Exploring Adaptive Reverberation Systems in Video Games

Completed as part of a BSc in Creative Music Technology by Luke Child

Summary

This report aims to explore adaptivity in reverb systems and how they could improve the workflow of game development. Furthermore, this project aims to create a new adaptive system that can harness cross-disciplinary information to create symbiotic relationships between various design teams. A system is developed that utilises the rendering process of raycasting to gather environmental information provided by non-audio designers and interpret this information for use with a reverb filter. This system is compared against current methodologies for adding reverberation to video games in both implementation time and the successfulness of environment representation in the characteristics of the reverb filtering. Initial findings suggest that such a system would be beneficial in reducing implementation time for reverberation and, with additional parameter control via environmental stimuli, could also produce reverb filtering that can rival current methods of reverb implementation with less effort required.

Acknowledgements

I'd like to pass on thanks to my supervisor, Phil Phelps, for the guidance and support throughout my project, and for the occasional grilled chicken lunch breaks where we discussed this project.

Additional thanks are given to the members of the Wednesday Reading Group for the continued support both through this project and university, and for being the nicest people I've had the pleasure of knowing.

Finally, love and thanks to my inspirational partner, Paige, my parents Sharon and Tom, my sister Megan, and my nans Leslie and Phyllis for helping me get this far in the first place.

Contents

Introduction

With the popularity of video games reaching new heights with each passing year, the need for efficient turnarounds in game development is being ever more necessary to match the demand for new game experiences. With video game sales at circa £100,000,000,000.00 for 2018, and growth being calculated over the next 3 years (UKIE, 2018), it seems appropriate to develop futureproofed, adaptive systems for use in video game design to maxmise the workflows of design teams to keep up with demand. This ideology is applicable to all areas of video game technology; this dissertation will focus on audio, specifically reverberation.

Adaptivity is not a new concept in video games. In games such as Borderlands 2 (Gearbox Software, 2012), when playing with a party of friends, collecting experience points becomes faster or slower depending on how close to the groups collective level your own level is, meaning there is never a large gap between the highest and lowest levelled player. In audio, music is often used as a signifier to alert a player when an enemy is nearby, a health bar is low, or when they have entered a new area. The Elder Scrolls IV: Oblivion (Bethesda Game Studios, 2006) alerts the player that danger is nearby by changing from slow and peaceful medieval music to using intense, full-orchestra compositions. Adaptivity in reverberation is lesser explored, however. Certain modern games such as Overwatch (Blizzard Entertainment, 2016) have developed delay-based systems to help reinforce the effects of their reverbs, however a fully-adaptive reverb system is yet to be implemented as a new standard in video game design. Currently, a primary method of adding reverberation in video game engines is to use a designated volume or area. when a player is colliding with it, the area adds reverb filtering in varying degrees to the corresponding sounds affected by it. In the Unity engine, the game engine used in this dissertation, these are Reverb Zones (RZs henceforth). These are placed into a level like environment objects and given a parameter snapshot that outlines the reverb filtering they provide. However, when allocated to a level, they are static, providing a filtering characteristic that does not change. With reverberation in the real-world constantly changing due to a listener's environment, it seems fitting that reverberation in an interactive, player-driven environment should adapt, too. Proposed is the concept of an Adaptive Reverb System (ARS henceforth): a system that identifies characteristics about the environment that is it currently housed in at any given moment. It then appropriates this information to the parameters of a reverb filter. Since reverberation is determined not only by sound design, however by the physical properties of an object, it is fitting to explore how adaptivity in an ARS may provide benefits in development workflow when creating a game.

In video game design, there exist multiple design teams whom work independently of one another to design assets for use. Through the exploration of adaptive systems in the development stages of a game, symbiotic links between design teams can be discovered; showing awareness to cross-discipline information can inform an audio system using graphics data, for instance. For reverb systems, this could include harnessing object properties or interpreting rendering information such as distance and visibility. Further afield, this dissertation also discusses how adaptivity in a reverb system could enhance other in-game mechanics and will explore case studies that document this. Such a system will be looked at in the context of the Agile software development method through a game development lens (Harteveld, Kortmann. 2009). With Agile development being the underpinning for many modern software development teams, it seems fitting to explore systems that further increase the agility of this development workflow. Specifically, this project will determine how systems specific to video games such as the ARS can have benefits in various design stages for game development compared to current solutions.

Outlined below are the objects to be achieved by this project:

Context

-Explore the current state of audio in contemporary video games and engines

-Isolate video game reverberation and outline its specific purpose in video games

-Understand the effect of reverberation on a player's gameplay experience

-Recognise the technical limitations of real-time DSP in interactive contexts

-Discuss current reverb implementation practices and systems in place

Design

-Outline the basis for an ARS being conscious of research material

-Consider existing projects and experimentation in video game reverberation

-Isolate successes and downfalls in these projects for consideration in ARS development

-Develop a comprehensive skillset in the Unity Engine by assessing its framework and audio engine

-Research Unity rendering practices (raycasting) for use in the ARS.

Implementation

-Utilise raycasting to gather virtual environment information.

-Analyse gathered data and reach appropriate conclusions

-Apply information to corresponding reverberation filter parameters

Conclusion

-Compare outcomes of the ARS against existing reverb practices.

-Consider further mechanics that may provide useful information for parameter mapping

Background Research

Project Concept

The basis of this project stems from the early modding communities of Microsoft's (previously Mojang's) 2009 video game Minecraft. Minecraft randomly generates environments or biomes for the player to explore; a feature in each biome is caves and cavern systems. One 'modder' created a modification for the game that recognised when a player entered a cave, and resultantly added reverberation to the sounds heard within the game. The modification provided a system that adapted to its surroundings to adjust a reverb filter according to environment information. The idea of a system that adapts to given inputs serves as the primary purpose for this report and will be explored in the context of audio and video games. Specifically, the idea of the ARS will be discussed and prototyped to uncover the benefits and pitfalls of adaptive audio design. Furthermore, the impact of adaptivity in video game design will be assessed to determine its influence on efficiency in game development workflow.

Game Development Workflow

Picture lock: Draw comparisons between motion picture/tv audio; locking video edit to then add audio Making systems that, if you need to edit video at a later date, the audio is no longer messed up. Adaptivity would flip this on its head, creating bins of audio/video and having the system assemble it before your eyes. – working dynamically in real time. Tension between end goal: dynamic, and human design development process – linear.

When designing and creating digital media, there are multiple design teams creating a variety of components. Considering anything with audio and a moving image, it seems desirable to tackle the production linearly; creating an element, acknowledging its creation, and moving on to the next stage. A common term used in film and television for this is 'picture lock'; a state in which the video has been decided complete for showing and can be passed on to a sound designer to add sounds and music. An issue with such a linear approach is that "that picture lock is often not all that firm, and the picture continues to be edited while the music is written." (Holman, 2010). If one element of a design team wishes to commit a change after a section is locked, much of the work created after the lock stage will need to be adjusted to accommodate for the changes. This is a problem that is resonant throughout the creation of digital medias. In video games, for instance, audio may be created to match the visuals of a created animation.

However, should this animation need to be extended or shortened after committing audio to it, “the version of the [animation] has often changed, and sync is no longer ensured” (Holman, 2010). To combat this, it seems pertinent to bring about the removal of linearity within the development process, and to provide systems that are better equipped for dealing with change – adaptive systems. In the case of this report, the concept of adaptivity in development processes will be explored through the lens of video game development and design, with a heavy focus on audio and reverberation.

Video game creation requires the input of many different teams for success. Depending on the size and scope of a project, varying levels of specialty are needed. Smaller projects may omit certain elements such as audio designers and opt to use sourced assets from the internet, whereas large projects may have a collective of audio designers working on individual elements of a game, such as foley, implementation or VO for example. A typical workflow may follow the Agile and SCRUM method of development. Approaches towards Agile development via games design have approached, such as Triadic Game Design (Harteveld and Kortmann, 2009), however, a standard model for Agile will be used for the context of this report.

Agile is a method of design that challenges developers to tackle complex projects using iterative design methods, solving problems and introducing features in smaller stages rather than releasing a complete product in a singular launch (Atlassian, n.d.). SCRUM furthers this ideology by providing developers with iterations of a fixed-length called ‘Sprints’. These sprints are performed with the goal of completing a small area of the project, with heavy focus on teamwork. “With Scrum, a product is built in a series of iterations called sprints that break down big, complex projects into bite-sized pieces" (Atlassian, n.d.). Agile and SCRUM workflows rely heavily on turnaround and completing tasks in a given time limit to a high standard. Video game development works in a way that benefits from this kind of approach to development. Development teams in video games are made up of many different design teams, and as such it is important that they can understand and utilise eachothers work in order to work efficiently. If a level designer is creating a level for a graphics designer to texture, it is important that both teams can work in harmony with little need for interaction. If a ‘medieval’ game with many wooden and stone textures was to be created, it would not necessarily be appropriate for a level designer to create in-game features with modern design elements such as inch-perfect constructions or physics-defying shapes. This becomes particularly difficult when discussing the role of audio alongside other design teams. Focusing on reverberation it may be down to the audio designer to craft the characteristics of a reverb.

However, those characteristics need to match the shape and physical properties of the environment that it affects; two jobs completed by the level designer and graphics designer respectively. To fully utilise an Agile development practice, better interplay between design teams needs to be explored. Adaptive systems may hold the key in creating better workflow between teams by using information from all parties to create a unified whole.

Cross-discipline Working in Game Development

In order to build an ARS that can adapt to information from its surroundings, the focus of each design team should be decided from the start. This way, an orthogonal approach to adaptive system design can be achieved where each design team works symbiotically with another without altering each other's work. Considering an ARS, the characteristics that make up reverberation in real life depend on a multitude of factors. Physical room properties, air density, and the characteristics of the source sound are all factors that influence the reverberation that filters sound in a given space. When the digital realm is approached, each of these influences is handled by a separate design team:

-Level designers will handle the shapes and sizes of a space.

-Graphics designers will handle the textures and visual aesthetic of a space.

-Audio designers will handle the resultant reverberant quality of a space.

-Physics designers (should a game require it) may influence 'space' with factors such as water or gas, thus changing the air characteristics.

In standard practice, it may be common for each design team to work independently to then accumulate when each stage reaches a completed stage. Provided that guidelines between teams are followed, this may be successful in small projects, however with larger scale projects, discrepancies in ideas may result in those ideas not being correctly realised. As with the aforementioned example of a medieval game level, if an audio designer is told to craft a reverberation filter to represent a large area such as a dining hall, this could be interpreted as a cavernous space or a large padded cell for instance, when in reality they may require a completely different aesthetic. Studies have shown how reverberation can enhance the emotional qualities of instruments on playback (Mo et al, 2016) and the same principle could be applied to emotional response to space, should it be required. For example, Story Designers may wish for a room to have a dark and overbearing tone, or perhaps a lighter and artificial tone, both of which cannot be gauged from a room's characteristics alone.

Furthermore, in abnormal scenarios, such as polygonal, abstract room shapes, or perhaps rooms with a multitude of materials, the work of the audio designer becomes exacerbated as they have to cater for multiple variables associated more so with physical design than audio design.

These insights draw the conclusion that it is beneficial to all parties to provide cross-discipline information to help inform each other's resultant design process. This idea can be applied to the ARS, the focus of this project, however it is first important to understand the parameters of digital reverberation and how these can resultantly be informed by information from other design disciplines.

Understanding Reverberation

Reverberation "refers to the prolonging of sound by the environment… it interacts with the environment, carrying with it to the listener an imprint of the space, and a sense of the objects and architecture present" (Valimaki, 2012). When a sound is present and audible within a space, the characteristics of the space (provided by its physical makeup) influence and filter the sound before it is heard. This could be in the form of frequency filtering through absorption, or through the prolonging of sound through reflections. These filters occur and execute naturally in the real world and are ever-present in day to day life. However, the simulation of such phenomenon in digital realms poses a plethora of computational issues – simulating the movement of billions of particles at once is no light task. Instead calculated approximations are performed to create plausible impressions of spaces that, though not perfectly accurate, present filtering with a high enough fidelity to give the listener a believable experience.

Perceptual Fidelity

The idea of fidelity is a brief point for consideration in-game. Fidelity is defined as "the degree to which the detail and quality of an original, such as a picture, sound, or story, is copied exactly" (Cambridge Advanced Learner's Dictionary and Thesaurus (4th ed), 2013) Audio fidelity is a comparison of similarity between an audio recording of a sound, and the original, unrecorded sound. Digitally, this degree of similarity can be measured through digital recording means such as sample rate, bit depth and hardware quality, however the question of 'does this recording sound like the thing it represents?" is harder to answer. For reverb, this could be translated to "does this room sound like its corresponding room in real life?".

Media Studies professor Dr Kristine Jørgensen states that perceptual fidelity "demonstrates how sounds can support a lifelike and credible environment without conforming to standards of realism" (Jørgensen, 2007, pp. 61). It can be understood thus that perceptual fidelity takes the principles of fidelity and asks the listener to consider the reproduced medium against their own dispositions towards the original version of the medium – "does this plausibly sound like what I THINK it should sound like?". Jørgensen identifies that sound must provide functional value to an player regarding the object it is descriptive of, using the term "functional fidelity" to describe this.

Applied to digital reverb, the chaotic nature of dispersion means billions of particles would need to be simulated per second to achieve a real fidelity even close to that of real life. However, in pairing a listener's disposition of how a space sounds (such as knowing the sound of one's voice in a bathroom) with an approximation of that room's characteristics using an ID3L2 reverb, reverberant qualities can be created that are both perceptually and functionally valid. For reference, I3DL2 is known as the Interactive 3D Audio Level 2, is a set of guidelines for creating reverberation outside of a standard stereo field for more realistic sound environments (IASIG, 1999). With the concept of perceptual and functional fidelity in mind, links between the roles of graphics design and audio design can be made to provide a reverberation experience that satisfies the expectations of the listener.

Utilising Visual Stimuli

When a graphics designer approaches the texturing of a level, they have control over the look and feel of each object. Wooden objects can be given wooden textures to make them appear grained and rough, and vice versa with metallic surfaces being smooth and shiny. In reality, the properties and visual aesthetic of materials are contemporaneous, however in digital realms, they exist separately. Graphics designers determine the aesthetic, level designers or physics programmers determine the properties of an object, such as mass and smoothness. For an ARS, it would make sense for the aesthetic and physical properties of an object to co-exist in one format. This way, the system could interpret this information, and decode it to influence the character of the filtering it provides. The materials that a room provides can be discerned by a listener purely through listening and association through previous experience – sounds in a wooden room sound 'wooden'. Taken from an unpublished PhD thesis by Dr Peter Svensson, it is noted that "It the degree of mismatch between aurally and visually perceived room size is often mentioned as being one of the factors influencing the acceptance of such a system (Larsson, Västfjäll and Kleiner, n.d.).

A proposed solution to this is to provide textures with physical properties, so as to allow the graphics designer to add visuals to a game level whilst also providing basic physical information about the texture they are to use. For instance, for metal textures, a graphics designer could provide not only the visuals of a metallic object

Discussion

It is evident that information can be used in a cross-discipline fashion to help inform an ARS. It is noted that in current workflows, a degree of interplay is needed between teams to maximise the efficiency of their respective design processes, however, there seems to be little interaction between audio and other mediums currently. It seems fitting that audio designers, especially in the case of the ARS, should be provided with basic information regarding a game environment so that an ARS could be easily implemented, be it for prototyping, or for the final version of a video game.

Research Studies

Though there is a lack of crossover between audio and other design teams, there are examples in modern video games that show significant benefits from the implementation of such a workflow. Two case studies are outlined below, the first exploring the idea of using distance parameters and raycasting to provide information about player and enemy location, and the second using adaptive audio to instill and control emotional feedback from players.

Research Example 1 - Overwatch

At the Game Developers Conference in 2016, Scott Lawler and Tomas Neumann from Activision Blizzard outline a customised 'quad delay' for their 2016 video game 'Overwatch'. They presented an early delay system that utilises raycasting to inform four delay lines, EQ units and panners. (Lawlor and Neumann, 2016). The example given is a player shooting a rifle into a square surrounded by buildings, versus shooting into a vast open space. When in the populated area, the resultant gunshot reflections sound "brighter" and fuller. When fired into the vast space however, the reflections are "bassier [and] more muffled". The use of panners in this instance help a player to instinctively recognise where a player is in their virtual environment in relation to the direction they are currently facing. Furthermore, a secondary use for this system is in providing the player with spatial information on enemy players and their corresponding locations. This system captialises on the informative nature of sound and reverberation and uses it to give players an advantage in-game.

FIGURE 1 - USING RAYCASTING TO INFORM AN EARLY DELAY SYSTEM IN OVERWATCH (BLIZZARD ENTERTAINMENT, 2016)

The conference overarchingly discusses the use of sound as a source of information to the player, as opposed to a solely aesthetic approach. Lawler and Neumann take an approach that is conscious of a concept known as transdiegesis. To understand this concept, diegetics must first be understood. Diegetics, with an audio focus, studies the origins of a sound in a given environment. Jørgensen defines this as "a concept that defines and separates spaces of action and information", stating that "diegetic sound originates from within the fictional universe… while extradiegetic sound is a commenting aesthetic feature with no direct connection to an actual source within the fictional universe" (Jørgensen, 2009, p. 98). Expanding this, it concerns itself with whether a sound exists visually in the created universe (such as the ambience of a city in an urban environment - diegetic) or in the universe of the player themselves (background or thematic music - extradiegetic). Many sounds fall easily into these categories, but when discussing the idea of cross-diegetic audio, a third category is needed – transdiegesis. When considering reverberation, it can provide both aesthetic quality and spatial information. Using the context of Overwatch's early delay system, both can be discussed. Firstly, if a player is shooting a weapon in a given space, they will expect their weapon to be affected by the environment's characteristics. The reverb can provide a realistic sheen to the sound through appropriate filtering, but in the case of a first-person shooter such as Overwatch, further information can be extrapolated. If an enemy is firing their weapon, which can be heard by the listener, in an open space, their weapon will be reverberated to reflect the open space. This information can be heard by the listener to disambiguate the location of the enemy; a large open space with little reflectivity may suggest the player is outside, for instance. By being conscious of this, the design of adaptive audio systems, especially the ARS, can be informed in such a way to not only streamline the addition of audio to a video game, but also to help players with understanding their surroundings with minimal additional effort from other design teams.

Research Example 2 - Alien: Isolation

Though not in Overwatch particularly, adaptive reverberation could be used to enhance the emotion of a situation also. Studies have shown that reverberation can add emotional characteristic to instrument performance with results showing that large reverbs a "strongly significant effect on Mysterious and Romantic", and a "medium effect on Sad, Scary, and Heroic" emotions (Mo et al, 2016). With this in mind, and while being aware of the emotional characteristics of a location, emotional response could be enhanced through using adaptive reverb that takes locational cues to adjust its intensity; the use of a large, booming reverb could be used to signal a dangerous enemy – a scary situation.

Alien: Isolation (Creative Assembly, 2014) uses sound as an informer of where the enemy, the Alien, is at a given moment. When footsteps begin to sound out with heavy reverberation, the player instinctively perceives them to be distant, but with the intent of getting closer. With less of a filtered (drier) signal, the player understands the enemy to be closer, since the sound seems to travel less distance to travel before arriving at the player. The player soon develops an association between the tension and worry of the enemy to the filtering effects provided by a distant, boomy reverb. Not only does the reverb filtering provide the player with information on where they are in-game (diegetic), it also provides them with information on the dangers within a level (extradiegetic) and amplifies the implied emotion of the situation at hand. Creative Assembly sound designers Sam Cooper and Byron Bullock further detail their uses of adaptive sound systems in Alien; Isolation in an interview with The Sound Architect Bullock states that "the alien listens for sounds in the environment, looks for light from your torch and can see you in the light. We had to create a sound engine that could cope with that, a dynamic system that could react to both the player and *Alien*. This then allowed us to change the sound, music and mix dynamically to help to suit any situation" (Bullock, 2014).

Research Studies Discussion

The completed studies give insight into the role of adaptive systems in two modern video games. Both examples show successful implementations of systems that provide the player with additional gameplay information due to their adaptive natures. This speaks less towards improving the efficiency of development workflow; having these systems may not speed up a development process, however, should new areas or information be added to the game, the systems will not need tweaking to compensate. These studies also represent the further benefits an adaptive system could have to a player's gameplay experience.

The gathered information argues in favour of reverb being a transdiegetic and an emotion-reinforcing sound function. Relating this to the ARS, audio designers could spend more time refining the characteristic of a reverb in specific scenarios to better inform players of their surroundings rather than spending time implementing basic reverb in the first place. Similarly, communication between design teams would allow for easy additions to game levels (such as adding a new area or map) that would instantly work with the ARS. This cumulatively provides a system that is adaptive to its environment and any corresponding changes, that reinforces the emotional narrative that a scene could provide whilst also informing listeners about points of interest in their environment.

Introducing Unity and the Unity Audio Framework:

Software is required to create video games. A common term for such software is 'game engine', which typically involve a 'run time' and an 'editor' as the hub for a video games development, and a 'core' feature set of functionalities. FIND A GOOD DEFINITION QUOTE HERE. This project will be using a game development 'engine' called Unity, a free video game creation engine launched in 2005. (Unity, 2019) Currently, Unity is appeals to indie developers and students because of its free tier (zero start-up costs for aspiring developers) and has a vibrant community developing asset (including graphics, audio, and code) sold in the Unity Asset Store.

As standard, Unity comes with an audio engine (an engine within an engine) similar to many mainstream Digital Audio Workstations (DAW's henceforth) such as Cubase or Pro Tools. Audio in Unity can be approached with the mindset of using a DAW, however with the addition of spatialisation and logic added in. Unity can use audio files as in-game assets and assign them to various functions. For reference, this section of the dissertation will refer to elements found in the Unity 2018.3 User Manual (Unity Technologies, 2018).

Parallels between Unity Audio and DAW Channel Strips

In many instances, audio plugins are used as inserts. This functions well when it is necessary for every dry sound before it in a signal needs to be processed. This method results in the dry sound and resultant processed sound becoming a singular entity. It is often necessary however to treat dry sound independently from the effects that affect it. Audio effects can be set up in parallel, allowing for a dry version of a sound as well as a processed version to exist and be treated independently of one another.

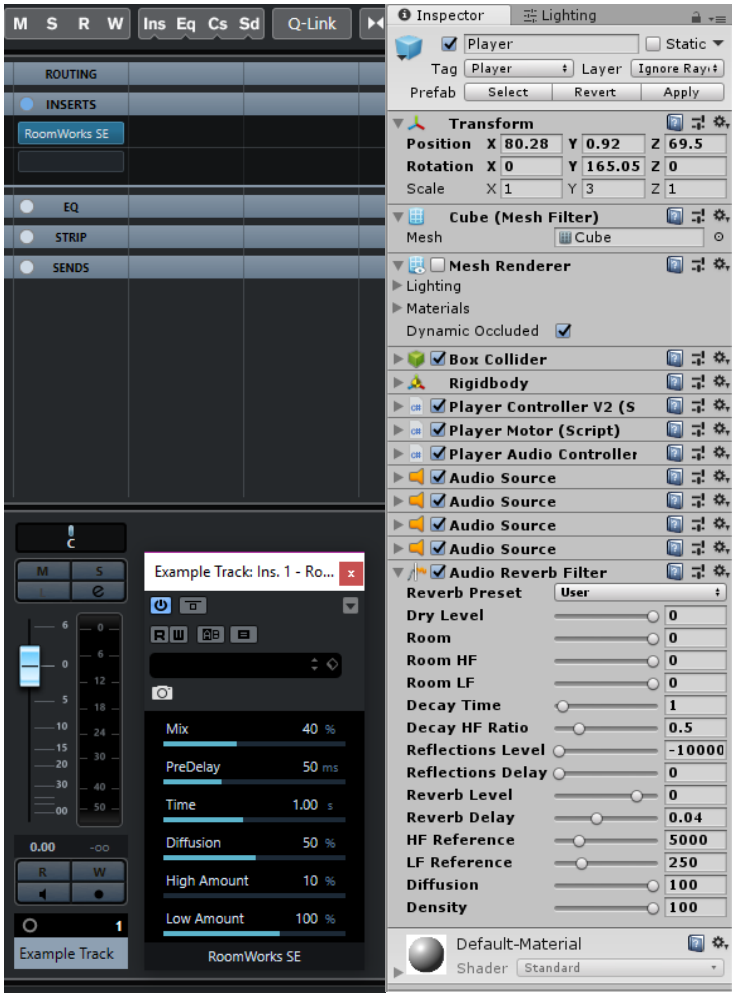


FIGURE 2 -CHANNEL STRIPS IN DAW'S SUCH AS CUBASE ARE SIMILAR TO GAME OBJECT HIERARCHY IN UNITY. THE REVERB FILTERS IN BOTH PICTURES ARE IN SERIES, FILTERING ALL AUDIO BEFORE THEM.

Two common scenarios are presented that identify the benefits of parallel effects processing over series processing:

Scenario 1 – Series:

A player is a location that has a sound source to their right and a reverberant area to their left. The dry and mono sound will need to be panned in relation to the listener, and as such is placed rightwards. The sound propagates throughout the reverberant area, also. However, since the reverb filter is placed on the Unity game object that houses the sound, it must originate from the same location; rightwards of the listener. The issue here is that the dry source sound is not independent of the reverb filter, meaning the filtering cannot be placed separately.

Scenario 2 – Parallel:

This scenario is the same as scenario one, however the reverb filter is included as a send. The dry sound on a game object is sent in varying degrees to an independent reverb unit. This filter can be placed separately from the original dry sound. Thus, if the reverberant location was panned left, the filtered version of the dry sound could be heard leftwards, whereas the dry sound would still be heard rightwards.

Listeners and Sources:

The base principle that Unity revolves around is the concept of 'Listeners' and 'Sources'. A listener acts as the receiving hub for all audio in a game. It can be inferred that a listener acts as the games 'ears', offering an audio receiving location within a given 3D space. In real life, a person's ears act as the position in space that audio uses as reference; sound to the left side of someone's ears sounds as if it on the left, for instance. Since video games do not have 'ears' as standard, a listener must be provided. Often, video games have a playable character or viewpoint that the player sees from, and this character needs to act as the reference point for audio, in place of the player themselves. A source, on the other hand, functions as an audio emitter that represents the location of a sound within a space, and concurrently plays that sound in relation to the position of the listener.

Audio in 3D video games can undergo spatialisation in order to better represent positioning in its corresponding environment. Spatialisation is "the process of causing something to occupy space or assume some of the properties of space" (HarperCollins, 2018), and the Unity standard spatializer makes use of level attenuation, stereo panning, and basic filtering to create the impression of space. Similarly, in a DAW, parameters such as stereo amplitude difference, fractional delays and all-pass filtering can be used to create the impression of movement in a 3D space (see 'PsychoPan' in bibliography), however Unity's are dynamically-

altered based on in-game location information; the further an audio source is from the listener, the quieter it is, for example. Much of this information is gained through the use of Head-Related Transfer Functions. A head-related transfer function is “a function used in acoustics that characterizes how a particular ear (left or right) receives sound from a point in space” and allows for the digital replication of acoustic phenomenon such as Interaural Time Difference (ITD) or Interaural Level Difference (ILD), the difference in time of receival and difference in pressure level between a human’s ears, respectively. The Unity Audio Spatializer SDK handles this should it be used.

Some games have begun to place the ‘ears’ of the video game onto the player, themselves. With the advancement of VR, the player can be used as the viewpoint of the character in-game through the use of a motion-sensing headset. Games such as 2017’s Skyrim VR and Fallout 4 VR by Bethesda Game Studios utilise the first-person elements of their video games to replace the in-game character with the player themselves through the use of VR headsets. When this is done, the aforementioned ‘ears’ needed as a reference point become that of the players. Since VR headsets have headtracking capabilities, every turn a player makes can ‘turn’ the listener within the game to receive sound relative to the rotation of the players head. The listener within the game adapts to the positioning of the player, meaning that audio designers can focus purely on including sounds in the environment and let the audio engine handle the rest. Studies have shown that 3D audio can further help with navigation; with no difference between using 3D audio over a physical map. (Sandberg et al, n.d.)

Listeners and source are the foundation of audio playback in Unity, and alone can provide a plethora of information to the player. However, solely using listeners and sources cannot provide sufficient information about the environment that houses a sound. To do this, Unity uses Reverb Zones to provide spaces with characteristic ambience.

Reverb Zones and Filters

A RZ is a representation of a physical space that, when a listener is present within it, filters sounds that are received by the listener. On its own, a RZ acts as a boundary that a listener can enter to receive filtering in various degrees; an RZ is the frame of a real-world room, however with room characteristics omitted.

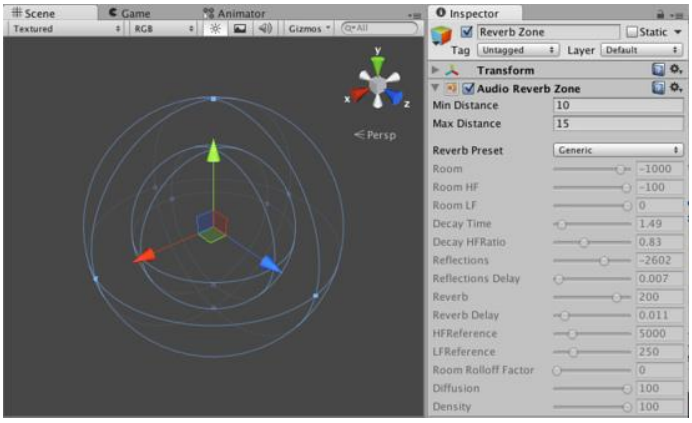


FIGURE 2 - HTTPS://DOCS.HUIHOO.COM/UNITY/4.3/DOCUMENTATION/COMPONENTS/CLASS-AUDIOREVERBZONE.HTML

As standard, RZs are spherical spaces that can be placed into a level. They come with a reverb filter as standard (explained below) that can be given a parameter snapshot. This means that when a listener enters the space, the user-defined reverberation parameters filter sounds recevied by the listener.

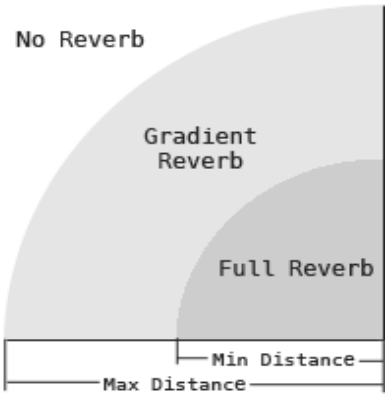
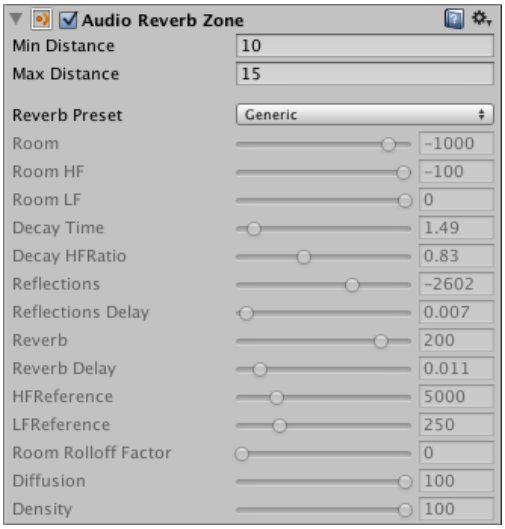


FIGURE 3 - THE BASIS FOR UNITY'S REVERB ZONE (UNITY DOCUMENTATION)

The processing of a dry sound is handled in Unity's sound engine by a 'reverb filter' component. This is compliant with the I3DL2 specification (REF?). The reverb filter component behaves like an 'insert' on a DAW channel – processing dry audio to source buffers

before it based on its set parameters. This is the premise for n RZ, in which the filter is applied to audio sources that collide with the zone. A typical approach, however, may see the designer applying a singular reverb filter to the master audio output to apply reverberation to an entire game sound mix. This master effect would then have a wet/dry parameter that controls the intensity of the reverb blend to represent changes in environment, such as moving from outdoors to indoors. RZs act as areas that house a reverb 'snapshot': a set of defined parameters that, when activated, alter the parameters of a referenced object to match their own. An RZ acts similarly to a sound emitter in the sense that distance calculations are performed using a 3D version of the Pythagorean equation a2 + b2 = c2 - this works out the magnitude of a 3D vector. The information gathered from working out the vectors adjusts the parameters collectively to create the impression of distance attenuation.



Reverb zones are a prime example of an area of crossover regarding project development. Since RZs and filtering alter sounds in an environment, it seems logical that they are dealt with by the audio designer. That said, RZs are placeable objects that can be moved around in a level and, with advanced knowledge of Unity, can be created in various shapes using alternative collision meshes. This role is suited more to a level designer, whom has better skills in object placement and collision technology in Unity. Referring to the ARS, if a system were to be created that did not require placeable zones to signal that an area warrants reverberation, this issue would be avoided.

Raycasting

In order to create a system that doesn't require RZs, locations in a game would need to constantly provide information about themselves to an incessantly-active reverb filter. It could be assumed that the video game houses one singular reverb zone that is dynamically changing as the player moves around. The snapshots provided by reverb zone are static; at any given point in the same space, the reverb characteristics will sound the same, varying only in their distance attenuation. The governing principle of the ARS is that its reverb filter it houses should adapt to the surroundings it is presented with. As such, a static reverb filter will not suffice. Though the spatialisation functions provide information on distance, they do not provide sufficient information on additional environmental properties, such as surface material, reflectiveness, location/occlusion etc. In order to gather this information, the ARS will make use of a rendering technique called 'raycasting'. Raycasting is the process of casting an invisible 'ray' from a given 3D coordinate in a specified direction. This ray collides with other game objects and can provide identification and distance information on the objects it collides with. The process of raycasting is used within digital graphics representation and mathematics originally in 1982 to render geometry models (Roth, 1982). This has since been improved to be used as a basis in general 3D rendering processes. By expanding the base functionalities of the ray collision, further information, such as information about an object's renderer (the visual properties of an object, `such as texture, metallicness and smoothness).

Renderers and Textures:

All visible physical objects within a game have a renderer. A renderer allows for an object to be made visible in-game and to have physical properties, such as texture, metallicness and smoothness added to it. If an object is smooth and shiny (metal), then the object should look that way, as opposed to a dull, rough object. In real life, these physical properties vary how sound is altered following a collision/reflection. When in a small and tiled room where the tiles have a shared property of both smooth and reflective, it is assumed that the room with sound resonant with quick, dense reflections; like a bathroom. It seems sensical for audio to behave in the same way when faced with objects that look like they should filter sound in a particular way; a small carpeted room with wallpaper walls produce a vastly different sound compared to that of a large, metallic warehouse. Textures in Unity can have both 'metallicness' and 'smoothness' applied to them as standard. As such, this makes reading values from them easy, as graphics designers will have set these up when adding their

textures to objects in game. The player will also be set up with an expectation as to how sounds should be affected by a space due to prior experience, as discussed earlier. Providing objects with custom rendering properties is normally the job of the graphics designer. Though it may not need be a perfect replication, the graphic designer should be able to design objects that both provide subconscious information regarding a space's physical properties, whilst also informing the characteristics of the filtering in place.

If this model of design is applied to the workflows discussed earlier, then more efficient practices can be created. If a graphics designer could provide characteristic information about the environments they are visualising such as object materials, smoothness, density etc, then the ARS would be able to access this information via raycasting to influence the characteristic of its reverb filtering. The audio designer would then be able to place the ARS into a world while providing minimal assistance to adjusting the parameters of the reverb filter.

Creation of the Adaptive Reverb System

Summarising the issues with audio and RZ's in Unity, an ARS can be designed to be both efficient in its performance, successful in its representation of spatial information, and user-friendly for teams who wish to implement it. During the design process, awareness was given to the origins of the project in reverb modifications in Minecraft. The Minecraft universe is parameterised in such a way that all aspects of the environment are available to 'modders'. The damage status of a block, the amount of air blocks in a given space, or any other block type for that matter are all easily accessible. Unity can function in this way given objects are designed in such a fashion that allows this process to happen. With this in mind, the ARS was designed with a 'gather as much information as possible' approach, and future iterations will expand on this.

The final version of the ARS was designed after considering the results from a series of prototype studies completed to the building of the current ARS. The results from these studies are documented below. Analysis will be completed using a system of successes, problems and solutions. SWOT analysis was considered for this stage, however, was ultimately deemed inappropriate due to the nature of the assessed studies. Astute readers will note that, though mentioned previously, this is not a project that followed Agile software development practices. The concept of Agile was discovered later on in the development, though the designing of the ARS does follow an iterative design approach.

Case Studies

Control - Reverb Zones

It is first important to collect results using the current methods of reverberation offered by Unity. A small level was prototyped to show how RZs function as standard and also to legitimise the problems hypothesised previously.

Problems:

-Reverb Zone Shapes

As standard, RZs are spherical. For boxed shaped areas, this could pose issues with spheres being too small as to not fully encompass a room, or too large and fall outside the boundaries of a room, thus providing reverberation around the outer edges of a room.

-Static Snapshots

Reverb filter snapshots on RZs are static; the audio designer will set these before runtime, and they will stay this way while the game runs. Reverb snapshots could be swapped in and out to provide variation, however reverberation constantly changes in real life as a listener navigates a space, and it seems appropriate for an in-game reverb system to do the same.

Proposed Solutions:

It is possible to create custom-shaped RZs, which may seem easy to a level designer. However, if it is the job of an audio designer, this may be outside the scope of knowledge if they are less familiar with using the engine. By eradicating RZs for the method outlined in the ARS, this problem would cease to exist.

Prototyping Preparation

Prior to prototyping the ARS, a few basic quality-of-life additions were created to accommodate it:

-Creation of a player and controller

A basic 'player' game object was created to house sound effect playback through C# scripting. Audio playback controls consist of:

- -Gunshot sounds on Left Mouse Button
- -Vocal sample on P key
- -Looping Music Sample on O key
- -Looping Walking sample on W key (when held down)

These sounds were chosen to represent common sounds found in a first-person shooter game.

Furthermore, a simple player controller was added to control the position of a camera to allow a user to move around the level:

- -Forward Movement on W key
- -Leftward Movement on A key
- -Backward Movement on S key
- -Rightward Movement on D key
- -Movement Speed Increase on Shift key (when held down)

-Creation of materials

To mimic the lack of a graphics designer in this project, materials were created to represent property. As mentioned before, Unity materials have a plethora of parameters. Specifically, the smoothness and metallicness will be read, so materials will be created with attention to these parameters.

-Creation of prototype spaces

Finally, to give the user a level to navigate around, basic level design will be completed. The Prototyping Pack from Digital Konstruct will be used to populate the environment to avoid the need to model complex 3D shapes. (Digital Konstruct, 2017)

Prototype 1 - Raycasting and Colliders

Section 1a – Filters and Raycasting

In the first iteration of the system, the reverb filter was set up to be a part of the player game object. In doing this, every sound that was activated from on the player object (i.e. all the aforementioned sounds) would be filtered via one reverb filter. This meant that all scripting elements could be contained in one script that referred to its own game object, keeping the ARS tidy on one object.

Once the filter was functioning statically, raycasting was implemented. Initially, in order to collect a large sample of data, a large amount of raycasts were used, and cast from multiple angles in a sphere-like pattern. Though this provided a large range of values for distance and smoothness, it became a highly demanding process to compute at a fast rate; large values of raycasting performed in many directions up to 3 times per second.

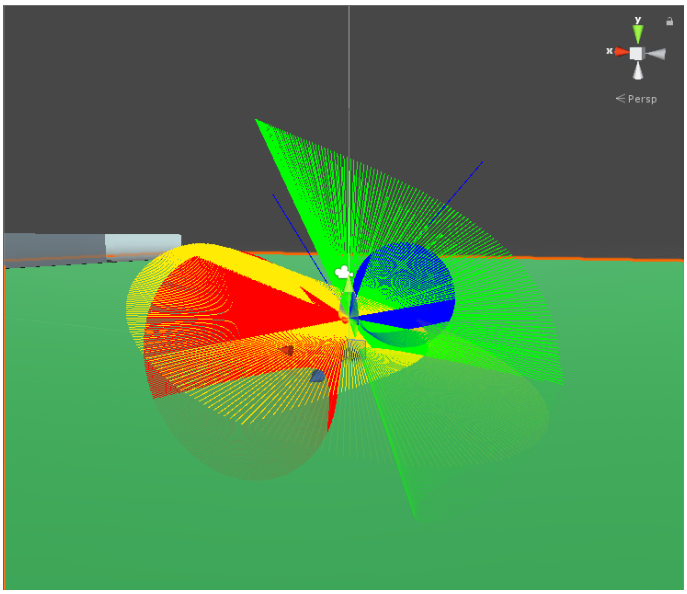


FIGURE 3 - INITIAL USES OF RAYCASTS WERE INFORMATIVE, HOWEVER INEFFICIENT DUE TO THE COMPUTATION REQUIRED FOR LARGE AMOUNTS OF RAYCAST

These values also were accumulated and averaged to find the mean value, creating a large workload per second, even for a small, barely populated environment. Furthermore, raycasts are infinite in length if they are not capped when implemented. This meant that if a player was outside, a raycast could collide with nothing (like the sky, for instance). Resultantly, a distance average would be unrepresentative of the space and would always set its mapped parameter to maximum value.

$$d = \frac{\sum_{i=1}^{n} x_i \ + x_1 \ + x_2 \ldots \ + x_n}{n}$$

d = mean distance

n = amount of raycasts

i = index

x = distance obtained from raycast, where $x \leq \infty$

In this case, the value d is mapped to the reverb filters decay which has a maximum value of 20 seconds. Values above this that are sent to this parameter are truncated to 20.0.

That said, the concept of using raycasts to collect environment data and apply this data to a reverb filter did prove successful in practice, albeit with poor initial results.

Section 1b – Performance Issues

Two problems quickly arose from section 1a; performance issues arising from casting too many rays, and the fact of very large distance values being obtained from the raycasts. To fix the performance issues, varying degrees of raycast amounts were considered. By limiting the amount of raycasts performed, lesser degrees of accuracy were captured by averaging values, however the reduced need for casting resulted in a less computationally-expensive method. Unity's tutorial section outlines a series of best practices when it comes to raycasting. The most efficient raycasting is aware of:

-Using smaller amounts of rays

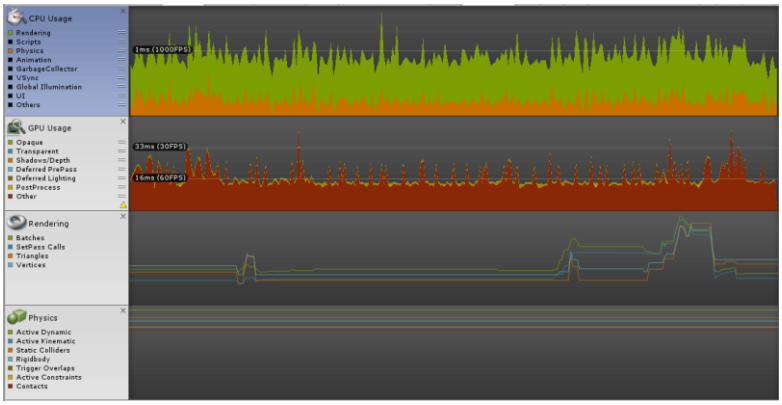


FIGURE 4 – UNITY PROFILER WITH LOTS OF RAYCASTS

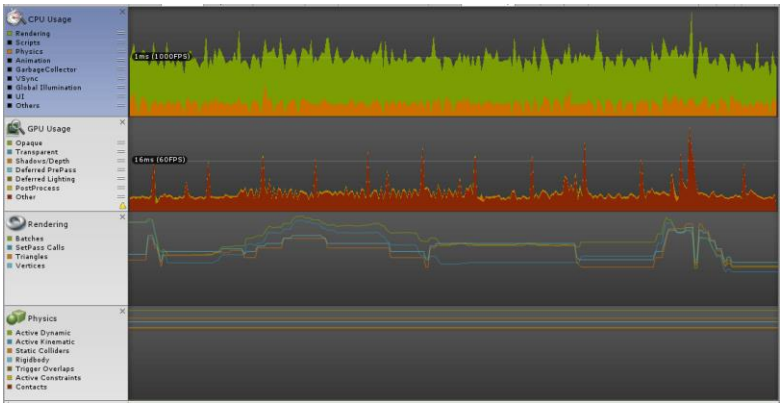
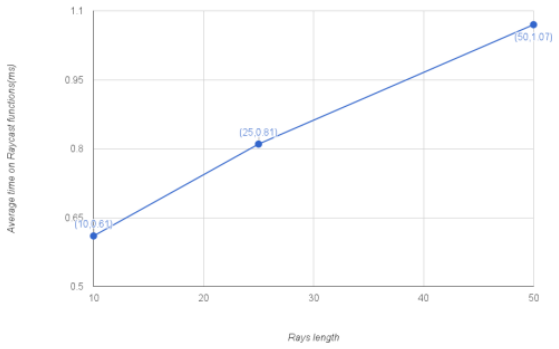


FIGURE 5 – UNITY PROFILER WITH LESS RAYCASTS

Significantly less workload on the GPU with less casts.

-Capping raycast lengths so it is tested against less objects



-Being aware of the collider types that the ray is colliding with

> Mesh colliders provide a more complex geometry to deal with than primitive colliders such as spheres and cubes, meaning more effort is needed from the physics engine per frame for complex

An interesting area of further for study in collision and colliders is that of proxy geometry. In order to reduce the impact of complex static meshes, algorithms have been explored that reduce the complexity level of such meshes by combining areas of dense amounts of meshes (such as a large non-enterable building with decorative features) into a singular mesh. Game developer Epic Games documents this in a study conducted in their game engine Unreal (Epic Games, 2019).

Before Proxy		After Proxy	
Object Count	22	Object Count	1
Triangle Count	27,308	Triangle Count	4,032
Material Count	6	Material Count	1

In doing this, larger areas can be considered as a singular entity, opposed to a multitude of smaller parts. Referring to the previous images, this would significantly reduce the amount of collisions that would occur, as rays would consistently collide with a single, large mesh. This works ideally for an ARS, as it means that not only are environment objects better optimised for use, the system would be interpreting less information at a given time. Furthermore, regarding perceptual fidelity, the influence of multiple smaller object is negligible in the overall scope of the ARS; the influence of a decorative A4 flyer in a room would provide little influence on the room characteristics of a church, for instance.

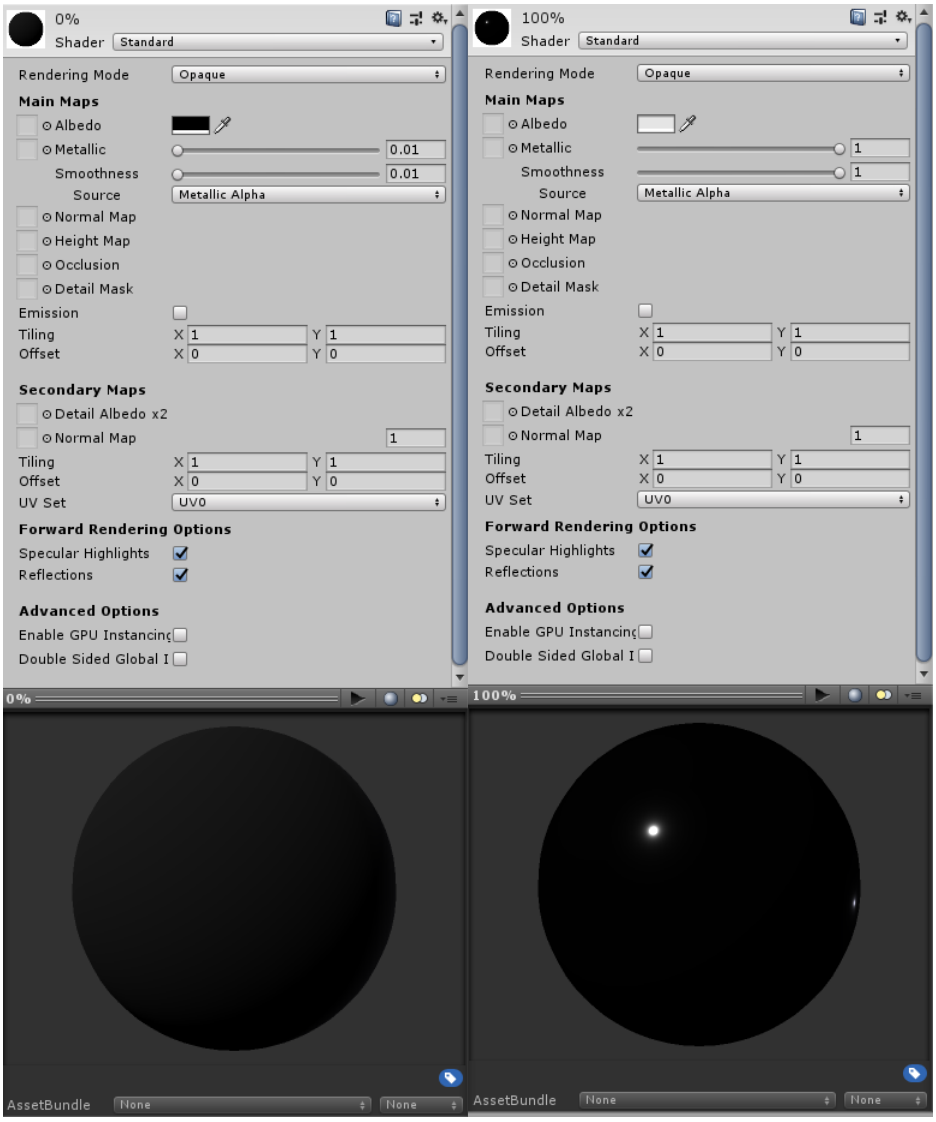
Prototype Discussion

When creating a final design for the ARS, it will be important to consider using smaller amounts of raycasts to minimise the workload of the physics engine. Also, if the system were to be used in a video game, awareness on proxied geometry would be needed to evaluate how this may influence the parameters of the ARS since fewer environmental stimuli are provided. This study shows benefit in creating an ARS as a level designer would simply need to place down environment objects and the system would work out an estimate on the sizing of the space it inhabits.

Prototype 2 - Material Properties and Parameter Influencing

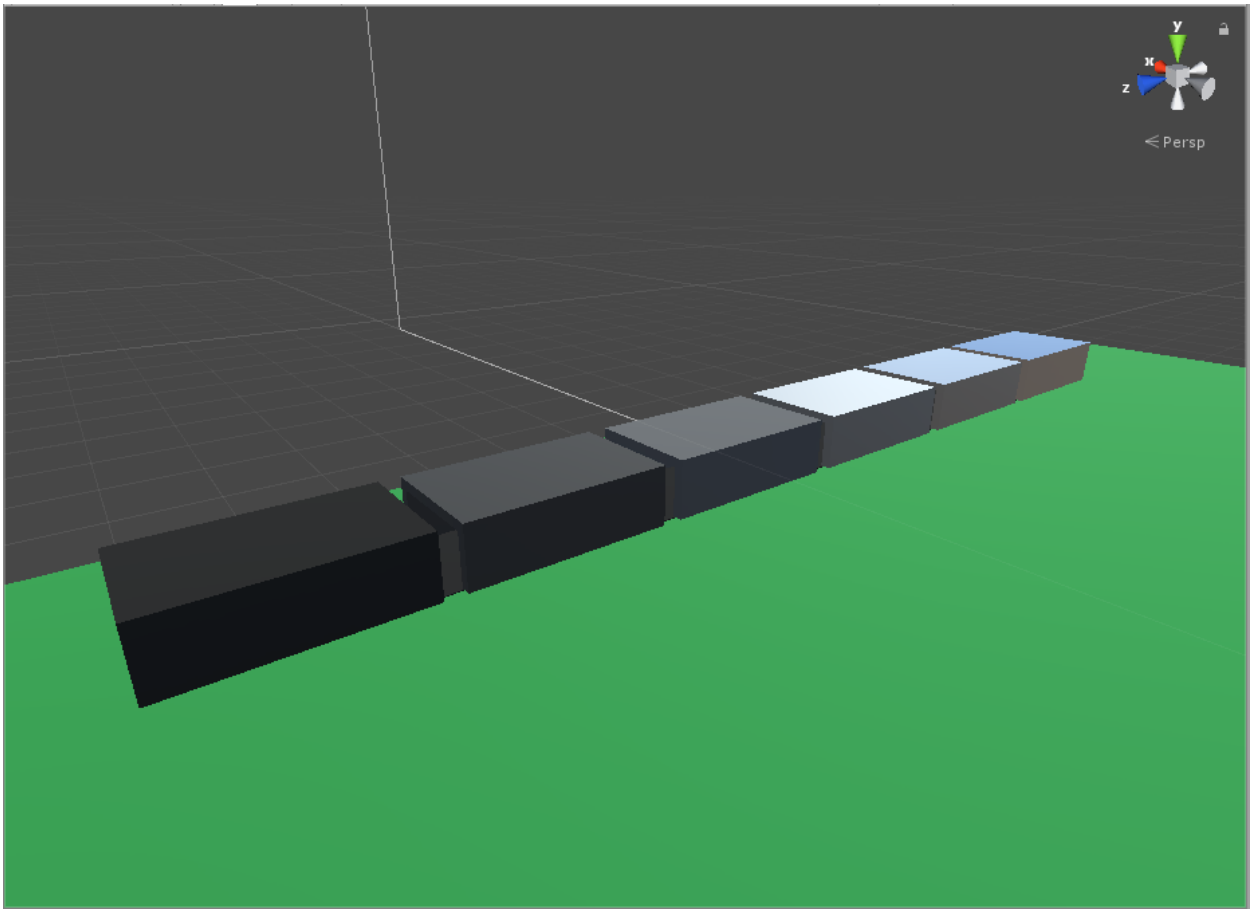
Section 2a - Material Design

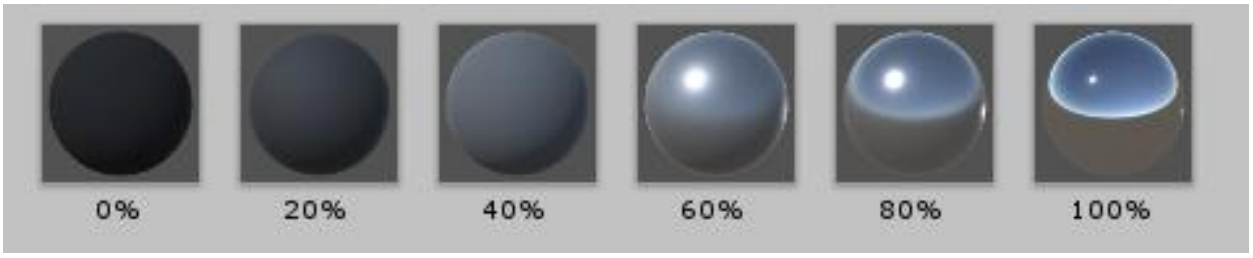
With a raycasting system in place that calculated for distance, the next stage of reading environment information could be tested. When a raycast collides with an object, it can gain access to the other components present on that object. As mentioned earlier, visible objects in Unity have renderers that house materials. The first step was to create a collection of materials for use with these renderers that house various values of smoothness and metallicness. For testing purposes, extreme values were used to make the resultant filtering more noticeable.



Section 2b - Reading and Interpreting Material Properties

With the materials created, they could be applied to various environment objects. An efficient method for testing this study was to apply materials with a gradient of parameters in a tunnel-like system. This allowed for the player prototype to wander down the tunnel and experience the influence of higher and lower smoothness and metallicness values as they changed.





Initial results from this were correct in their implementation, however, did not provide tonal qualities for the reverb. The smoothness value worked sufficiently; it was mapped to the reflections level of the reverb filter and, dependent on the smoothness of a surface, provided reflections higher or lower in level. However, issues arose from the mapping of metallicness.

Metallicness was mapped to both the room HF reference frequency and room HF level. At max value for both parameters (which initially occurred at metallicness value 1.0) an unnatural and brash reverb tone was produced. Though aesthetically this could be used, the filtering created would not naturally exist in a real-world location. This was understandable due to the extreme values gained from the materials, so scaling functions were added to reduce the prominence of changes. This produced a less noticeable - though debatably more pleasant – reverb characteristic from the filtering.

Prototype Discussion:

Adding influence from material properties proved successful in better representing the visual aesthetic of in-game environments as reflective and metallic locations now added a reflective and metallic quality to the reverb filter. The tunnel example showed the ARS working successfully in a context with changing environments and allowed also for environments to have their materials changed without needing to change the parameters of the reverb filter. This is helpful in suggesting additional symbiotic benefits between graphics and audio designers, as with expansion of the materials and the information they provide, further information such as object density, absorbency of parallel real-world materials and the resultant EQ filtering could be obtained and mapped to the relevant parameters of the filter.

Prototype 3 - System Refinements

Section 3a: Scaling Parameters for Less Extreme Results

Now that material values were being fed to the filter, parameter refinement was undertaken. To improve the workload undertaken by the GPU, each raycast had its distance capped. Particular interest was given to the raycast in front, since it represents the raycast most similar to the player's line-of-sight. Since sight provides significant amounts of information, it seemed logical to give the forward cast a higher priority of influence over the resultant values on the filter parameters.

Section 3b - Altering the Filter Signal Path

A final issue that arose in this case study came with the addition of non-player sounds. Since the reverb filter was a component on the player, only sounds that originated on the player were affected by the ARS. This was fixed by the creation of an audio mixer with a reverb filter send separate from the player object. Similar to how this process works in a DAW, rather than using the reverb filter as an inserted effect, a single filter is placed on the master output channel. This filter then processes all outgoing sounds within the DAW. In the case of the game, all sounds are processed by the current set of parameters the ARS is receiving. In order for this to be possible, the parameters of the reverb filter need to be exposed to scripting.

Prototyping Discussion

The refinement stages at this point made the ARS sound more appropriate to the visual stimuli on screen and also more accommodating to sounds not originating from the player. Problems arise by using a master reverb filter as sounds should be affected by the characteristics of the room they originate in rather than the current room the ARS is in. This would be more of a quality-of-life improvement however and is not a priority in current testing.

Further Study - Testing Environment Density and Issues with Optimisation

A small study experimenting with testing the amount of objects in an area was completed. This used raycasting to register every time a new collision was detected and keep a tally of objects in an area. This could then provide the ARS with influencing values for the level of reflections and dampening that could occur. However, keeping record of various information from multiple objects using raycasting caused high loads of work for the GPU due the constant collision detections. It may be of interest to explore using a large spherical collider rather than additional raycasts to detect collisions to reduce GPU workloads and provide a more inclusive

area for detections. This would provide better data from a populated level; this could be an extension of a functioning version of the ARS when applied to a built video game.

Adaptive Reverb System Design

With research and studies in place, the final design for the ARS can be completed.

The core mechanic behind the system is to use intervallic raycasting to capture information about a 3D virtual environment. In turn, this information will be decoded and applied to a reverberation filter that can be accessed at any time. To better match the characteristics of the environment, object renderer properties will be accessed to provide information about the physical properties that make up the environment. By calling the raycast system regularly, the reverberation filter will update itself as the player navigates a level and adapt to any changes it finds in the environment.

As aforementioned, the system will be designed, built and tested using the Unity games engine, with no external or third-party libraries or plugins.

Adaptive Reverb System Flow of Execution

Outlined below is an overarching description of the ARS and its flow of execution:

-Raycasts are regularly fired out from the location of the listener.

Unity houses a method called 'InvokeRepeating'. This method can be called in the Start method and will call a user-given method at regular intervals thereafter.

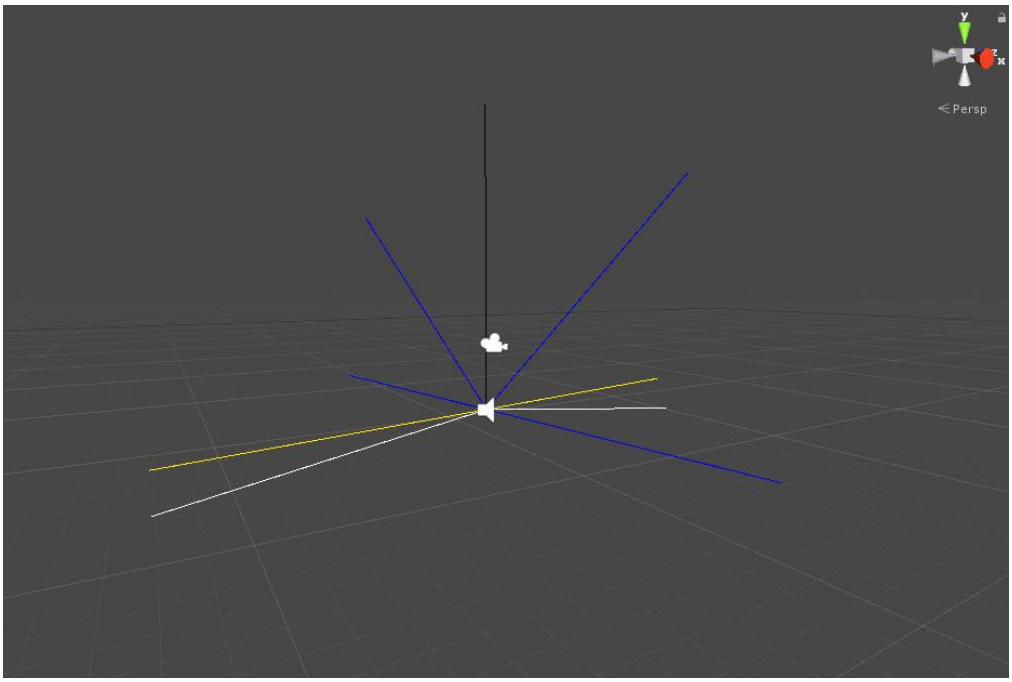


FIGURE 6 – SHOWING RAYCASTING IN A 3D ENVIRONMENT. RAYS ARE FIRED FROM A CENTRAL SOURCE AND PROVIDE INFORMATION ON OBJECTS THEY COLLIDE WITH.

From the Unity API:

"Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds."

A further method is created that fires raycasts in multiple directions from a given source point. The distance result from each raycast is then stored in an array. This method is used as an argument in the InvokeRepeating method, so as to fire raycasts at a user-provided rate.

-The raycasts collide with objects in the surrounding environment.

The raycasts are fired and eventually collide with an object. They calculate the distance in which they have travelled, and this information is then stored in an array. The raycast distances are then averaged to create a mean distance. This figure is then mapped to the reverberation filter's 'Decay Time' parameter.

-If multiple raycasts discover they are outside, the reverb is lessened.

Each raycast also checks the tag of its collision. An objects 'tag' is an identifier than can be set by the user to name an object in a particular category. If the collision tag is that of an outer boundary, then the reverb characteristics are nullified to better represent an outdoor space. If this check is not performed, then the resultant reverb characteristics sounded large and cavernous, which does not universally represent the sound of an outdoor space.

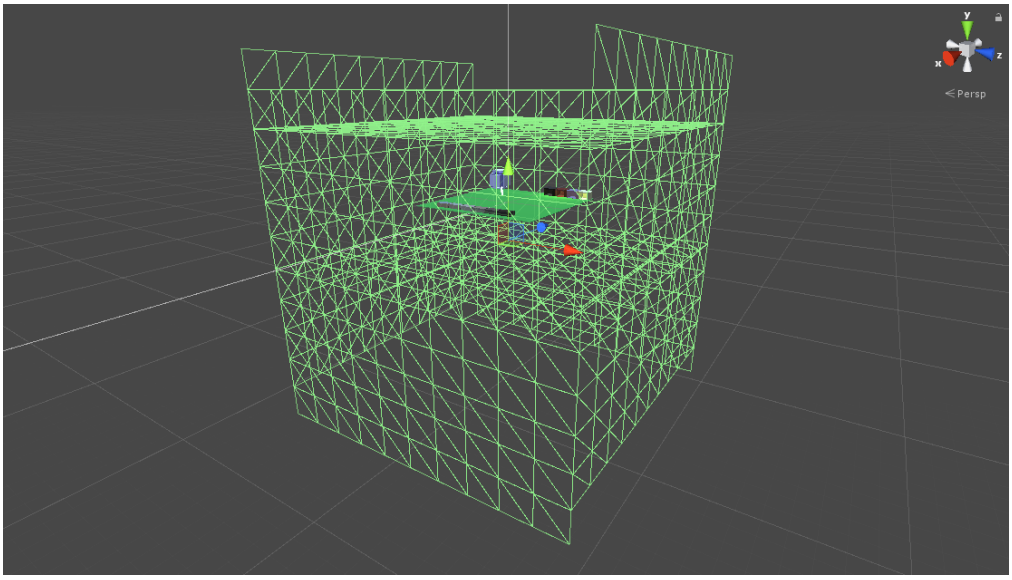


FIGURE 7 - OUTER BOUNDARIES ARE COMPLETED TO CATCH 'INFINITE' RAYCASTS THAT OTHERWISE WOULD NEVER EXPERIENCE A COLLISION.

-The raycasts access the renderer information of each object they collide with.

If the raycast returns having collided with a valid object, the objects Renderer properties are accessed and stored in a separate array. An objects Renderer allows for the object to be seen with a material, and also houses values for physical properties about that object. Specifically, the 'Smoothness' and 'Metallicness' parameters are read. These values are averaged to create a mean value, and then mapped to the 'Reflections Level' and 'HF Reference' respectively.

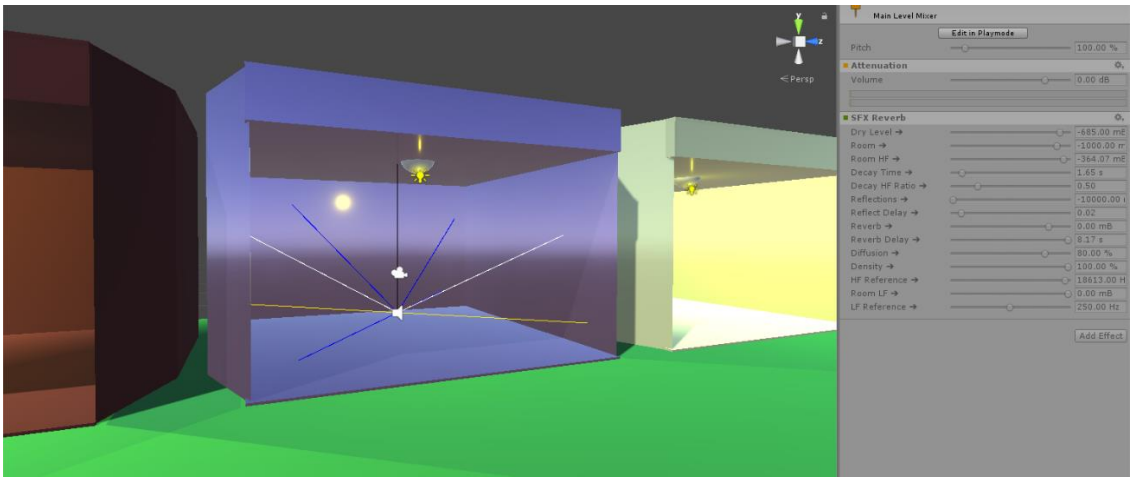


FIGURE 8 - SMOOTH AND METALLIC ROOMS PROVIDE THE REVERB FILTER WITH VALUES THAT REPRESENT THEIR VISUAL AESTHETIC.

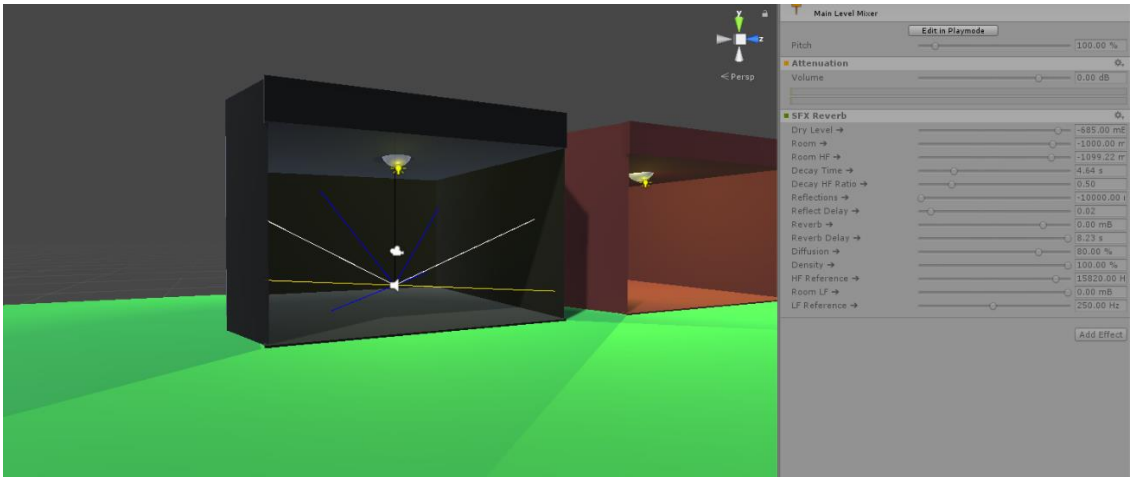


FIGURE 9 - SIMILAR TO FIGURE 6, THIS ROOM IS DULL AND NON-SMOOTH, AND ADJUSTS THE REVERB FILTER AS SUCH TO REPRESENT ITS APPEARANCE.

This entire process is invoked every 0.1 seconds. This value was chosen to allow for regular updating that could update fast enough to match a player's movement, however, not too quickly to be too computationally-demanding for a device.

Testing and Evaluation

Testing the ARS against Reverb Zones

Summatively, the concept of an ARS shows a significant improvement to increasing the efficiency of reverb addition in game development, however currently with a detriment to reverb aesthetic. Videos 1 and 2 on the supplied USB drive demonstrate the implementation and use of RZs and the ARS, respectively.

Time Taken to Implement

Overall, the time taken to implement RZ's was longer due to the time it takes to position them in a level. **VIDEO ONE** shows the process of adding zones to a pre-made level. As shown in the video, RZs are spherical. Though this wasn't too much of an issue for boxed rooms, convex or concave rooms would be much harder to implement RZ's for, as multiple polygonal shapes would be needed to perfectly cover the volume of the room.

VIDEO TWO demonstrates using the ARS. Room size, shape or implementation were not an issue as the system worked this out at runtime. With environment objects set up prior to adding audio, all that was needed is to place the script on the player. Since boundaries were already in place due to object colliders (thanks to the level designer/physics designer) and the textures were pre-mapped with physical properties (thanks to the graphics designer), implementation was very simple, only requiring tweaks in script to set an appropriate tone for filter parameter values when the player is 'outdoors'. Also, **VIDEO 3** shows how the ARS adapts to changing zones. An RZ would be required for each stage new area meaning that walking through an area such as a changing corridor would require multiple RZ's with independent parameter snapshots. The ARS deals with this change automatically.

Summary:

RZ's are simple to add however require customisation to allow them to fit spaces easily. This process does not take such a long time that it becomes an issue, however, does require more time for accurate placement and the crafting of user-generated reverb filter snapshots for each instance of an RZ. The ARS simply needs to be placed on the player and given reference to an appropriately named, exposed reverb filter on the master mixer. Once this is in place, the environment controls the snapshot parameters that are currently hooked up, with the leftover parameters being hard-coded as to whether the player is indoors or outdoors.

Reverb Characteristics

In its current state, the ARS provides reverb based on its surroundings, however the characteristics of the provided filtering are not as refined to match the environment as is desired. Given, the environments in the video demonstrations are only visualised with colours and smoothness/metallicness, thus do not offer much context as to the material they represent. However, when a material is smoother and more metallic, there is a noticeable difference in the reverb timbre that corresponds with the surrounding materials; a smooth and metallic space offers filtering that is dense with higher frequencies, and vice versa.

RZs are more representative of the spaces they are in at this time. Since each parameter is set pre-runtime and does not change, reverb can be crafted prior to match set each parameter specific to how a material would react in the real world. Though this process is lengthier, it provides a better filtering result. Furthermore, RZ's have distance roll-off functionality built in. This means that transitions between different reverb snapshots is smoother. The issue with the ARS is that the current state checks for environment changes every 0.1 seconds. If two areas have vastly different characteristics (such as a concert hall to a dry, padded room) then the change between is glitchy as the parameters must jump large distances to adjust for the varying environments. A fix for this would be to provide interpolation between values (parameter smoothing) for less sudden changes. This was more prominent when the reverb filter was in series on the player game object as sounds could have their reverb filtering changed while still playing. Moving the filter to a parallel send fixed this issue for the most part, but parameter smoothing would create a more seamless transition between varying areas.

Summary:

With further exploration in how to interpret environment data and match it to its corresponding reverb filter parameters, the ARS could achieve the same results as RZ's do currently. Since RZ's can be hand-crafted, they offer better scope for representing real-world materials. This said, if a graphics designer created materials with further information that corresponded to a currently-unused reverb parameter, this problem would be fixed. Additionally, smoothing between parameters would solve glitching audio outputs as samples would be filtered with gradual changes rather than large, sudden changes.

Assessing the ARS

Raycasting

Raycasting as a basis for a reverberation system has the potential to work successfully with further expansion. In its current state, the mean value calculation works in creating a representation of a rooms size. One problem posed by this, however, is when a room has an anomalous value for distance, such as a small room with a doorway to a large space. A workaround for this would be to assess each value and for each set of raycasts and calculate a maximum value. If a value from a raycast at that time exceeds the threshold, then that value would be capped/scaled to create a more intelligent average.

Object Reading

Reading values from objects minimises the additional work needed between development teams. As mentioned previously, by having graphics designers give objects physical properties based on real-world examples, an audio designer can read these values and map them to the parameters of the reverb filter. An issue faced here is how to appropriately translate object characteristics to reverb filter parameters. It would be useful to create a standard on the mapping of physical object properties to the parameters of an I3DL2-complient reverb unit for future use. Furthermore, currently the ARS has parameters with no dynamic mapping, instead being statically initialised when the game is launched. This standard would make the process of mapping more streamlined for better game design efficiency.

Discussion

The system shows potential for an ARS in video games. The current system works in prototyped levels however is yet to be tested in a dedicated video game. A logical next step would be to test this system in a full game environment to better gauge the performance when a level is fully populated.

Further Implementations

Completing the project to its current state poses further areas for exploration. These are outlined below:

Immediate Goals

To prove the functionality of the ARS, it would be helpful to provide a playable level with further gameplay features. This way, the system could be analysed in context to see its reaction to more populated levels and also its performance with additional sound effects, physics calculations and graphics rendering happening simultaneously. A basic game will be created in preparation for the Viva examination to demonstrate this.

Designing a Custom Reverberation Filter

Though the Unity reverb filter is sufficient for providing appropriate filtering, an all-inclusive system that houses both the raycasting and filtering could be created to further streamline the process. The current Unity Audio Engine does not allow for symbiotic use of DSP and rendering (raycasting) so an expansion of the engine, though possible, is unlikely for the time being. Instead, it may be helpful to expand the current reverb filter so that it better informs novice users on appropriate mappings.

Occlusion

One key element in psychoacoustics is the correlation of visual perception to audio perception. To expand on the ARS, simple occlusion systems could be incorporated. Occlusion in video games explores the characters line-of-sight in relation to elements of their environment. In an audio context, a player could be in one room with a sound being emitted from an adjacent room. Though the sound would be audible, the sound is occluded by items such as walls. By calculating whether a player (therefore the listener) can see the sound source or not, further EQ filtering can be provided to create the impression of a sound being nearby, however in another space. Consider standing outside of a music venue while a concert is happening inside. Though the sound is audible, the layers of environment between the listener and the source provide filtering to the sound. Resultantly, the sound becomes muffled in quality, which the listener would associate with being in another space. This could be implemented to the ARS with further raycasting that represents the players line-of-sight. If a sound source can be seen by the player, then no filtering is applied. If the source is occluded, then filtering is applied.

Application to Virtual Reality

An interesting proposed use for the ARS is in VR. VR brings about new levels of immersion in video games by commandeering a player's field of view and places them into environments through VR headsets. Such headsets track a player's real-world rotation and as such could be used to include additional features such as orientation and visual cues in the ARS. Much of a person's auditory information can be enhanced or clarified through visual stimuli also (ADD A REFERENCE HERE). By including such stimuli, the reverb's fidelity can be perceptually improved as the player feels as if they are in the environment that the reverb filter is affecting.

Summary

The extensions provide above pose appropriate implementations to be added to the ARS. These will be explored additionally to this dissertation and implemented into future versions of the system. The resultant goal for the ARS would be to have it function in a fully-fledged 3D video game as a replacement for standard RZs. Though this goal is ambitious for a single person, should a small team be assembled to create such a project, the adaptivity of the ARS would make for an easy implementation into future projects should the required standards for the system be followed.

Conclusion

The concept of an Adaptive Reverb System provides a plethora of solutions to current issues in video game audio. The literature explored reinforces the necessity for exploring technical audio systems in video games due to a current lack of industry innovation in this area. Furthermore, the literature includes extended research that can be used to inform better audio system design for future projects. The research provided a foundation for the proposed ARS to be built from and also brings forth additional areas for research to improve the quality represented by the current system. In its current state, the ARS functions as a dynamically-adapting reverberation unit that can easily be applied to typical video game scenarios. Additionally, the system improves upon current game design workflow practices by offering a symbiotic system that allows varying design teams to create assets solely in their area. Combined, these assets are interpreted by the ARS with minimal adjustment needed, reducing the need for cross conflicts between design parties.

Many issues arose from this project. Appropriate mapping was a prominent issue that requires further research in order to execute it properly. Without a standard in place, literature is limited, and requires the translation of real-world physics principles into the digital realm. Though basic workarounds are provided in this project, these could be improved through further dedicated research and documentation.

Overall, the ARS successfully provides an intelligent workaround for current reverberation practices in the context of video games, and documents further improvements to be made that could challenge conventions in video game audio and bring about new standards for adaptive audio systems in video game engines.

Project Reflection

This project served as a transition from audio to game design, aiming to teach about fundamentals of game development in Unity, and the applications of audio to interactive media. It achieved in teaching the basics of prototyping and design using Unity and C# scripting (where there was little prior knowledge) and has opened up many pathways to extend this knowledge in subject areas relevant to the topic at hand. Furthermore, it has sparked an interest into other design areas separate for audio, and more so how these areas could be influenced by game audio to create new and exciting methods for developing game mechanics; such as using audio as a controller or using live audio input at runtime to determine in-game action.

Managing this project proved difficult due to its scope varying on a few occasions. Though the topic overarchingly stayed consistent, developing a clear narrative was problematic as there were many points up for consideration. This was beneficial in providing various doorways into new topics, but these discussions often fell short of being in-depth, more so offering ways into deeper exploration. It wasn't until later into the project that a solid narrative was carved into the write-up that a better flow of information. This was counterintuitive to the original Gantt proposal, also. Since the scope did change, the Gantt chart underwent a plethora of changes also. Sectioning out parts was a beneficial addition, however the nature of a developing project meant that new information could present itself and offer a new way to see the project through a previously unknown lens. It would have been useful to cater for this by developing a modular Gantt chart that housed generalised areas and milestones that could be substituted to accommodate for changes in direction.

The poster presentation helped inform a final narrative and direction however, as distinguished industry members provided crucial insight and avenues that would be of the most relevancy to explore, whilst isolating areas that were perhaps beyond the necessary scope of the project. Furthermore, discussing the project with those unfamiliar with the concepts of reverb and game design allowed for the knowledge gained from this project to be better understood. The benefits of learning-through-teaching and retrieval practice have been studied intently (Koh, Lee and Lim, 2018) and in the case of this project, explaining the information learned and presented in the background research, especially to those unfamiliar with the subject area, proved useful in developing a lasting understanding of the final project.

Summatively, the project has been both successful in introducing the fundamental concepts of game design and how audio can be included in this, with this cumulatively resulting in an adaptive reverb system that functions as intended, albeit requiring further work. As a result of this, a passion for furthering this knowledge into the creation of a small 3D video game has been found and this will be started over the summer of 2019 with a small group of other UWE students.

References

Atlassian. (n.d.). *Agile best practices and tutorials | Atlassian*.

Atlassian. (n.d.). *Sprints | Atlassian*. [online]

Blizzard Entertainment. (2016). *Overwatch*. Blizzard Entertainment.

Clinton, K. (2006). Agile Methodology in Game Development: Year 3. In: *Game Developers Conference*. [online]

Cooper, S. and Bullock, B. (2014). *ALIEN: ISOLATION AUDIO INTERVIEW WITH SAM COOPER & BYRON BULLOCK!*.

Creative Assembly. (2014). *Alien: Isolation*. Sega.

Epic Games. (2019). *Proxy Geometry Overview*. [online]

Ferrara, F. (2018). Agile Game Development process for Indies. [Blog] *Gamasutra*.

Fidelity. (2013). In: *Cambridge Advanced Learner's Dictionary and Thesaurus*, 4th ed. Cambridge: Cambridge University Press.

Gearbox Software. (2012). *Borderlands 2*. 2K Games.

Harteveld, C. and Kortmann, R. (2009). Triadic Games Design workshop. In: *ISAGA 2009*. [online]

Holman, T. (2010). *Sound for Film and Television*. 3rd ed. Elsevier Inc, p.164.

Interactive 3D Audio Rendering Guidelines. (1999). 2nd ed. [ebook] Los Angeles, California: MIDI Manufacturers Association.

Jørgensen, K. (2009). *A comprehensive study of sound in computer games*. Lewiston, NY: Mellen.

Koh, A., Lee, S. and Lim, S. (2018). The learning benefits of teaching: A retrieval practice hypothesis. *Applied Cognitive Psychology*, [online] 32(3).

Kortmann, R. and Harteveld, C. (2019). Agile game development: Lessons learned from software engineering. In: *ISAGA 2009*. [online] Singapore, pp.5, 6, 7.

Larsson, P., Västfjäll, D. and Kleiner, M. (n.d.). *AUDITORY-VISUAL INTERACTION IN REAL AND VIRTUAL ROOMS*. Gothenberg: Chalmers University of Technology.

Lawlor, S. and Neumann, T. (2016). Overwatch - The Elusive Goal: Play by Sound. In: *Game Developers Conference*. [online]

Mo, R., Wu, B. and Horner, A. (2016). The Effects of Reverberation on the Emotional Characteristics of Musical Instruments. *Journal of the Audio Engineering Society*, 63(12).

Prototyping Pack (Free). (2017). Germany: Digital Konstrukt.

Roth, S. (1982). Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18(2), pp.109-144.

Sandberg, S., Calle Håkansson, C., Elmqvist, N., Tsigas, P. and Chen, F. (n.d). *USING 3D AUDIO GUIDANCE TO LOCATE INDOOR STATIC OBJECTS*. 5th ed. [ebook] Gothenberg, Sweden: Chalmers University of Technology.

Spatialisation. (2018). In: *Collins English Dictionary Complete and Unabridged.* 13th ed. Glasgow: HarperCollins.

The Elder Scrolls IV: Oblivion. (2006). Bethesda Softworks, 2K Games.

UK Video Games Fact Sheet. (2019). [ebook] London, England: The Association for UK Interactive Entertainment, p.3.

Unity Technologies (2018). *Unity - Manual: Unity User Manual (2018.3)*. [online] Docs.unity3d.com.

Unity. (2019). Unity Technologies.

Valimaki, V., Parker, J., Savioja, L., Smith, J. and Abel, J. (2012). Fifty Years of Artificial Reverberation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(5).

Bibliography

Baracskai, Z. (2010). *Portfolio of Compositions*. Ph.D. University of Birmingham.

Gallacher, N. (2013). Game audio — an investigation into the effect of audio on player immersion. *The Computer Games Journal*, 2(2).

Jørgensen, K. (2007). On transdiegetic sounds in computer games. *Northern Lights: Film & Media Studies Yearbook*, 5(1), pp.105-117.

Prakash, E., Brindle, G., Jones, K., Zhou, S., Chaudhari, N. and Wong, K. (2009). Advances in Games Technology: Software, Models, and Intelligence. *Simulation & Gaming*, 40(6), pp.752-801.

Tsingos, N., Gallo, E. and Drettakis, G. (2004). Perceptual audio rendering of complex virtual environments. *ACM Transactions on Graphics*, 23(3).

Välimäki, V., Parker, J., Smith, J., Savioja, L. and Abel, J. (2016). More Than 50 Years of Artificial Reverberation. In: *AES 60TH CONFERENCE ON DEREVERBERATION AND REVERBERATION OF AUDIO, MUSIC, AND SPEECH*. AES.