Mode, what later often occurred was an internalised understanding of the performance system. So in the same way a composer of music does not need to play the music before they can write it, so it is possible to design puzzles that alter or hybridize performance objects, and know ahead of time how they will behave. Essentially, the skilled developer can execute performative elements of the puzzle in their mind and know how it will thus respond for the player.

However, for myself this internalised understanding continually evolved with my improvisatory practice and with the continual augmentation and refinement of the performance system. Thus even without explicitly entering into a space of narrative/puzzle authoring, such solutions would emerge naturally out of improvisation or even out of the process of testing and bug-fixing a new component.

To expand on the process introduced in Section 3.4:

- either single performance objects are pre-configured, or combinations of performance objects are pre-configured and pre-routed
- parameter controls extraneous to the context of the narrative/puzzle in question are hidden away
- if necessary a new interface is configured (and new visual components created as needed), consistent with the visual/architectural/environmental/narrative atmosphere of the site at which the puzzle is situated in the narrative game world
- a means to evaluate the “success-state” of the interaction is implemented, by which the game recognizes the puzzle is complete and assigns the reward. This is often also a hybrid process where, for example, the completion of the puzzle might depend on the player setting 3 parameters to the correct value. Thus 3 switches are used, one to test each parameter, and an aggregate switch tests the state of all 3 switches, and if all are satisfied then the puzzle is complete.

5.2. Performative Gameplay of Narrative and Puzzles

At this point the performative nature of *Pallas*’ Story Mode should be apparent due to the fact that, as has been discussed, the expressive content of Performance Mode has filtered down into the gameplay elements of Story Mode.

This research started with the creation of a performance framework, initially based on improvisation, where the player has the maximum expressive freedom. Gradually it became clear how certain elements could, through the expressive engagement of developers, be crystallised and incorporated into fixed units of narrative and game-objectives. Then finally those units could once again be assessed for their player-centric expressive potential, and adjusted/reiterated to maximise that expressiveness.

The unique aspect of this last step, compared to the rest of the research, is that this expressiveness must also be measured against the given narrative and aesthetic context in which that puzzle is situated. That is, it is not sufficient to simply maximise the expressive potential of such puzzles, they must also be congruent with their narrative/aesthetic context otherwise the player's imaginative engagement in the puzzle is diminished. This imaginative engagement is another kind of feedback loop, and although this exegesis has already touched on the importance of the consistency of the fictional world in helping to maintain the integrity of a performance, the inclusion of narrative progression in this consideration adds another dimension to this feedback loop. For example, if a given puzzle looks or feels as if it is not congruent with its surrounding fictional context – both in the sense of where it is situated in the game space, and where in the timeline of the game's unfolding narrative – then the player will have neither the desire nor the means to fully engage with it on a narrative level, and solving it would thus simply be a mechanical process of getting it out of the way (rather than performing it as part of the unfolding story that they are co-creating) in order to hopefully return to the consistency of the narrative context.

Having no game-objectives or intrinsic goal, *Electroplankton's* fictional context could be easily subverted by an adventurous or antagonistic player without necessarily destroying the expressive feedback loop of "performing with" that context. However, the same cannot be said for games in which player-performed music is an essential mechanism for advancing the narrative. In such key works as *Loom* (Moriarty, 1990) and *Zelda: Ocarina of Time* (Miyamoto, 1998) musical gameplay (which in both cases involves the playing of specific melodies to solve puzzles) is inseparable from narrative context, and as such an antagonistic or subversive musical performance by the player would disturb or shatter that fiction. *Pallas of Vines*, having both Story and Performance Modes, is open to both forms of subversive play. In Performance Mode there is no limitation imposed on the player to force them to make their music harmonious with the game context, indeed such exploratory play could even lead players (and audiences) to expand their ideas about what constitutes a performance that is harmonious with *Pallas'* aesthetic.

> *Generally speaking, people know that things work better when they respect the limits of a mimetic world as indicated by its structure and affordances as well as the model of it that people are building through experience. In exchange for this complicity, people experience increased potential for effective agency...People may likely push on the edges of a mimetic world as part of exploration or even in an effort to hack it. Designers need to be flexible and to apply new constraints when they observe actions that disturb the desired structure of experience.* (Laurel, 2013)

> *Play is about more than make-believe; it's about re making belief, redrawing frontiers of the imagination through performances of actions, identities, and ideologies previously unfulfilled (or assumed to have been outright impossible). Inherent in creative and critical play is an element of virtuosity, which…involves exceeding "the limit of what seems possible, or what the spectator can imagine [and] insistently mobilizing, destabilizing, and reconstituting borders"* (Cheng, 2014)

While such exploratory play can also expand a player's experience in Story Mode, here the context of any given puzzle is the primary determinant of what constitutes the aesthetic congruency of such gameplay. While *Loom* and *Ocarina* both have just one very simple musical interface for puzzles, the range of possibilities for designing different musical puzzle interfaces in *Pallas* is as broad as the performance framework. This means that for a given puzzle there may be more or less constraints than for another, and thus a greater or lesser opportunity for a player to subvert the developer's aesthetic intentions. For myself as a developer, these constraints are put in place both to make the puzzle design process manageable, and to focus in the player's attention on the few parameters that are important for understanding a given puzzle (which is especially important for new players). This is a crucial distinction in the context of this research, as "maximising expressive potentials" necessarily means also allowing players great scope both to extend their skills through exploratory play (which includes the scope to make their own mistakes), and to intentionally subvert the intended or common uses of the game in order to expand their

palette of performative techniques or make an aesthetic statement about the nature of the game itself.

In the context of a fiction in which players must develop new skills in order to advance the narrative, such configurative constraints function as pathways and signposts without which a player's expressive engagement can become dispersed or disrupted through a sense of unfair difficulty, ambiguous motivation, directionlessness and so on. Thus contrary to Performance Mode, and somewhat paradoxically, Story Mode relies greatly on constraints in order to focus and amplify the player's expressive engagement with the narrative, and these constraints themselves must also be congruent with the game's fictional context.

> *Some constraints on interactors' choices and actions are technically essential to any designed interaction. The question is how those constraints should be determined and expressed. Some explicit techniques for introducing constraints…can be destructive of people's engagement in the activity by forcing them to "pop out" of the mimetic context.* (Laurel, 2013)

6. Conclusion: Expressive Dynamic Relationships for Player and Developer

The overarching principles displayed in this framework are centred on the dynamics of relationships, and the understanding of the moment-to-moment context in which aggregate relationships can be reframed as a single expressive unit in a higher-order context. As both player and developer relationships with game spaces and real world performance spaces become more sophisticated, and players/developers are more readily able to navigate between them, the relationship between performance and gameplay will thus continue to develop into new hybridized mediums where original creativity is paramount: co-creative dialectics between developers, players, hardware and software performance components, and responsive game worlds.

In this research dialectic interactions underpin all the performative interactions available to both player and developer. By including the perspective of

developers and their expressive potentials into this framework, this dialectic interaction can be seen to take place not just between all the units involved in and synchronously present in any given moment of gameplay, but also asynchronously between the performative moment and the developer's pre-authored intentions and narrative communications.

As systems of this nature continue to be refined in the future, modular components of both code and concept can be translated, evolved and integrated into new games that will continue to facilitate the expressive potential for developers to mirror the performative experience of players during the development process. As players, by playing performatively, continue to master and find their own ways to expand these expressive potentials, they evolve and feed back new techniques that developers can again adapt and integrate into their own performance techniques and performative development frameworks in subsequent works, and thus a community of creative expressive performance and play continues to unfold.

In conclusion, the original contributions of this work to the field of videogames and expressive gameplay can be summarised as follows:

- expressiveness in gameplay has been defined, as it relates to the realisation of player intention during gameplay. Subsequently a measure by which expressiveness in gameplay can be planned for (before the fact) and analysed (after the fact) has been devised and demonstrated
- through a practice-led approach a game has been designed, created, and demonstrated to effectively unify audio and visual performance with gameplay dynamics and narrative context. This is all encompassed in an expressive, co-creative framework for performative gameplay – a collection of game objects, behaviours and play-contexts framed by a dialectic principle of branching and nesting structures of interdependence
- by dividing this framework into Story and Performance modes, key contingencies have been illustrated that allow both modes to remain expressive and narratively contextualized. Broadly speaking, these are designing puzzle and narrative constraints in Story Mode that retain sufficient co-creative affordances for player expressiveness, and in Performance Mode

providing sufficient game world context while maximising the player's ability to co-create within, or subvert, that context

- Bogost's unit operations, Murray's "virtual world full of interrelated entities", and the author's own discussion of branching and nesting structures have been practically applied, thus demonstrating the value of those theories as framing principles that can enhance the expressiveness of moment to moment gameplay, when both game design and player performance/creation participate in the formation of dynamic dialectic gameplay systems

Bibliography

Books

Bogost, I. 2006, *Unit Operations: An Approach to Videogame Criticism*, MIT Press, Cambridge

Bogost, I. 2010, *Persuasive Games: The Expressive Power of Videogames*, MIT Press, Cambridge

Caillois, R. 1961, *Man, Play and Games*, University of Illinois Press, Illinois

Cheng, W. 2014, *Sound Play: Video Games and the Musical Imagination*, Oxford University Press, Oxford

Collins, K. (ed) 2008, *From Pac-Man to Pop Music: Interactive Audio in Games and New Media*, Ashgate, Hampshire

Collins, K. 2008, *Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design*, MIT Press, Cambridge

Collins, K. 2012, *Playing with Sound: A Theory of Interacting with Sound and Music in Video Games*, MIT Press, Cambridge

De Koven, B. 1978, *The Well Played Game: A Player's Philosophy,* Anchor Press, New York

Heidegger, M. 1960, trans. Krell, D. 2008 "The Origin of the Work of Art" in *Martin Heidegger: The Basic Writings*, HarperCollins, New York

Huizinga, J. 1955, *Homo Ludens: A Study of the Play Element in Culture*, Beacon Press, Boston

Kwastek, K. 2013, *Aesthetics of Interaction in Digital Art*, MIT Press, Cambridge

Laurel, B. 2013, *Computers as Theatre: Second Edition*, Addison-Wesley, Boston

Liebe, M. 2012, "Interactivity and Music in Computer Games" in *Music and Game: Perspectives On a Popular Alliance*, Springer VS, Berlin

Minsky, M. 1986, *The Society of Mind*, Simon and Schuster, New York

Moorman, P. 2013, *Music and Game: Perspectives on a Popular Alliance*, Springer VS, Berlin

Mosely, R. 2016, “Nintendo’s Brand of Ludomusicality” in *Keys to Play*, University of California Press, California, Chapter 5-1, pp. 243-250

Mosely, R. 2016, “The Ludomusical Emergence of Toshio Iwai” in *Keys to Play*, University of California Press, California, Chapter 5-3, pp. 258-263

Murray, J. 1997, *Hamlet On the Holodeck: The Future of Narrative in Cyberspace*, The Free Press, New York

Oxford Press, 2018, “Oxford Dictionary of English” in *oxforddictionaries.com*, Oxford University Press, Oxford

Peters, G. 2009, *The Philosophy of Improvisation*, University of Chicago Press, Chicago

Strank, W. 2013, "The Legacy of iMuse: Interactive Video Game Music in the 1990s" in *Music and Game: Perspectives On a Popular Alliance*, Springer VS, Berlin

Upton, B. 2015, *The Aesthetic of Play*, MIT Press, Massachusetts

Articles, Journal Articles and Research Papers

Aallouche, K. et al, 2007, “Implementation and Evaluation of a Background Music Reactive Game” in *IE '07 Proceedings of the 4th Australasian conference on Interactive Entertainment*, RMIT, Melbourne

Begy, J. 2010, *Interpreting Abstract Games: The Metaphorical Potential of Formal Game Elements*, Masters Thesis, MIT, Cambridge

Bencina, R. 2005, "The AudioMulch Process: Software Development in Musical Practice" in *Australian Computer Music Conference 2005 Proceedings*, Queensland University of Technology, Brisbane

Bencina, R. 2005, “The Metasurface – Applying Natural Neighbour Interpolation to Two-to-Many Mapping” in *International Conference on New Interfaces for Musical Expression 2005 Proceedings*, University of British Columbia,

Vancouver

Cameron, D. & Carroll, J. 2009, “Encoding Liveness: Performance and Real-Time Rendering in Machinima” in *Proceedings of DiGRA 2009 Conference*, London

Carlson, C. & Wang, G. 2011, *Borderlands: An Audiovisual Interface for Granular Synthesis*, NIME 2011 Proceedings, University of Michigan, Ann Arbor

Carroll, J. & Cameron, D. “Machinima: Digital Performance and Emergent Authorship” in *Proceedings of the 2005 DiGRA Conference*, Finland

Fernandez-Vara, C. 2009, “Play’s the Thing: A Framework to Study Videogames as Performance” in *Proceedings of the 2009 DiGRA Conference*, London

Georgen, C. 2015, “Well Played & Well Watched: DOTA 2, Spectatorship, and Esports” in *Well Played*, Volume 4, No. 1, ETC Press, Pittsburgh, pp. 179-191

Gilbert, R. 1986, *Maniac Mansion: Proposal*, game design document, Lucasfilm Games, California

Gilbert, R. 1990, *Mutiny on Monkey Island: a step by step breakdown of the game flow*, game design document, Lucasfilm Games, California

Grace, L. 2014, “Critical Games: Critical Design in Independent Games” in *DiGRA Proceedings* 2014, Utah

Gursoy, A. 2013, *Game Worlds: A Study of Video Game Criticism*, Masters Thesis, MIT, Cambridge

Hamilton, R. 2014, *Perceptually Coherent Mapping Schemata for Virtual Space and Musical Method*, PhD Thesis, Stanford University

Hamilton, R. et al. 2011, “Multi-Modal Musical Environments for Mixed-Reality Performance” in *Journal on Multimodal User Interfaces*, Springer, Berlin, No. 4, pp. 147-156

Herber, N. 2007, “The Composition-Instrument: Musical Emergence and Interaction” in *Hz*, No. 9, Fylkingen, Stockholm, <http://www.hz-journal.org/n9/herber.html>, accessed 3/1/2015

Iwai, T. 2006, “Tenori-On” in *NIME 2006 Conference Proceedings*, Paris, <http://www.nime.org/proceedings/2006/nime2006_172.pdf>, accessed 5/8/2014

Jorda, S. 2003, “Interactive Music Systems for Everyone: Exploring Visual

Feedback as a Way for Creating More Intuitive, Efficient and Learnable Instruments" in *Proceedings of the Stockholm Music Acoustics Conference 2003*, Stockholm

Jorda, S. et al 2009, "The reactable: Tabletop Tangible Interfaces for Multithreaded Musical Performance", *Kepes Journal*, Vol 6, No. 5, pp. 201-223

Kayali, F. et al 2011, "Serious Beats: Transdisciplinary Research Methodologies for Designing and Evaluating a Socially Integrative Serious Music-Based Online Game" in *DiGRA Conference Proceedings 2011*

Magnusson, T. 2005, "ixi software: The Interface as Instrument" in *Proceedings of the 2005 International Conference on New Interfaces for Musical Expression*, Vancouver

McConnell, P. & Land, M. 1994, *Method and Apparatus for Dynamically Composing Music and Sound Effects Using a Computer Entertainment System*, patent, Lucasarts Entertainment Company, California, <http://www.google.com/patents/US5315057>, accessed 13/2/2016

McCormack, J. et al 2007, *Generative Composition with Nodal*, research paper, Centre for Electronic Media Art, Monash University

McCormack, J. & Mcilwain, P. 2005, "Design Issues in Music Composition Networks" in *Proceedings of Australasian Computer Music Conference 2005,* pp. 96-101

Mitgutsch, K. & Weise, M. 2011, "Subversive Game Design for Recursive Learning" in *DIGRA 2011 Proceedings*, Utrecht

Oldenburg, A. 2013, "Sonic Mechanics: Audio as Gameplay" in *The International Journal of Computer Game Research*, Vol. 13, Issue 1, <www.gamestudies.org>, accessed 1/5/2014

Pichlmair, M. & Kayali, F. 2007, "Levels of Sound: On the Principles of Interactivity in Music Video Games" in *Situated Play: Proceedings of DiGRA 2007 Conference*, Tokyo

Rafinski, A. & Zielke, M. 2013, "Defragging the Magic Circle: From Experience Design to Reality Design" in *DiGRA 2013 Proceedings: DeFragging Game Studies*

Ramirez, F. 2015, "Playing for the Plot: Blindness, Agency, and the Appeal of Narrative Organisation in Heavy Rain, in *Well Played*, Volume 4, No. 1, ETC Press, Pittsburgh, pp. 51-70

Ramsay, D. & Paradiso, J. 2015, “GroupLoop: A Collaborative, Network-Enabled Audio Feedback Instrument” in *NIME 2015 Proceedings*, Louisiana State University

Schacher, J. & Neff, P. 2015, “The Fluid and the Crystalline: Processing of the Music Performing and Perceiving Body” in *Proceedings of The 11th International Symposium on Computer Music Multidisciplinary Research (CMMR)*, pp. 16-19, June 2015, Plymouth

Sidhu, S. 2013, *Poetics of the Videogame Setpiece*, Masters Thesis, MIT, Cambridge

Smith, G. 2015, “Spore’s Playable Procedural Content Generation” in *Well Played*, Volume 4, No. 1, ETC Press, Pittsburgh, pp. 89-105

Stenros, J. 2014, “In Defence of a Magic Circle: The Social, Mental and Cultural Boundaries of Play”, *Transactions of Digital Games Research Association,* DiGRA, Vol. 1, No 2, pp. 147-185

Tannenbaum, J. & Bizzocchi, J. 2009, "Rock Band: A Case Study in the Design of Embodied Interface Experience" in *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games,* pp. 127-134

Verneau, L. 2013, *Paralect: An Example of Transition Focused Design*, Master’s Thesis, University of Southern California, California

Wang, G. 2015, “Game Design for Expressive Mobile Music” in *NIME 2015 Proceedings*, Louisiana State University

Westecott, E. 2009, “The Player Character as Performing Object” in *Proceedings of the 2009 DiGRA Conference*, London

Yang, Q. & Essl, G. 2015, “Representation-Plurality in Multi-Touch Mobile Visual Programming for Music” in *NIME 2015 Proceedings*, Louisiana State University

Games, Instruments and Interactive Media

Carlsen, J. 2013, *140*, video game, independent release, <http://game140.com/>, accessed 17/4/2014

Cavia, 2008, *Korg DS-10*, video game instrument, Nintendo Australia, Victoria

Conway, J. 1970, *Game of Life*, cellular automaton, Liverpool

Elliot, J. 2010, *A House In California*, video game, Cardboard Computer, Chicago, <http://cardboardcomputer.com/games/a-house-in-california/>, accessed 12/4/2014

Fischer, R. 2015, *TouchOSC*, mobile audio control application, Hexler, Karlsfeld, <http://hexler.net/software/touchosc>, accessed 12/4/2015

Flanagan, R. 2014, *Fract OSC*, video game, Phosfiend Systems, Montreal, <http://fractgame.com/>, accessed 28/4/2014

Furukawa K. et al, 1999, *Small Fish*, multimedia instrument/score application DVD, ZKM Centre for Art and Media, Karlsruhe

Gabriel, P. 1996, *Eve*, video game, Real World Multimedia, London

Gilbert, R. 1987, *Maniac Mansion*, video game, Lucasfilm Games, San Francisco

Griffiths, R. 2015, *Glitchspace*, video game, Space Budgie, Dundee

Hines, A. 2016, *Oxenfree,* video game, Night School Studio, California

Iwai, T. 1987, *Otocky*, video game, ASCII Corporation, Tokyo

Iwai, T. 1996, *SimTunes*, video game, Maxis, California

Iwai, T. 2005, *Electroplankton*, video game, Nintendo, Kyoto

Kay, R. et al, 2005, *Guitar Hero*, video game, Red Octane, California

Kay, R. et al, 2007, *Rock Band*, video game, Electronic Arts, California

Key, E. 2013, *Proteus*, video game, independent release, <http://www.visitproteus.com/>, accessed 11/4/2014

Klimus, C. 2015, *Twine*, interactive non-linear authoring tool, independent release, <http://twinery.org/>, accessed 14/5/2015

Martin, A. 2013, *Starseed Pilgrim*, video game, independent release, <http://www.starseedpilgrim.com/>, accessed 11/4/2014

Matsuura, M. 1996, *PaRappa the Rapper*, video game, Sony Computer Entertainment, Tokyo

Matsuura, M. 1999, *UmJammer Lammy*, video game, Sony Computer Entertainment, Tokyo

Matsuura, M. 1999, *Vib Ribbon*, video game, Sony Computer Entertainment, Tokyo

Miyamoto, S. 1998, *The Legend of Zelda: Ocarina of Time,* video game, Nintendo, Kyoto

Miyamoto, S. 2002, *The Legend of Zelda: The Wind Waker,* video game, Nintendo, Kyoto

Miyamoto, S. 2006, *The Legend of Zelda: Twilight Princess,* video game, Nintendo, Kyoto

Mizuguchi, T. 2001, *Rez*, video game, Sega, Tokyo

Mizuguchi, T. 2011, *Child of Eden*, video game, Ubisoft, Montreal

Moriarty, B. 1990, *Loom*, video game, Lucasfilm Games, San Francisco

O'Brien, C. 2014, *Tiber Synth*, web-based experimental synthesiser, independent release, <http://tibersynth.prtcl.cc/>, accessed 14/5/2015

Oliver, J. 1999, *Quilted Thought Organ*, game-based performance environment, independent project, http://julianoliver.com/output/qthoth, accessed 21/9/2014

Oliver, J. 2003, *q3apd*, game-based auto-performance environment and tool, independent project, <http://julianoliver.com/output/q3apd>, accessed 21/9/2014

Oliver, J. 2009, *Fijuu*, game-based interactive audio installation, independent project, <http://www.fijuu.com/>, accessed 5/8/2014

Phillips, L. 2016, *Blokdust*, browser-based sound-creation utility, <https://blokdust.com/>, accessed 26/8/2016, Brighton

Zolotov, A. 2014, *SunVox*, audio application, independent release, Ekaterinberg

Game Development Software and Systems

Gilbert, R. & Wilmunder, A. 1987, *Script Creation Utility for Maniac Mansion (SCUMM)*, point and click adventure game engine, Lucasfilm Games, California

Unity Technologies, 2016, *Unity*, game development environment, Unity Technologies, San Francisco

Lectures and Documentaries

Anderson, M. 2016, “Designing the Audio for Inside: A Game That Listens” at *gamasutra.com*, <http://www.gamasutra.com/view/news/277712/Video_Designing_the_audio_for_Inside_a_game_that_listens.php>, accessed 24/7/2016

Antonio, L. 2014, “The Art of The Witness” at *GDCvault.com*, <http://www.gdcvault.com/play/1020552/The-Art-of-The>, accessed 19/5/2015

Bahrami, M. et al, 2016, “Experimental Gameplay Workshop” at *Game Developer’s Conference 2016*, lecture, San Francisco

Blow, J. 2007, “Jonathan Blow: Design Reboot” at *youtube.com*, <https://www.youtube.com/watch?v=K0kup_anLeU>, accessed 4/8/2015

Blow, J. 2007, “Jonathan Blow: How and Why” at *youtube.com,* <https://www.youtube.com/watch?v=RsT-5VSqk8I>*,* accessed 4/8/2015

Blow, J. 2008, “Fundamental Conflicts in Contemporary Game Design” at *youtube.com,* <https://www.youtube.com/watch?v=mGTV8qLbBWE>*,* accessed 4/7/2015

Blow, J. 2010, “Video Games and the Human Condition” at *youtube.com*, <https://www.youtube.com/watch?v=SqFu5O-oPmU>, accessed 10/7/2015

Blow, J. 2013, “The Witness Walkthrough with Jonathan Blow Developer Exclusive Gameplay PS4 & PC” at *youtube.com,* <https://www.youtube.com/watch?v=rDSrYiheVow>, accessed 29/12/2014

Blow, J. 2016, “Storytime with Jonathan Blow at PAX East 2016” at *youtube.com*, <https://www.youtube.com/watch?v=UwBl7Rnkt78>, accessed 17/5/2016

Cardwell-Gardner, P. 2014, “Cadence Dev Chronicles: Part 2” at *youtube.com*, <https://www.youtube.com/watch?v=h8hunxj83Xw>, accessed 4/7/2015

Cardwell-Gardner, P. 2014, “Cadence Dev Chronicles 3: Parallel Prototyping” at *youtube.com*, <https://www.youtube.com/watch?v=W1Hw1uhEwOw>, accessed 4/7/2015

Crawford, J. 2014, “Preserving a Sense of Discovery in the Age of Spoilers” at *vimeo.com*, <https://vimeo.com/91436410>, accessed 27/4/2015

Elliott, J. 2014, “Designing for Mystery in Kentucky Route Zero” at *gamasutra.com*, <http://www.gamasutra.com/view/news/220199/Video_Designing_for_mystery_in_Kentucky_Route_Zero.php>, accessed 20/7/2016

Fullerton, T. 2012, “Walden: The Video Game" at *youtube.com*, <http://www.youtube.com/watch?v=TaZ9BiSJ-KI>, accessed 11/4/2014

Gilbert, R. 2011, "Maniac Mansion Post Mortem", lecture given at Game Forum Germany*, at youtube.com,* <http://www.youtube.com/watch?v=wNpjGvJwyL8>, accessed 27/4/2014

Graves, J. et al, 2016, “Storytelling Tools: Using Music & Sound Creatively” at *Game Developer’s Conference 2016*, lecture, San Francisco

Haggis, M. 2016, “Writing ‘Nothing’: Storytelling with Unsaid Words and Unreliable Narrators” at *Game Developer’s Conference 2016*, lecture, San Francisco

Johnson, M. 2015, “Handmade Detail in a Procedural World” at *gdcvault.com*, <http://www.gdcvault.com/play/1022751/Handmade-Detail-in-a-Procedural>, accessed 15/12/2015

Juul, J. 2010, “The Pure Game: A Short History of Video Game Aesthetics” at *youtube.com*, <https://www.youtube.com/watch?v=NwAuHhpbgfY>, accessed 1/1/2016

Juul, J. et al 2010, “Art History of Games: Panel Discussion with Jesper Juul, Frank Lantz and John Sharp” at *youtube.com,* <https://www.youtube.com/watch?v=dbqvU_Q66i4>, accessed 2/10/2015

Key, E. 2012, “Abstraction and Experience: Observations on Proteus” at *gdcvault.com*, <http://www.gdcvault.com/play/1016480/Abstraction-and-Experience-Observations-on>, accessed 15/12/2015

Kipnis, A. 2015, “Practice 2015: Considerations for Expressive Simulation” at *vimeo.com*, <https://vimeo.com/149287018>, accessed 8/2/2016

Lowood, H. 2010, “Players Are Artists Too” at *youtube.com*, <https://www.youtube.com/watch?v=PVnUDWhCEpE>, accessed 18/7/2015

Moriarty, B. 2016, “Jonathan Blow Interview @ WPI” at *youtube.com,* <https://www.youtube.com/watch?v=tRHnHlV96Jg>*,* accessed 17/5/2016

Murray, J. 2010, “NYU Game Center Lecture Series: Janet Murray (Parts 1–6)” at *youtube.com*, <https://www.youtube.com/watch?v=pPV1wMVGPDM>, accessed 3/1/2016

Nesky, J. 2015, “50 Common Game Camera Mistakes and How to Fix Them”, at *gamasutra.com,* <http://gamasutra.com/view/news/259610/Video_50_common_game_camera_mistakes__and_how_to_fix_them.php>, accessed 28/11/2015

Owens, P. 2014, *Double Fine Adventure*, video documentary series, downloadable distribution, 2 Player Productions, Portland

Pearce, C. 2010, “Play’s the Thing: Games as Fine Art” at *youtube.com*, <https://www.youtube.com/watch?v=3x7GTjQFT18>, accessed 31/12/2015

Phillipps, C. 2016, “All Choice No Consequence: Efficiently Branching Narrative” at *Game Developer’s Conference 2016*, lecture, San Francisco

Platz, C. 2014, “ECHO::Canyon 2014” at *youtube.com*, <https://www.youtube.com/watch?v=enGvXaNLbo8>, accessed 16/12/2014

Robinson, E. 2016, “Punching Up the Juice with Proactive Audio” at *Game Developer’s Conference 2016*, lecture, San Francisco

Rohrer, J. et al, 2010, “Artgame Sessions”, at *gdcvault.com*, <http://www.gdcvault.com/play/1012249/Artgame>, accessed 1/8/2014

Rohrmann, A. 2015, “Creating Hyper-Adaptive Game Music on an Indie Budget” at *gdcvault.com*, <http://www.gdcvault.com/play/1021886/Creating-Hyper-Adaptive-Music-on>, accessed 15/12/2015

Sessler, A. 2013, “The Witness: Jonathan Blow on His PS4 Open-World Puzzle

Game – Adam Sessler" at *youtube.com*, <https://www.youtube.com/watch?v=16wLW9hJTkg>, accessed 1/8/2014

Sharp, J. 2010, "The Art History of Games" at *youtube.com*, <https://www.youtube.com/watch?v=p4m3VI1DexA>, accessed 31/12/2015

Smith, R. 2016, "Advanced Environmental Storytelling in 'Spider: Rite of the Shrouded Moon' " at *Game Developer's Conference 2016*, lecture, San Francisco

Swink, S. 2007, "IGS 2007: Innovation in Indie Games w Swink, Gabler, Chen, Mak, Blow" at *youtube.com*, <https://www.youtube.com/watch?v=PR-ZtrOGHiY>, accessed 4/8/2015

Ten Bosch, M. 2015, "Practice 2015: Marc Ten Bosch: Exploring and Presenting a Game's Consequence Space" at *vimeo.com*, <https://vimeo.com/150690511>, accessed 18/1/2016

USC School of Cinematic Arts, 2014, "Leviathan Featured at CES" at *USC School of Cinematic Arts Website,* <http://thecreatorsproject.vice.com/blog/leviathan-the-future-of-storytelling>, accessed 11/4/2014

Victor, B. 2012, "Inventing On Principle" at *vimeo.com*, <https://vimeo.com/36579366>, accessed 16/12/2015

Victor, B. 2012, "Stop Drawing Dead Fish" at *vimeo.com,* <https://vimeo.com/64895205>, accessed 9/7/2015

Victor, B. 2013, "Drawing Dynamic Visualizations" at *vimeo.com*, <https://vimeo.com/66085662>, accessed 18/12/2015

Victor, B. 2013, "The Future of Programming" at *vimeo.com*, <https://vimeo.com/71278954>, accessed 19/12/2015

Victor, B. 2014, "Seeing Spaces" at *vimeo.com*, <https://vimeo.com/97903574>, accessed 19/12/2015

Victor, B. 2014, "The Humane Representation of Thought" at *vimeo.com*, <https://vimeo.com/115154289>, accessed 20/12/2015

Weinberg, J. 2016, "Journey Into the DAG: Puzzle Dependency Charts, Tentacles and You" at *Game Developer's Conference 2016*, lecture, San Francisco

Zimmerman, E. 2016, “How I Teach Game Design: The User-Customizable Game” at *Game Developer’s Conference 2016*, lecture, San Francisco

Music Software

Ableton, 2016, *Live*, digital audio workstation, Berlin

Bencina, R. 2016, *AudioMulch*, modular audio software, Sonic Fritter, Melbourne

Carlson, C. 2011, *Borderlands*, software instrument, <https://ccrma.stanford.edu/~carlsonc/256a/Borderlands/index.html#Downloads>, accessed 27/8/2014
(see also iPad version (2014) <http://www.borderlands-granular.com/app/>, accessed 27/8/2014)

Cycling ’74, *Max*, music and multimedia development environment, Cycling ’74, San Francisco

McCormack, J. 2015, *Nodal*, generative music software, Monash University, Melbourne

Unity Code Libraries

Audiokinetic, 2016, *Wwise*, audio middleware for Unity, Audiokinetic, Montreal

Hourdel, T. 2015, “Colorful FX: Post-Processing FX for Unity 3D” at *thomashourdel.com*, <http://www.thomashourdel.com/colorful/>, accessed 13/2/2017

Ledvina, M. 2014, “MIDI Unified 3.0” at *assetstore.unity3d.com*, <https://www.assetstore.unity3d.com/en/#!/content/576>, accessed 13/2/2017

Pennington, J. 2014, “Audial Manipulators” at *assetstore.unity3d.com*, <https://www.assetstore.unity3d.com/en/#!/content/17032>, accessed 13/2/2017

Schlupek, S. 2014, “UniOSC” at *uniosc.monoflow.org*, <http://uniosc.monoflow.org/>, accessed 13/2/2017

Starscene Software, 2014, “UniFileBrowser” at *assetstore.unity3d.com*, <https://www.assetstore.unity3d.com/en/#!/content/151>, accessed 13/2/2017

Zanon, G. 2016, “G-Audio: Dynamic Audio for Unity” at *g-audio-unity.com*, <http://www.g-audio-unity.com/>, accessed 13/2/2017

Web Pages and Articles

Alexander, L. 2012, “How XCOM Enables Players to Tell Their Own Stories”, at *gamasutra.com*, <http://www.gamasutra.com/view/news/178600/How_XCOM_enables_players_to_tell_their_own_stories.php>, accessed 2/6/2015

Antonio, L. 2016, “Trusting the Player and Trusting Your Game” at *twelveminutesgame.com*, <http://twelveminutesgame.com/2016/06/trusting-the-player-trusting-your-game/>, accessed 22/7/2016

Ashwell, S. 2015, “Standard Patterns in Choice-Based Games” at *heterogenoustasks.wordpress.com*, <https://heterogenoustasks.wordpress.com/2015/01/26/standard-patterns-in-choice-based-games/>, accessed 2/12/2016

Bevan, M. 2013, “The SCUMM Diary: Stories Behind One of the Greatest Game Engines Ever Made” at *gamasutra.com,* <http://www.gamasutra.com/view/feature/196009/the_scumm_diary_stories_behind_.php>, accessed 2/2/2017

Boom, H. 2013, “FRACT Audio Tech: Getting Started” at *fractgame.com,* <http://fractgame.com/news/126-fract-audio-tech-getting-started>, accessed 31/1/2017

Boom, H. 2013, “FRACT Audio Tech: Connecting to Pure Data” at *fractgame.com,* <http://fractgame.com/news/127-fract-audio-tech-connecting-to-pure-data>, accessed 31/1/2017

Boom, H. 2013, “FRACT Audio Tech: Wrapping Pure Data” at *fractgame.com,* <http://fractgame.com/news/130-fract-audio-tech-wrapping-pure-data>, accessed 31/1/2017

Boom, H. 2013, “FRACT Audio Tech: Message Basics” at *fractgame.com,* <http://fractgame.com/news/131-fract-audio-tech-message-basics>, accessed 31/1/2017

Cannon, B. 2015, “Radio Days: Inside the Oxenfree Radio” at *nightschoolstudio.com*, <http://nightschoolstudio.com/radio-days-inside-the-oxenfree-radio/>, accessed 17/12/2015

Cifaldi, F. 2013, “The Technique Lucasarts Used to Design Its Classic Adventure Games” at *gamasutra.com,* <http://www.gamasutra.com/view/news/189266/The_technique_LucasArts_used_to_design_its_classic_adventure_games.php>, accessed 21/4/2016

Clark, T. 2012, “The Most Dangerous Gamer” at *The Atlantic*, <http://www.theatlantic.com/magazine/archive/2012/05/the-most-dangerous-gamer/308928/>, accessed 25/5/2015

Cook, D. 2007, “The Chemistry of Game Design” at *gamasutra.com,* <http://www.gamasutra.com/view/feature/129948/the_chemistry_of_game_design.php>, accessed 2/8/2015

Cook, D. 2009, “Three False Constraints” at *lostgarden.com*, <http://www.lostgarden.com/2009/11/three-false-constraints_29.html>, accessed 5/7/2015

Einhorn, A. 2015, “What, Not How: A Goal Centered Approach to Player Motivation” at *gamasutra.com,*<http://www.gamasutra.com/blogs/AsherEinhorn/20150529/244602/quotWhat_not_Howquot__A_goal_centred_approach_to_player_motivation.php>, accessed 5/7/2015

Felder, D. 2015, “Design 101: Complexity vs. Depth” at *gamasutra.com,* <http://gamasutra.com/blogs/DanFelder/20150521/243962/Design_101_Complexity_vs_Depth.php>, accessed 5/7/2015

Gilbert, R. 2014, “Puzzle Dependency Charts” at *grumpygamer.com*, <http://grumpygamer.com/puzzle_dependency_charts>, accessed 21/4/2015

Gilbert, R. 2015, “Dialog Puzzles” at *blog.thimbleweedpark.com*, <http://blog.thimbleweedpark.com/dialog_puzzles>, accessed 1/7/2015

Hanson, B. 2014, “Creating No Man’s Sky’s Infinite Soundtrack” at *gameinformer.com,* <http://www.gameinformer.com/b/features/archive/2014/12/22/creating-the-infinite-soundtrack-of-no-man-s-sky.aspx>, accessed 9/1/2015

Harris, D. 2013, “The Suite Science: Paul Weir Talks Generative Music” at *Rock, Paper, Shotgun*, <http://www.rockpapershotgun.com/2013/11/20/the-suite-science-paul-weir-talks-generative-music/>, accessed 5/12/2015

Karras, D. 2014, “Sound Synthesis Theory / Oscillators and Wavetables” at *wikibooks.org*, <http://en.wikibooks.org/wiki/Sound_Synthesis_Theory/Oscillators_and_Wavetables>, accessed 5/6/2015

Kukshtel, K. 2014, “The Clash Between Systems and Narrative Comes to a Head in The Hit” at *killscreendaily.com,* <http://killscreendaily.com/articles/clash-between-systems-and-narrative-comes-head-hit/>, accessed 9/8/2015

La Burthe, A. & Hen, D. 2012, “Procedural Audio with Unity” at *develop-online.net*, <http://www.develop-online.net/tools-and-tech/procedural-audio-with-unity/0117433>, accessed 5/6/2015

Maher, J. 2017, “Loom (or How Brian Moriarty Proved That Less is Sometimes More)” at *filfre.net*, <http://www.filfre.net/2017/02/loom-or-how-brian-moriarty-proved-that-less-is-sometimes-more/>, accessed 20/2/2017

Moriarty, B. 2015, “I Sing the Story Electric” at *ludix.com*, <http://ludix.com/moriarty/electric.html>, accessed 16/12/2015

Moriarty, B. 2015, “No, Ask Me About Loom: Classic Game Postmortem: LucasFilm Games’ Loom” at *ludix.com*, <http://ludix.com/moriarty/loom.html>, accessed 28/7/2015

Remo, C. 2008, “The Last Express: Revisiting an Unsung Classic” at *gamasutra.com*, <http://www.gamasutra.com/view/feature/3862/the_last_express_revisiting_an_.php>, accessed 11/9/2016

Still Eating Oranges, 2012, "Hi Your Recent Post About Conflict..." at *stilleatingoranges.tumblr.com*, <http://stilleatingoranges.tumblr.com/post/25248152114/hi-your-recent-post-about-conflict-was-an-interesting>, accessed 6/5/2014

Still Eating Oranges, 2012, "Mark Essen's Worlds Without Maps" at *stilleatingoranges.tumblr.com*, <http://stilleatingoranges.tumblr.com/post/25789094763/mark-essens-worlds-without-maps>, accessed 6/5/2014

Still Eating Oranges, 2012, "The Significance of Plot Without Conflict" at *stilleatingoranges.tumblr.com*, <http://stilleatingoranges.tumblr.com/post/25153960313/the-significance-of-plot-without-conflict>, accessed 6/5/2014

Still Eating Oranges, 2013, "Plot Structure All the Way Down" at *stilleatingoranges.tumblr.com*, <http://stilleatingoranges.tumblr.com/post/53045164430/plot-structure-all-the-way-down>, accessed 6/5/2014

Still Eating Oranges, 2014, "Kishotenketsu in Mario" at *stilleatingoranges.tumblr.com*, <http://stilleatingoranges.tumblr.com/post/76178051254/kishotenketsu-in-mario>, accessed 6/5/2014

Stuart, K. 2015, “Everybody’s Gone to the Rapture: Writing a Score for the End of the World” at *thegaurdian.com,* <http://www.theguardian.com/technology/2015/jul/30/everybodys-gone-to-the-rapture-video-game-sound-music>, accessed 9/8/2015

Stubbs, D. 2014, “Dynamic Narrative in The Hit” at *gamasutra.com*, <http://www.gamasutra.com/blogs/DanStubbs/20140402/214565/Dynamic_Narrative_in_The_Hit.php>, accessed 9/8/2015

Weinberg, J. 2016, “Game Developer’s Conference: Day of the Tentacle Puzzle Dependency Graph” at *thewebsiteisdown.com*, <http://thewebsiteisdown.com/twidblog/?p=947>, accessed 17/3/2016

Part 2

Appendices

Table of Contents

Appendix A: Diagrams of Game Object Functionality

See attached USB Drive.

Diagrams Table of Contents

Appendix B: Performance Manual

Also available as a separate document on attached USB Drive.

Note that there is no manual for Story Mode, however an in-game tutorial is available and is highly recommended before proceeding. From the title screen select “Story > New Game > Tutorial”

Appendix C: Description of External Code Libraries Used in Unity

The following code-libraries were used to augment Unity's capabilities in the manner listed below:

- G-Audio (Zanon, 2016): in *Pallas*' Performance Mode, G-Audio completely replaces Unity's default audio system, allowing for many otherwise unachievable functionalities such as real-time routing of audio (including real-time uninterrupted instantiation of DSP effects chains), real-time instantiation of audio channels, pulse-based envelope triggering of samples, smooth-continuous pitch-shifting and reverse sample playback

- Audial (Pennington, 2014): a collection of audio DSP effects that can be implemented in effects chains within G-Audio's channel system

- Wwise (Audiokinetic, 2016): used in *Pallas*' Story Mode to support real-time interpolation between spatially distributed synchronised music sources, as well as expressive musical parameter manipulation in puzzles

- Colorful (Hourdel, 2015): a collection of camera filter processing effects

- MIDI Unified (Ledvina, 2014): basic MIDI input/output library. Used here only to specify connections to MIDI input/output devices, and to retrieve incoming Note and Continuous Control data

- UniOSC (Schlupek, 2014): implements Open Sound Control connections and real-time messaging

- UniFileBrowser (Starscene Software, 2014): an in-game file-browser. Allows players to access their local hard-drive to load/save files: for example to load audio samples into a performance

Appendix D: Gameplay Videos:

See attached USB Drive.

Performance Mode:

Before watching please note the following:

- These videos are unedited: they each demonstrate unbroken passages of real-time gameplay dynamics.

- The intention of these videos is to demonstrate specific functionalities, as described in their titles 1-12 below. As such they should be considered instructional, and not taken as cohesive performances or polished audio-visual pieces in their own right. The below commentaries are focused purely on functionality and not on aesthetic outcome, and it is expected that the reader (via the entire contents of this thesis) build up an understanding of the many and varied ways a cohesive performance could be executed.

- These videos should be understood as a set where each will help to make sense of the others, and are intended to be watched in order. As such each subsequent commentary below omits any details that have been explained in a previous video's commentary. The commentaries should also be cross-referenced with the Diagrams in Appendix A and the Manual in Appendix B.

- The content of these videos does not represent an exhaustive demonstration of all the objects and functionalities available in framework. The Manual in Appendix B (in its entirety) should be considered the most complete list of functionalities.

1) Overview Example: Avatars, Audio and Visual Object Basics
2) Multiple Avatars, Multiple Sound Sources and Audio Busses
3) Sample Players, Pulses, Effects Chains and User Sample Pools
4) Synthesis, Arpeggiators and Synth LFOs

5) Pitch Objects, Note Sequencers and Sample Loopers
6) Lighting, Colouration and Links Between Audio and Visual Parameters
7) Camera Filters (this video intentionally has no audio)
8) Switches, LFOs, X/Y Panels, Parameter Store/Recall
9) Cyclical and Physics-Based Motion: Spinners, Orbits, Anchor-Motion
10) Hardware Mapping: MIDI Note and CC, Gamepad (Examples 1, 2 & 3)
11) Node Sequences
12) Basic Generative Structures (Examples 1 & 2)

A brief commentary on each of the above videos follows:

1) Overview Example: Avatars, Audio and Visual Object Basics

This video shows a range of the most common audio and visual functionalities. The player extracts a sample from a Sample Store object, thus creating a Sample Player sound source. The field around this sound is adjusted, thus affecting it listener-radius.
The player then creates a Delay effect (the blue cylinder), and uses the cursor to make a connection from the sound source to the effect. The field around the effect is adjusted, which changes how much of that effect is applied to the sound.
A second Avatar is created, which is then used to pick up and move the Delay effect. Moving it away from the sound source reduces the amount of Delay applied to that sound.
As the first Avatar has been automatically assigned as the listener of the sound source, the movement of the second avatar does not affect the volume or panning of that sound.
An LFO effect (the pink/red cylinder) is connected to the same Sound Source's effects chain, after the Delay effect and its field is adjusted with a more audible result. The second Avatar is connected to a fader on the LFO effect, such that the proximity of the avatar to that fader (which in this case represents LFO-rate) will determine its value.

A Global Light Source is created, as well as a standalone fader. 2 parameters of the light source and one parameter of the LFO are connected to the fader, so that the one fader controls all 3 parameters at once. Then the avatar is connected to that fader so that its proximity will control all 3 parameters. The light source is destroyed, and a Parameter LFO (update-rate) is created, which is used to oscillate the value of the LFO effect (audio rate).
A Pulse object and a new Sound Source are created. The Pulse is connected to the Sound Source and the player creates a new Avatar, connects the sound to the same Delay effect as the previous sound, and uses the cursor to move the playhead along the sound's waveform to determine which portion of the sample the Pulse will trigger.
A Camera Filter object is added, and one filter is created on it and manipulated, later followed by a second filter. The connected Avatar moves around the Camera Filter object to demonstrate the proximity relationship.
The player connects the Pulse to the original Sound Source and manipulates its playhead as previously described.
The player selects all the fields in the scene and reduces their respective radii simultaneously, thus fading out all sounds and all audio and visual effects.

2) Multiple Avatars, Multiple Sound Sources and Audio Busses

Many of the same principles from the previous video are applied in setting up the sounds in this scene. The second sound is created by duplicating the first. On the second Sample Player the bottom glyph is clicked (where a white dot appears), meaning that this sound is now playing in reverse.
Avatar proximity and rotation are used to demonstrate their effects on volume and panning on the second Sample Player, while the first sound (connected to the first avatar) remains stable in its volume and panning as that Avatar is not moving.
An Audio Bus is created and both sounds are connected. The sounds maintain their relative volume and panning (based on their individual avatar-listener positions), and a secondary layer of volume and panning alteration is applied via the third Avatar's position and rotation relative to the Audio Bus object. The

player toggles the appearance of the connection lines to illustrate where both the sound-to-avatar and sound-to-bus connections are taking effect.
By destroying the Audio Bus object, its mediation of the volume and panning of the sounds is immediately nullified, thus immediately returning the sounds to a direct relationship with their avatars.
Two new sounds are created from a second sample, and a single Avatar is connected as the listener of all 3 of the sounds in the scene. Thus at that point, the movements of a single Avatar are used to mix the volume and panning of all three sounds at once.

3) Sample Players, Pulses, Effects Chains and User Sample Pools
(Note there is a small audio glitch at around 2:30)

A User Sample Pool object is created and is set to “Auto add any new samples” meaning that if the player goes through the usual process of creating a Sample Store via a WAV file retrieved from their hard-drive, this WAV file will be store in the User Sample Pool instead (as shown by a golden sphere for each sample).
A Sound Source is then created and controlled by a Pulse as previously described. A duplicate of the sound is then also controlled by the same pulse and 2 effects are applied to it.
A second Pulse object is created and put into “Loop Sync Mode”, shown by its cylinder coloured red. Loop Sync Mode outputs pulses at a much higher rate, and is used to drive the position of the playhead (either forwards or in reverse) on any connected Sound Source.
The player creates a new Sound Source by dragging a sample from the User Sample Pool, and then connects a Delay effect. The player controls the final sustain and decay of this sound by setting a high feedback value on the Delay before moving the avatar-listener away.

4) Synthesis, Arpeggiators and Synth LFOs

The player creates a Synthesizer object, which by default is in monophonic mode. They cycle through some of the available waveform-types (Square,

Triangle, Sine, etc.), then manipulate FM rate and depth parameters. A second Avatar is then connected to the FM rate dial and controls it via proximity.
A Synth LFO object is created. This is conceptually the same as the Parameter LFO discussed in Video 1, but it is specifically designed to manipulate synth parameters at audio rate. While it is only connected to one parameter here, note that it could be connected to any number of compatible parameters on any number of Synth objects simultaneously. Synth LFOs can also be used to control other Synth LFOs.
It is connected to the FM depth dial, then the player adjusts its parameters and cycles through some of its waveform-types (Square, Triangle, Sine, etc.). The player enables note-input on the Synth (shown by the small gold sphere above it) and inputs notes via the QWERTY keyboard. The Synth is then put into Polyphonic Mode, and its ADSR amplitude envelope is adjusted.
A Pulse is connected to the Synth, automatically setting it to Arpeggiator Mode. The player continues to input notes which are thus arpeggiated by the Pulse. A set of 3 notes are locked into the arpeggiator (shown by the two gold bars above the synth).
A second Synth object is created, and set up to be arpeggiated by the same Pulse. The AM parameters on this second synth are manipulated, then a Phaser effect is connected and both the AM and Phaser rates are manipulated via the proximity of a third Avatar. An LPF effect is connected and manipulated. On the first Synth the player inputs notes while the Arpeggiator is locked on, thus adding new notes to its cycle. A Parameter LFO is created and used to oscillate the LPF frequency.
The first Synth is faded out via Avatar proximity and the second Synth is faded out and FM manipulated via Avatar proximity, while the Pulse is slowed.

5) Pitch Objects, Note Sequencers and Sample Loopers

Two Sample Players and a Pitch object are created. The Pitch object is connected to both, along with a Delay effect. Both samples are played and pitch-manipulated.
The Pitch object is put into Record-Player mode and the player uses scroll input (in this case trackpad-scrolling) to manipulate the pitch-wheel. The reverse-

glyph is toggled thus controlling reverse playback of both samples. The Pitch object's on-screen keyboard is revealed and used to input specific pitches to the Sample Player.
The Sample Player is targeted for Loop Recording (shown by a red sphere above it), then the player begins recording a loop (shown by a red sphere above the Avatar). A new Sample Player containing the recorded loop is created automatically as soon as the player completes the loop recording. In the same way a second loop is then recorded from a new Sample Player.
The player then initiates a Master Loop Recording. This is done by not targeting a specific sound source to record, thus defaulting to record the Master Output Channel (i.e. the aggregate of all audible sounds in the scene). The player then sets the resulting loop to play in reverse.
A Synth and a Note Sequencer are created and connected. The Note Sequencer is loop-record enabled (shown by the flashing red/white cube), and the player inputs a sequence of notes (in this case via the QWERTY keyboard). As the Note Sequencer loops, the player adds additional notes to the sequence. The Synth is targeted for loop recording, resulting in a new Sample Player looping a portion of the Synth's recorded output, which is then played in reverse.
A new Master Loop Recording is made, capturing portions of both the Synth's output and the previously recorded sample-loop of the Synth, and this is used to then complete the piece.

6) Lighting, Colouration and Links Between Audio and Visual Parameters

Note: the first half of this video intentionally has no audio

A Global Lighting object is created, and its sunlight and ambient light parameters are decreased, darkening the scene. The player moves the connected Avatar to show how proximity affects the lighting. The Global Lighting object is toggled on and off.
Two Local Lighting objects are created and altered. Other Global Light parameters such as light-angle (affecting the shadows in the scene), fog density

and fog colour are altered, and a single Fader is then used to control several parameters at once.
A Colouration object is created, connected to two Local Lights, and thus used to alter their colour.
A Sample Player is connected to a LPF effect and the LPF frequency is connected to the Fader that is controlling the Global and Local Light settings. This Fader is thus used to control both lighting and audio. A Parameter LFO is then used to oscillate that Fader, as well as the Colouration object (thus oscillating colours of the Local Lights).
The Global Light is eventually switched off, restoring default lighting to the scene.

7) Camera Filters

Note: this video intentionally has no audio

A Camera Filter object is created and a single filter slot is added. The player cycles through some of the available filters on that slot. The Camera Filter object is toggled on and off, and its field resized thus altering its Avatar proximity-relative intensity. A second slot is added and the player cycles through some of the available filters (note this works like an audio effects chain, where each subsequent filter is processed in a chain after any that precede it).
A second Camera Filter is created and proximity relationships are explored. The player then alters parameters via dials on individual filters, both via cursor and Avatar proximity.
A Global Light is then manipulated to demonstrate the breadth of influence this has over the Camera Filters.

8) Switches, LFOs, X/Y Panels, Parameter Store/Recall

A Switch object is created and is left in its default proximity mode (shown by the continuously visible field surrounding it). A usual Sample Player plus effects setup is created, and the effects are connected to the Switch. When the Avatar exits or enters the Switch's field, the effects are toggled on or off (shown be

their upper golden-rings disappearing) and this bypasses them on the Sample Player.
A Parameter LFO is also connected to the Switch, and used to control parameters on both effects. The Phaser effect is disconnected from the Switch (shown by the white sphere exploding), so the toggling of the Parameter LFO by the Switch can be better demonstrated.
A Multi-Parameter Gesture Surface is created. This outputs the X and Y position of the inset gold-square relative to its distance from the bottom-left of the panel (i.e. where the bottom-left is X = 0, Y = 0). A Lofi effect is created (the yellow cylinder). A parameter of the Lofi effect is connected to Surface's X value, and a parameter of the Phaser effect is connected to the Surface's Y value. The Lofi effect's sample-rate parameter (controlled by the Surface's X) is then scaled down into a more audible range via the two scaling dials on the fader. A Global Lighting object and a new Sample Player are created and toggled via the Switch.
A Parameter Store object is created (the pyramid with two spheres) and two effects parameters are connected to it. The player creates to storage nodes on it, stores different parameter positions in each (shown by the node turning red), and then both smooth-interpolates and instant-switches between the stored values, also demonstrating variable interpolation speeds via the dial on each node. The Sample Player is then triggered by a Pulse in order to demonstrate the Parameter Store object's ability to store, retrieve and interpolate between playhead positions along the Sample Player's waveform.

9) Cyclical and Physics-Based Motion: Spinners, Orbits, Anchor-Motion

An Avatar listening to a Sample Player is placed on a Spinning Platform, thus both the volume and panning of the Sample Player are modulated according to the Avatar's rotational position. The platform's spin-rate is adjusted to make this affect clearer. A parameter of an LFO effect is connected to the Avatar on the platform, so that this parameter oscillates according to the Avatar's rotation. Both of these Avatar connection-lines are made visible in order to make these proximity relationships clearer.

A Phaser object (connected to the Sample Player) is placed on the Spinning Platform, thus oscillating the amount of the effect applied to the Sample Player via its rotation. The Phaser's rate parameter is then connected to the Avatar who is not on the platform, thus also oscillating that parameter.
Two Orbiter objects are created and one is set in motion around the other. The position of the outer-most Orbiter is used to control the filter-frequency of a LPF (this momentary cuts off the sound as the filter-frequency parameter maps to the Orbiter's rotation speed, which at that point is zero. This is resolved when the player switches the mapping to the Orbiter's proximity). A more complex orbital relationship is created whereby the outer-most Orbiter's position is determined by both its own around a parent-object, and that parent-object's orbit around a higher-level parent-object.
A second Spinning Platform is created and set to a different speed, and the Phaser is moved onto that platform. The LPF effect's filter-frequency parameter is attached to the relative Y-position of an Anchor and Motion object pair (this Y-position is the Y-axis distance between the Anchor and the Motion (where the Motion is the moving sphere). Using, at different times, the cursor and the Avatar, the player applies various forces to the physics-simulation-based Motion object which thus applies its height to the LPF filter-frequency as it rolls over the terrain, flies through the air, and so on. The LFO effect's frequency is also mapped to this height parameter (relative Y-position) of the Anchor Motion pair.

10) Hardware Mapping: MIDI Note and CC, Gamepad (examples 1, 2 and 3)

Note: There are 3 separate videos for Hardware Mapping:

10.1) Gamepad

The gamepad is essentially pre-mapped, and is demonstrated here in a local-multiplayer context (although it can also be used for single-player in which case the camera would be following the Avatar as usual). In this context the camera remains fixed on Player 1 (who in this case is QWERTY + cursor controlled, and who remains stationary in this video) and the gamepad player is free to move around the visible play-area. The gamepad player is attached as the listener to

the Sample Player, and can assign themselves as the listener to any available sound source at any time. The player can pick up and move objects, and toggle any switchable object on or off. They can map their controls to any fader or dial-based parameter by targeting it with the Left/Right shoulder buttons (controlled by the index fingers) and moving the right analog stick up and down (right thumb) to alter the parameter.
In a single player context (not shown here) the gamepad can also be used to control the cursor, and thus all the inter-object connectivity, field scaling, and on-screen button toggling controls.

10.2) MIDI CC (Continuous Control)

Two effects are connected to a Sample Player and each has a parameter mapped to a dial on the MIDI hardware device. The hardware device is simply selected from an in-game menu (not shown here). In order to make a MIDI CC mapping the player places the cursor over the desired parameter and holds F on the QWERTY keyboard, then moves the MIDI CC control on the hardware (here the dial) and releases the F key. Repeating this method, any parameter can be remapped to a different hardware control (e.g. another dial, fader, etc.). A mapping can be toggled on/off at any time by right-clicking the parameter. Any visual parameter, as shown here with the example of a Local Light object, can be mapped in exactly the same way as an audio parameter. Any number of parameters can be mapped to a single hardware control (e.g. a single hardware dial could control any number of in-game parameters simultaneously).

10.3) MIDI Note

A Synth object is created and set to Polyphonic Mode. The MIDI hardware device is simply selected from an in-game menu (not shown here). Then, in order to create a MIDI Note mapping, the player places the cursor over a compatible object (such as a Synth, Sample Player, Pitch object or Note Sequencer), holds Shift + F (showing a cone-shaped “ready” glyph) and presses any note on the MIDI keyboard. While a note mapping is active, the object will show a small white sphere above.

As shown, multiple objects can be mapped to the same hardware. A second Synth is created and mapped to the MIDI keyboard and is set to a different waveform-type to differentiate its sound.
A Pulse is used to arpeggiate the Synth, as previously described, and the viewer can observe how this changes the way that note-input is received from the MIDI device.

11) Node Sequences

Two Sample Players, a chain of Sequence Nodes, and a Pulse are created. The Pulse is used to drive the sequence. The Sample Players are connected to different nodes in the sequence and are set to envelope mode, meaning that the nodes will trigger an envelope of the samples, according to where their respective playheads are positioned.
A third and fourth Sample Player are created, as well as a Delay effect which is then connected to all Sample Players. The player then spends some time adjusting the playhead positions and envelope parameters for each sample while the sequence plays.
The sequence is slowed, some of the envelopes are extended in order to match the slower tempo, and the sequence is eventually stopped.

12) Basic Generative Structures (Examples 1 & 2)

12.1) Basic Generative Structures 1

A Synth arpeggiator setup is created, as previously described, and a second arpeggiating Synth is added. Two Spinning Platforms are created to move a Synth and an Avatar listener, and they are set in motion at different spin-speeds. A third arpeggiating Synth is added on its own Spinning Platform and this is connected to a Lofi effect. The first Synth is placed on a newly created Spinning Platform and set to spin, again at a different speed.
Two Sample Players are created and are driven by the same Pulse that drives the Synths. A new Pulse is created, set to Loop Sync mode and used to drive

the playhead position of one of the Sample Players. A Delay effect is connected to one of the Sample Players and the player spends some time configuring the delay-time parameter to suit the Pulse rate.
A Parameter LFO is used to control the speed of the Loop Sync pulse. A loop is recorded from the Sample Player, its playback pitch is altered, and its listener is placed on a small Spinning Platform. A Phaser effect is connected.
A Delay effect is added to one of the Synths.
The recorded loop is destroyed, along with its Avatar and Spinner. The Phaser is connected to the Sample Player along with a LPF, which is set to oscillate via the Parameter LFO.
The Synths are gradually destroyed, then the Loop Sync pulse is toggled backwards and forwards to sustain a portion of the sample to the end of the piece.

12.2) Basic Generative Structures 2

This is an exception from all the other videos as it begins with a performance setup already established.

Less detail will be give here, as it is easy to see at a glance that this setup is very similar to the previous video. The main difference is that the Synths are just playing drones rather than arpeggiating, and there are initially two Sample Players being controlled by a Pulse and Loop Sync combination and connected to a single Delay effect.
The player spends some time adjusting the pitch of one of the Sample Players.
An Avatar is connected to a Synth's FM rate parameter and then placed on a Spinner, so that its spinning proximity will modulate the FM rate.
A new Sample Player is created as a duplicate of an existing one. This is also Pulse and Loop Sync controlled. The higher-pitched Sample Player is moved closer to the Delay, thus increasing its affect.
The previously stationary Synth is placed on its own Spinner and set in motion, again at a different speed from the rest.
A Phaser is added to the third Sample Player, and its Avatar is turned to the right in order to pan the sound left. The Phaser rate is modulated by a Parameter LFO.

The player selects and reduces the size of all fields simultaneously, thus fading out the scene.

Story Mode:

Please note the Performance Mode videos should be considered the primary illustration of the expressive capabilities of this research.

Chapter 1: Full Playthrough

A full commentary will not provided for Story Mode in the same step-by-step detail as Performance Mode. As discussed in the Exegesis all of the audio-visual interactivity in Story Mode has filtered down from the development of the Performance Mode framework.

In general the interactivity in the demonstrated section of Story Mode is much simpler than Performance Mode, as this section represents an early part of the game where players are eased into new concepts. Broadly speaking, the concepts applied are:

1) Avatar-proximity-based mixing of multiple sound sources, particularly in the case of the “background music”, whose various parts each emanate from specific epicentres around the game-space.
2) Melody puzzles: where the player learns/interprets a melody and must play it back (using the Avatar’s instrument) in the correct context. When the flame appears above the Avatars head, this means the player is playing notes via the Avatar’s instrument.
3) Draggable parameters: examples seen in this video include the sundial puzzle where a dial can be dragged up/down on a single axis, and the moon and stars puzzle where multiple parameters can be moved on X and Y axes (equivalent to Performance Mode’s Multi-Parameter Gesture Surface). These can control any of the parameters common to Performance Mode, and the video demonstrates many examples such audio effects, camera filters, and global lighting.
4) Note input into synthesisers or sample-players: the player-characters instrument (shown by the flame above the Avatar’s head) is the most

common example of this, but there are also several environmental examples in this video, such as the vertical music-stones, the stepping-stones, the spinning crystals, the song of freedom, and the lily-pad puzzles, each of which show different modes of interaction whereby the player can directly trigger individual notes or note sequences.

Appendix E: Pallas of Vines: video game for Mac OSX

See attached USB drive.

Appendix F: Performances, Conferences and Academic Involvement

Performances

- Music for Strings and iThings, Adelaide, 2014
  - a commissioned video-game piece for the Zephyr String Quartet, based on an early prototype of the research, 2014

- Art Gallery of South Australia, Adelaide, 2015:
  - Pallas of Vines as an audience-interaction performance

International Conferences

- Games for Change Festival, New York, 2015
  - Pallas of Vines as interactive installation

- Computer Music Multidisciplinary Research: Music, Mind and Embodiment, Plymouth, 2015
  - Pallas of Vines as interactive installation

- International Symposium for the Electronic Arts, Vancouver, 2015
  - Pallas of Vines as interactive installation

- New Interfaces for Musical Expression (NIME), Brisbane, 2016
  - Pallas of Vines as a 3 day interactive installation
    - Radio interview with ABC Brisbane at NIME (see attached USB, Appendix F)

- Unite, LA, 2016
  - Presentation on Pallas of Vines (see attached USB, Appendix F, for video of presentation)

- Montreal International Game Summit, 2016
  - Presentation on Pallas of Vines

Academic Involvement

- Visit to MIT Game Lab, including expert feedback from Philip Tan and Scot Osterweil, 2015

- University of Southern California's Game Innovation Lab: 6 months of Visiting Scholar research, including expert feedback from Chanel Summers, Sean Bouchard, Richard Lemarchand and Tracy Fullerton, 2016

Appendix G: Design Process Screenshot Archive

These screenshots were taken throughout the 3 year development of this research. They document the transition from an abstract 2D prototype to the fully realized Performance and Story environments of Pallas of Vines.

See attached USB drive.