Chapter 15
Making Sound Decisions in Game Audio

Gwen Guo

CONTENTS

15.1 What Is Game Audio? 287
15.2 Where Is Game Audio in the Production Pipeline? 288
15.3 Elements of Game Audio 290
   15.3.1 Music 290
   15.3.2 Sound Design 291
   15.3.3 Voice 293
   15.3.4 Implementation 295
   15.3.5 Examples of Game Audio Implementation 298
Ludography 301

15.1 WHAT IS GAME AUDIO?

Ever found yourself humming to a memorable music riff in-game or found yourself being unable to play first-person shooters (FPS) with the audio muted? In the discipline of game audio, it is both creative and technical, and both an art and for functionality. There are three common objectives for audio in games – to provide non-visual aesthetics, to build the world and to provide feedback to the player. Many other games have included audio as part of their core gameplay design, not limited to rhythm games.

The main difference between game audio and linear audio for films, TV and podcasts is its interactivity; if one watches a film, both the audio and visuals will play out exactly as intended at the exact timecode it was told to play at. In games, however, multiple permutations of player actions can
affect its journey, thus also directly affecting the way audio behaves as part of this journey.

15.2 WHERE IS GAME AUDIO IN THE PRODUCTION PIPELINE?

Whether the audio person is based in-house or outsourced, it is recommended that they enter the conversation from pre-production. A common saying goes in the film industry, ‘Sound is 50% of the film’. In the case of games development, we can safely say that it is 33.33% of games alongside visuals and gameplay. As generalized as it sounds, the main point of this quote is to emphasize how sound is as essential as visuals and is just as capable of conveying meaning, narrative and world-building. To some, this is stating the obvious, yet many game studios still only involve sound in the middle of the production cycle, or worse, at the end.

To understand why the audio team needs to be involved early, we must first understand what exactly goes into audio pre-production (Figure 15.1).

As with all creative disciplines, time needs to be allocated for the conceptualization of audio direction, whether it’s music or sound design. This could come in the form of designing sound to a linear screen capture of the game or a game design document. References can come from either the audio person or developers if they have something specific in mind; sharing audio clips and videos with audio is often clearer than describing sounds using words as the interpretation of mere words can often be subjective. A strong, solid foundation of audio direction will ensure that the game sounds cohesive throughout.

The audio department must also contribute to the discussion if the project needs additional tools such as audio middleware or engine plugins.
and it’s essential that their perspectives play a major role in making the purchase decision. Creating customized audio tools will be necessary with or without audio middleware. As different projects have their own unique tech structure, the audio department would need to have constant discussions with the programmers and designers on customizing these audio tools starting from pre-production. This will vastly improve the work pipeline as the team journeys deeper into production.

If the audio department has an audio director, the latter’s responsibility is to ensure that the audio team has adequate resources, tools and information from the development team to do the work, lead creative ideation and brainstorming (this does not mean that ideas come solely from the audio director), solidify creative direction, look out for their co-workers’ well-being and mentor their co-workers if needed. They will also be the main liaison for game developers to speak with as a representative of the audio department.

With adequate time given to the audio pre-production, the team can also more accurately scope and negotiate for the timeline in conjunction with the entire game development timeline and therefore cost for budget, effort and time. Some larger game projects assist with the scope by having the production team work on a vertical slice. This is an ideal milestone and structure for the audio team to best put their creative concepts to the test as well as iron out any pipeline-related kinks which could arise along the way, ensuring a smoother experience during the production phase for the main game later on.

An example of a game audio production pipeline is shown in Figure 15.2.

After pre-production, the audio department can dive deep into production. Figure 15.2 illustrates a typical production pipeline within the audio department. Production is only a portion of the audio department’s responsibilities – implementation and play testing are another huge part in the development cycle.

Depending on budget and the size of the company or game, some game studios may delegate the implementation task to a programmer from the development team. If so, it’s recommended that the audio department at minimum has access to a build where they’re able to hear the game in its entirety and suggest implementation improvements.

The audio department needs to have ownership of the game – having access to the build is essential to ensuring that the audio is at its best quality. Any sound asset created from production, no matter how great by itself, will not shine in-game if the implementation is poorly done.
15.3 ELEMENTS OF GAME AUDIO

15.3.1 Music

There are two main ways of composing game music: using software instruments and live recordings.

Software instruments are plugins which can be used within a digital audio workstation (DAW) such as Logic, Cubase or Reaper. While recordings manifest as waveforms in these DAWs, software instruments follow the MIDI notation format, or for specific DAWs such as Sibelius, music notation.

Samplers and synthesizers are the two main types of software instruments. Synthesizers mimic the capabilities of analogue synths, manipulating basic tones such as sine, square, sawtooth and white noise, subsequently running them through oscillators and editing their attack, decay, sustain, release (ADSR). After manipulation, the composer obtains their desired instrument. Synthesizers typically produce sounds which can be found in electronic genres of music.

Samplers contain pre-recorded waveforms which are triggered by MIDI – they can be looped or played as a one-shot sample, depending

FIGURE 15.2 Audio production pipeline. (© Imba Interactive. With permission)
on the instrument. Some samplers allow for more variety and advanced features. For example, a violin sampler can have different common violin expressions such as legato, pizzicato, tremolo, etc. Samplers can be used for all genres of music from orchestral to hip hop.

Although samples are able to achieve some degree of expression, no sampler can accurately mimic a live sessionist, especially if it’s a solo instrument. In a world where it still takes a considerable amount of effort to get software instruments sounding exactly like an acoustic instrument, this is where live recordings come in – live recordings are used when the song calls for a more naturally expressive performance than software can ever perform. However, live recordings require a professional recording space with decent microphones, a recording engineer and a seasoned sessionist. That being said, the result is almost always better than attempting the same genre and expression using software instruments.

Most game composers use a combination of both software and live recordings in their arsenal of wonderful composition tools. Many composers would argue that it’s often not just about the tools; building an identity with their music is often valued more than the gear and tools used.

In the context of games, building a brand and thematic identity for the project is one of its core functions, alongside the ability to influence how the player should feel. A great example would be the music of Thumper, with its music described as ‘rhythm violence’ – it is aggressive, intense and gripping. Even as a standalone OST (Original Sound Track), listeners are immediately able to discern the theme of the game that this music is written for.

Composing for interactive music requires composers to think like live mixing from a DJ, except that the player of the game is the DJ while composers provide the intro, outro, loops and stingers. The composer will have a big picture of the arrangement of the track, but ultimately the player actions will determine when the music transitions to its next intended destination.

15.3.2 Sound Design

Similar to the methods of production used in music, sound design also involves both synthesis and live recording, but they differ in materials being recorded.

Based on the introduction to synthesis written in the music section, sound designers may use synthesis to produce and manipulate basic electronic tones to their desired complexity. Some may choose to work on
plugins running on DAWs or visual programming software such as Pure Data, MaxMSP or Kyma (these can also be used for music). Essentially, the sounds are moulded by numerical and mathematical data, although they are presented in graphical format. An example of a sound synthesiser which uses purely algorithmic functions even in its presentation is Supercollider. The audio generated from these synths used in the context of sound design are typically used for non-organic ambiences and sound effects such as for the sci-fi genre of games, digital user interface (UI) sounds or magical effects. Organic sounds which can be mimicked using synthesis are sounds which are similar to white noise and its variants such as rain, wind, whooshes and simple impacts.

Pertaining to organic sounds, sound designers will most likely record the sounds using a microphone and sound interface setup – the two main types of live recording for sound effects are foley recording and field recording.

Foley recording is popular in film sound where the foley artist performs in sync to picture while the sound engineer records. In the case of game audio, foley artists sync to a recorded screen capture of the game animation, but for delivery format, the recording is separated into single one-shot sounds for implementation. Foley artists have a range of props and tools to help them create the sounds they need. The point of foley recording is to only capture a single source of sound in isolation; they’re recorded in a controlled space – a room which is heavily padded to prevent reflections and is silent. Examples of foley recordings in the context of game audio include footsteps on various surfaces, cloth movement, water splashes, physical object impacts and whooshes.

Field recording is almost exclusively used as ambiences or sound effects recorded outdoors if it’s quiet enough; if the prop is based outdoors or is too large to fit into the controlled space (i.e. a pool of water, large rock). Due to the nature of audio implementation, it’s important for the recordist to have an initial vision of what they’d like the final product to sound like in-game, then break them down into layers – those layers will be recorded separately. To illustrate, imagine that the player is standing in the middle of an open field – they hear a cacophony of birds constantly chirping, some wind rustling through the grass, occasionally a bee would fly past the player and a distant waterfall. The recordist will record each of these elements separately – a recording of birds chirping WITHOUT any other sounds including wind, JUST the sound of grass rustling in the wind, a bee buzzing in isolation and finally just a waterfall.
Recording these elements separately provides the audio integrator more flexibility when mixing for the game. The integrator will be able to pan the bee buzz from left to right, or right to left however they want without affecting the panning of any backing track. They will also be able to place a point source of the waterfall at where it should be – the waterfall model in the game engine. About to enter an ominous situation? Simply attenuate the birds and keep the grass rustling.

As with music, sound designers often use a combination of synthesis and physically recorded sources to create their final asset. Often, there are also musical elements within a sound effect, so the principles of music can also be applied to sound design.

It’s also common practice to use sample libraries to design a sound the same way a painter uses a palette of paint. Sample libraries are pre-recorded material which can be bought online or accumulated by the sound designer over years of recording. There is absolutely no shame in using sample library sounds as a layer for your sound creation. There are a couple of things the sound designer needs to take note of: the licensing model for assets obtained from the sound library and the overall originality of their work. Where relevant, the sound designer may not want their sounds to resemble anything else out there in the market and also achieve a unique identity for their game sounds, therefore sufficient processing and manipulation to the sample is recommended.

Sound design is not limited to simply playing the supporting role in games. It can achieve aural identity the same way music does. A collection of sound effects will be able to best convey the overall theme for the game rather than a single sound effect in isolation, yet there are examples where sound effects are memorable; the alert chime in the *Metal Gear Solid* series is unmistakable, or the ascending synth when Sonic spins before travelling.

Thinking like a game designer sets us apart from film sound designers – we predict player choices, prepare for differing permutations on top of aesthetic intention. This influences our work process from the way we record and compose our audio assets to how we export our files.

15.3.3 Voice

In the realm of indie game audio, voice is often regarded as a tertiary consideration in comparison with music and sound design. However, good voice acting can make or break a game. Whether it’s about using voice as a narrative element or creature design, the voice talent in question needs
to be suited for the function. Voice performance can’t be fixed in post, so it’s essential that team and/or their voice director finds the most suitable talent via voice casting.

For voice casting, several factors need to be considered. What’s the personality of the character in question? What circumstances are they faced with? What emotions and energy do they portray in the game? Is the voice talent able to achieve the intended performance naturally? Does the personality of the voice match that of the game character? Having voice talents submit an audition would be ideal in the casting process. A well-cast voice talent and a voice director with a clear vision will ensure a smoother recording as they’re able to perform with minimal revisions, maximizing the time and budget spent in the recording room, as well as representing the game character accurately.

Before entering the actual voice recording process, it’s important for the script to be confirmed and formatted for easy reading and marking. It’s not recommended to make changes to the script on the spot, or worse, change the script after the recording is done. Again, this is to avoid situations where the voice talent will have to re-record the lines which will incur costs in both time and money. Furthermore, the filenames should be consistent, especially when localization is involved. This will ensure an efficient process for the implementation of localized dialogue.

During the recording process, having a professional studio with a qualified sound engineer is ideal. A well-padded, quiet recording space and an engineer who’s able to listen out for audio discrepancies while organizing takes and files is well worth the money. However, for those on a budget and can’t consider a professional space, setting up a non-reverberant space in a quiet location could be an option. For example, a walk-in closet surrounded by plenty of clothes or a makeshift vocal recording booth covered by a heavy blanket. Unless having a reverberant voice recording is the intended effect, the voice recording should be as clean as possible. Remember that voice is produced with the physical body so do remember to hydrate and set a maximum time limit for each recording (recommended two to three hours per session); ensure that the voice talent is not overworked as it may affect the consistency of their voice, their morale and energy.

For gear, those on a tight budget should minimally consider a USB condenser microphone as it’s able to capture a good dynamic range from the voice without it sounding too washed out or tinny. Overall, a good, clean recording will minimize the time needed for clean-up later on. As the famous saying in film sound goes, ‘Not everything can be fixed in post’.
In reference to the previous statement, there are some things which can be fixed in post. Light, consistent background noise can be fixed in post. If you’re recording in a makeshift space, that space may pick up low industrial hum or noise wash from a distant air conditioner. The voice editor can then use denoiser hardware or software in their DAW to remove the noise. Examples include the Izotope RX series or Cedar products. However, a noise floor which is too high will be tedious or impossible to be removed, even with the best tools; an attempt to do so in a noisy recording may result in a voice recording that has missing frequencies or artifacting. Dialogue levelling ensures consistent dialogue volume levels where a whisper can be audible in-game and a scream doesn’t hurt the ears. Mouth clicks are a bane in dialogue editing, but they would have to be edited out nonetheless. Some popular methods which sound engineers recommend voice talents who often have unintended mouth noises is to eat an apple or suck a Tums tablet before recording.

Generally, the work required in post-production for dialogue can be minimized if adequate preparation is done before recording takes place; from good microphones, suitable voice talents and a decent recording venue to finalizing the script before recording and having a streamlined voice and localization pipeline. Once you have your clean, levelled voice recording, you can use plugins to manipulate the voice if the character is not entirely human to enhance the identity of your character.

The addition of voice to your game project can hugely add to your game theme and identity. In narrative-heavy games such as games from Bioware, a great voice performance is essential to player immersion. Competitive games such as Quake and DOTA hugely benefit from voice as players gain immense satisfaction whenever they receive vocal feedback for their achievements such as ‘Double Kill!’ Overwatch not only uses voice to give their characters deeper personalities, their voices are also important for player and enemy feedback. Even casual games such as Candy Crush benefit from celebratory vocal gestures.

15.3.4 Implementation

Determining the behaviour of sound playback in-game is a core role of implementation and also an integral part of game audio in general. Closely intertwined with game design and programming, good audio implementation ensures that all possible scenarios in the game are accounted for. Something as simple as footsteps involve implementation, and can also be as complex as you’d like, depending on the level of detail.
In many games, footsteps are a sound effect you’d hear almost throughout the game, so it’s important that there are variations of footsteps to prevent it from sounding unnaturally monotonous. The sound designer and foley artist may provide a range of footsteps which cover different surface materials and different performances between run, walk and jump. The audio implementer ensures that these footsteps are randomized as they playback to prevent repetition, with the footprint assets changing based on material the player’s feet collide with in-game and also whether the player is running, walking or jumping. The sounds may even randomize in pitch or be separated according to heel and toe for greater variation.

Another example; it’s common in match-three puzzle games to play an ascending success tone if players consecutively achieve combo chains – the longer the combos are strung together, the higher the pitch of the sound plays up till a max success of combo chains. Similar to randomization, these sound assets are exported as variations, but they are played in a fixed sequence instead, so players develop familiarity that their success is growing with ascending notes.

Interactive music behaviour is also done by the audio implementer. Apart from simple crossfading between music tracks as a player travels between locations, exporting your music track as stems provides more flexibility for implementation. For example, when the player pummels a boss until its health reaches 20%, a one-shot stinger plays and a new drum loop plays perfectly in sync with the existing loop, informing the player that the boss is weak and it’s a crucial moment for the player to finally beat it.

When it comes to tools, most game engines already have a native audio engine – Unity 5 onwards have the audio mixer to be used in tandem with the audio sources, and Unreal Engine has its own powerful audio functions. If the audio implementer is well-versed in programming, or there is a dedicated audio programmer, or if the project requires a heavily customized audio solution, and adequate time has been factored into the game production pipeline to include the construction of audio tools, perhaps the audio team may choose to work with the native audio features of the game engine.

Another powerful solution for audio implementation is audio middleware. Popular tools such as Wwise, FMOD and Fabric are commonly used by both AAA and indie developers alike. They’re the key to efficient audio implementation without the audio implementer needing to directly code. These tools export audio events instead of audio files; within these events,
not only can the implementer determine an audio asset to play back, stop or pause, they can also set mix presets (volume changes), determine which effects to play and randomize as many sounds in a single event.

Although some audio implementers work exclusively on the audio middleware, having access to the game engine build is still essential. Most audio implementation can be best previewed and tested in the game engine itself, because the build takes 3D positioning, actual player response and gameplay triggers/collision boxes into consideration.

Three-dimensional positioning of a physical object within the game engine determines where the sound will play and how it transmits the signal within the 3D space. We will discuss two main factors in positional audio – volume and panning.

For example, an audio source will be attached to a large waterfall, but because it’s a large feature, it can’t just be a single point source – if the player is able to travel from the top to the bottom of the waterfall, multiple audio sources will need to be considered, or a spline that creates a single audio source following the player can also be implemented. Also, to simulate real life, an audio source with too extreme panning would be unnatural, so implementers will explore this audio feature called ‘spread’, which determines how directional the sound should be. An audio source from a 3D space would also attenuate accordingly in volume – the further you are from the object emitting sound, the softer it will be and vice versa.

In film sound, the overall mix is done at the final mix stage with a huge console of faders and knobs. The same can be applied to games, except that the mix is done in the audio middleware or game engine. The concept of volume faders still applies. Because player response is unpredictable and non-linear, it’s important to test out every possible scenario in the game to ensure that the audio plays back in a balanced manner. Similar to DJing, the volume controls of sounds in a game are controlled by player interaction and therefore code instead of a human DJ.

Apart from volume and panning, there are other more customized effects which can be used in various situations in the game project. Reverb will allow players to feel the space with sound – a footstep playing in a cathedral sounds very different from being in the bedroom. Typically for positional sounds, reverb implementation comes in the form of collision boxes – when the player enters the collider box, it will switch to a reverb preset. Sounds which aren’t positional and therefore won’t take its 3D position into consideration may have its effects either baked into the audio file or implemented within the event. Another popular effect would be the
low-pass filter, an effect which muffles sound when you’re underwater or behind a wall, to cite from many other examples.

A major feature which sets us apart from film sound is the need for optimization. For game audio, it’s always a tough balance between disk space and memory usage. An audio file which is compressed will require less disk space but add to the memory used to uncompress it, whereas audio which is uncompressed will require hardly any memory but will take up loads of disk space. A general rule of thumb is to export short but frequently triggered one-shot sound effects as uncompressed sounds while long, occasional sounds like ambiences and music be compressed, and, of course, there are exceptions.

The best practice for audio file compression is to take place within the audio middleware (if you’re using middleware) or engine. Therefore, audio assets should be exported in its uncompressed .wav or .aiff format as it’s more efficient and flexible to compress files later on than to bounce and re-export already compressed files, with the latter potentially leading to double work if the audio team realizes that it’s not the ideal format.

There are also additional loading options for audio, like streaming and preloading, enabling programmers and audio implementers to have more choices for memory optimization. Also be aware that optimization is different across all platforms – it’s important that the audio team gets a chance to test the game on all the platforms that it’s being released on.

15.3.5 Examples of Game Audio Implementation

For music implementation, there are several basic music transitions the audio implementer can explore – branching allows phrases to simply transition to the next start of the phrase. It is similar to the transitions of pop songs where a verse, chorus and bridge are discrete phrases. The implementer may choose to transition immediately or in the next bar/beat (Figure 15.3).

Crossfading is another common transition where the current track fades out the same time the incoming track fades in. The audio implementer may choose to define the duration of the crossfade and the types of fade curves – it could be a straight line or a logarithmic curve of varying steepness.

Choosing whether to use branching or crossfading depends on the genre of music and intended aesthetic. Tracks with a regular beat like dance music within an energetic portion of the game may prefer to use branching because the entry of the next track needs to be impactful and
be introduced with a punch. Other tracks like ambient music or situations which are less impactful might want to explore crossfading as a transition option.

Another transition is layering, here the composer exports stems (layers of instruments) instead of a full track, allowing the audio implementer more flexibility for transitions. This allows for all the stems to play at the same time, but the system decides which stems are audible. An example

FIGURE 15.3  Music transitions. (© Falk Au Yeong, soundtrec. With permission.)

FIGURE 15.4  Final Fantasy XV. (© Square Enix.)
in Square Enix’s *Final Fantasy XV* is when the player is in a pit stop like ‘Hammerhead’, the same song may be playing when you’re outdoors and inside a shop, yet when player enters the shop, the instruments of the track change while maintaining the tempo, beat and melody (Figure 15.4).

The term ‘audio-driven gameplay’ is not limited to rhythm music games. In Playdead’s *INSIDE*, the animations are synced to the sound instead of the usual opposite way. This can be best illustrated in the game’s marching scene, where mindless human bodies march in unison and the player has to follow when they move forward or jump. As players die in the scene, the song doesn’t transition or cut – it plays throughout. As a player revives, both the animation and sounds of the march continue seamlessly (Figure 15.5).

In Ninja Theory’s *Hellblade: Senua’s Sacrifice*, there is hardly any visual 2D HUD to indicate the player’s health or how they’re feeling, therefore the game relies largely on sound to convey these gameplay aspects (Figure 15.6). Voice is a huge aspect of the character, Senua, where her thoughts do not simply play a single narrative sentence – short voice snippets often play back in her mind, very similar to how we think and be conscious of the ‘voices in our head’. To achieve the latter effect, the voices were recorded using a binaural microphone and implemented to be sparse or crowded with thoughts depending on her state of mind.

In conclusion, the basic understanding of non-linear audio in the context of games will hopefully encourage sound designers and game
developers to think about elegant, clever and creative audio solutions for their game projects regardless of platform and scope.

**LUDOGRAPHY**