Audio design and functions

STAY SILENT

FEATURES OVERVIEW

Gaspard Morel

NETEASE GAMES

网易游戏 | 艺术设计中心 ART DESIGN CENTER

THE GAME

VR First Person Shooter

Online multiplayer

Quick matchs in arena-like maps

Players are invisible to the others

Use items and sounds to pinpoint your enemy

Capture flags on the map to win

Cowboy vs Alien aesthetic

Items have various effects like shields, detectors, interfering with your enemy’s equipment, etc

DESIGN PRINCIPLES

Style

Focus

Mixing

STYLE

- Hyper realistic type of sound
- A mixture of influences, with old guns and wood/metal mechanisms, but also alien technology that needed to be more slick and colourful
- Details on Guns, UI and ambience (these are the main gameplay interactions)
- Interaction sounds like UI and movement must incorporate meaning and variations being related to the specific action a player can do. In most cases, generic sounds could either confuse the user experience or break the immersion
- Reactive audio, separated into parts and layers. Outsourcing the production and at the same time retaining control over the possibilities of easy iteration over the layers would have been impossible. Everything about SFX was produced by myself

AUDIO FOCUS

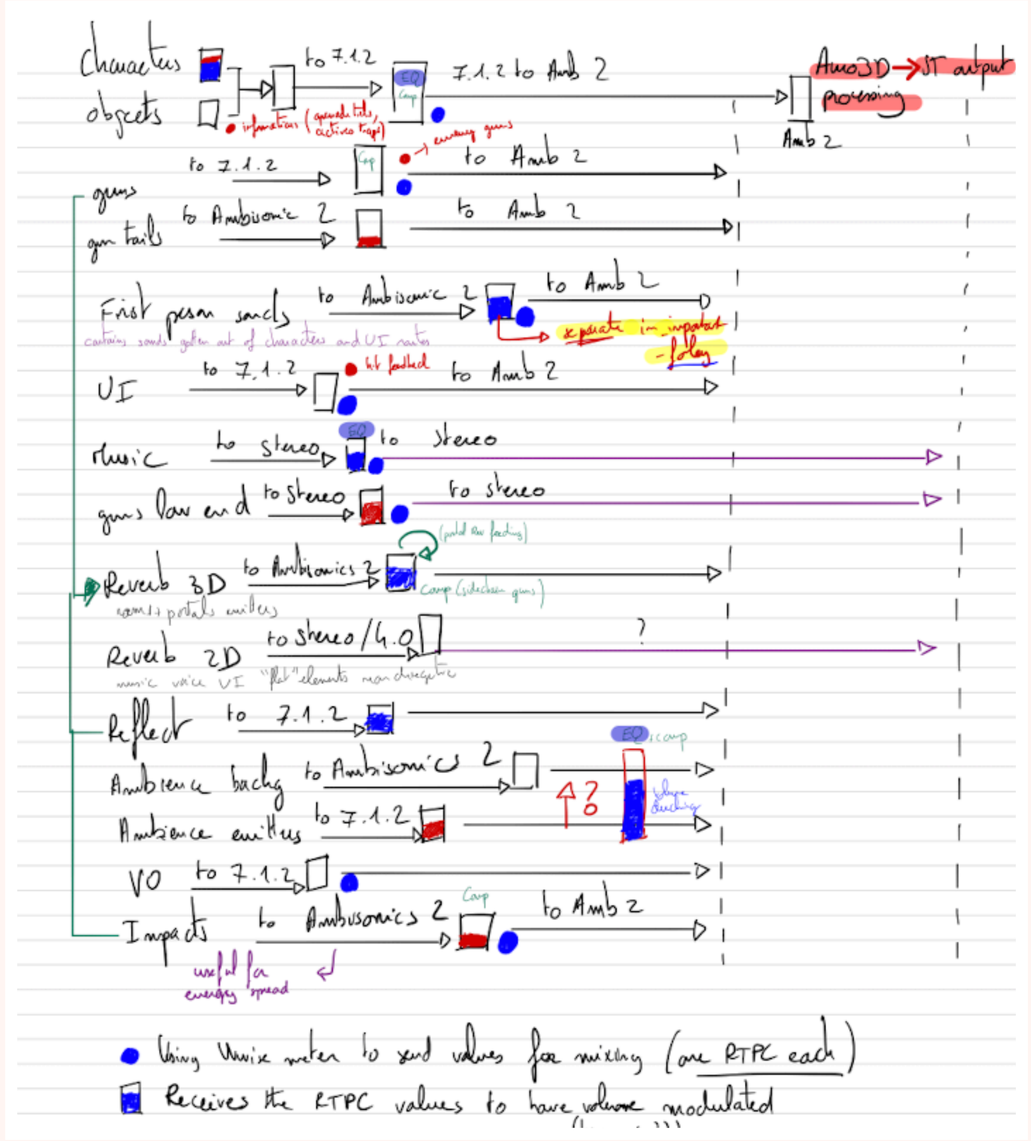
- The game features two mains phases, that is a map observation and enemy research using gadgets or sound, then gunfights
- I opted for a moody ambience and mixing for the first phase, and something between modern FPS games and an arcade feel for the second phase
- Environment details are placed through the map, subtleties in the movement, use of weapons and items can be heard when using it in the first phase
- During combat, these details are mixed down favouriting impact and UI feedbacks

MIXING

- The goal was to determine early which sounds were going to be important, which channel configurations were needed and were to place level side-chaining and Compression/EQ
- Taking into account the gameplay, the Audio focus, and the VR medium 9DOF, a mix of channels configuration were used for the buses (Stereo, Ambisonics, 7.1.2) leading up to a binaural rendering for a majority of the buses
- Once the structure is made, adding sounds sources and adjusting the balance is pretty straightforward

MIXING

- Guns, UI and VO have priority over the rest
- Interactions and movement comes second
- Space feedback (reflections and reverberation) are next, with Ambience emitters and Music
- Ambience background then glue everything together when the soundscape is calmer



Mixing buses notes for direction

DESIGN PRINCIPLES

Integration

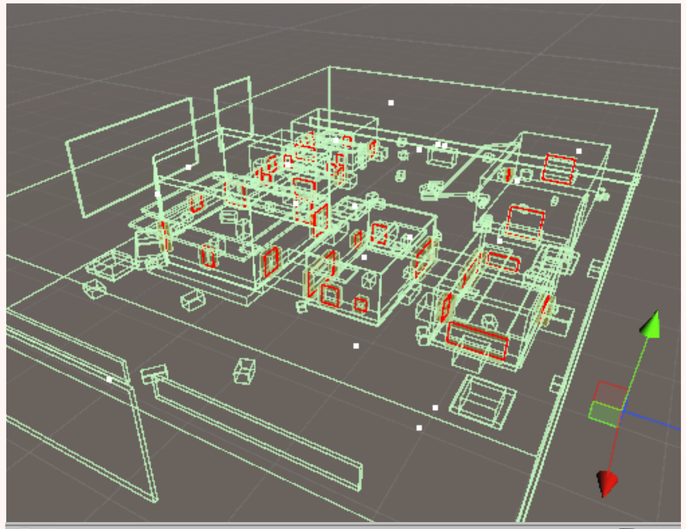
Spatial Audio

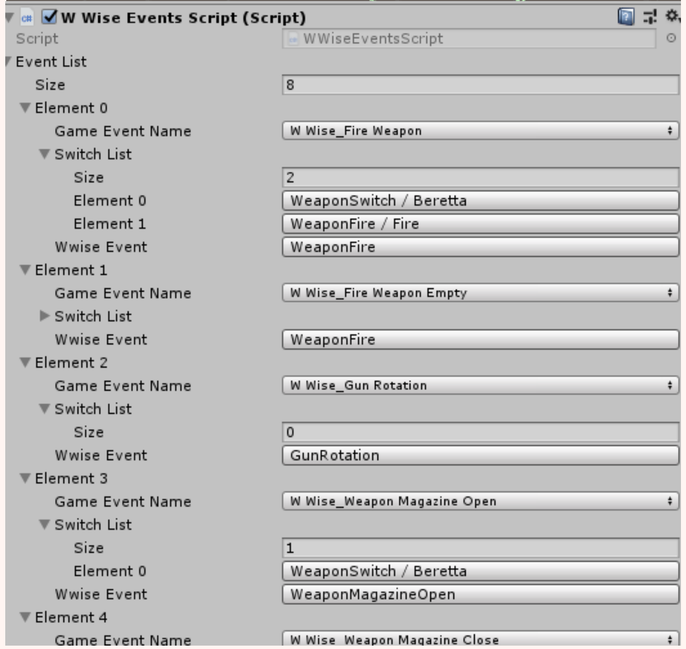
Wwise Reflect

Performances

INTEGRATION

- Communication, efficiency and independence were the main goals
- Separate Audio objects from the Unity scene files by using specific prefabs (enables versioning protection and integration responsibility, and enables adding audio content freely to maps or props)
- Using shared Wwise Events calling Switch structures for simpler management for Programmers when using with their Game Events - Wwise Events associations
- Game events and Wwise tool script: I designed a simple script to be used when dealing with an audio object that needed several events. Each Game event was exposed, allowing me to tie Wwise events and Switch values for this specific object





INTEGRATION

- Even with a relatively limited amount of content (5-6 maps, 8 weapons, 15-20 items, UI and additional content), the audio integration was depending on all teams schedules and wishes
- Organising and managing in-game objects (names, instantiations, prefabs) was important
- Keeping track of configurations for more complex emitters, ie Spatial Audio, obstruction, attenuation distances, listener and emitter cone filtering, management rules was critical
- Among these documents, keeping a tidy integration list was also key. It was also useful to take the time to give two levels of details in it, as in the basic events and parameters information for programmers that don’t want to be involved, and another level of details about parameters effects, game objects details, and events video recording for those who wanted to understand everything

INTEGRATION

- I also took on the responsibility to integrate audio components in the game items and content, using nested prefabs for versioning and instances:
  - In the items objects, managing position and audio colliders as well
  - On the player components, getting position around the head, feet and hands for interaction and colliders for bullet pass-by events
  - On UI and map components, again for position and specific variations in game events usage

SPATIAL AUDIO

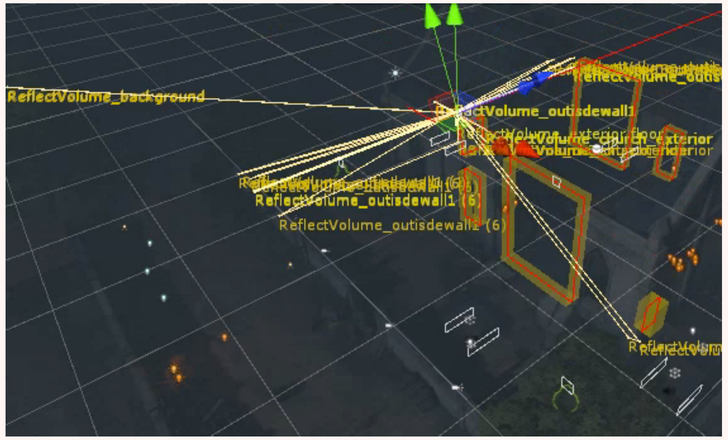
- The goal was to obtain precise position details and sound propagation through the environment for better immersion
- The static maps meant that I could control rooms and portals placement and size fairly easily
- Opening and closing some portals was required with breakable elements in walls late in the project
- Performances needed some management, Spatial Audio taking up a majority of the audio CPU time

WWISE REFLECT

- Dynamic gun tails generation and localised dynamic tap delay emitters was a goal for diversity, immersion and a dynamic soundscape (more than for a strategic gameplay advantage, since the maps are fairly simple and movement is not fast)

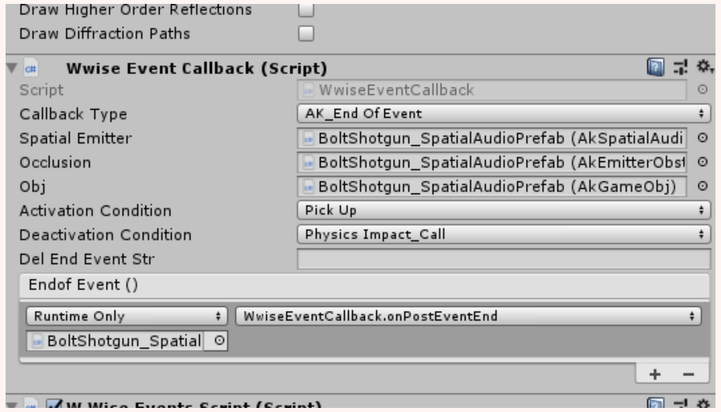
- Only for high intensity emitters like guns and grenades. Following Wwise's logic, dry sounds are implemented, sent to a Reflect plugin and reverberations, and the reflections are also sent to the reverberation buses

- To get aesthetically pleasing results, the reflections were heavily iterated with distance (from the source and from the reflection) using the available setup tools and reverb aux sends. With constraints of time for optimisation, only algorithmic reverbs are used right now (Convolution might become an option, especially for open spaces)



PERFORMANCES

- Spatial Audio in Wwise can use a high level of CPU time over one or two frames when having to deal with a lot of portal paths between rooms
- The goal was to find the bottlenecks and the conditional logic to control it a bit more (number of portals, distances, global capping, etc)
- When the project was using many items at once in the game maps, another tool script was made using custom conditions and callbacks to deactivate Spatial Audio components when the emitter was not used



PARAMETERS USAGE

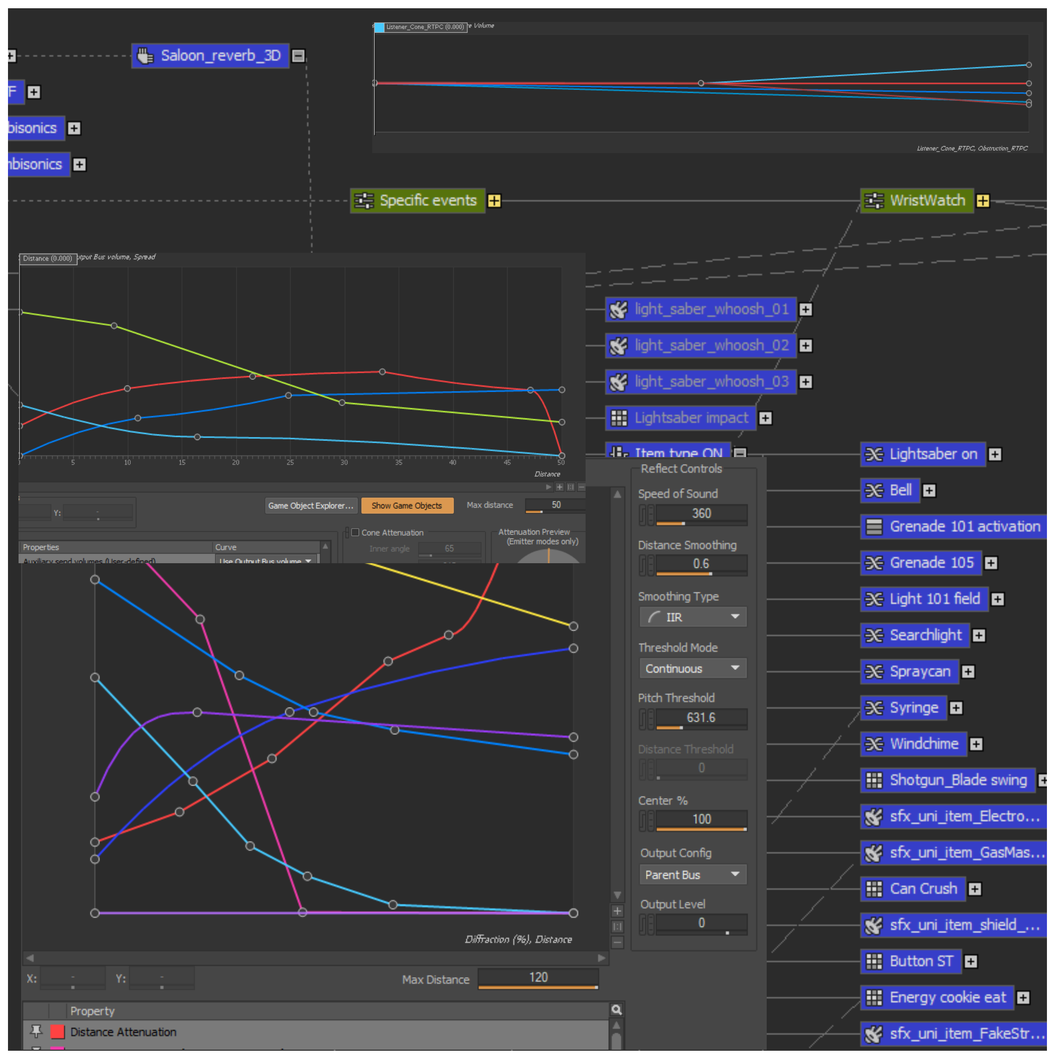
Vision focus

Distance

Ambience

Movement

Music structure



VISION FOCUS

- Since the player has heavy agency over where to look, this interaction can help provide a clearer mix
- Player focus is also increased when choosing the emitters that will be filtered/pitched/ mixed down when the player don’t look at them or they are not relevant to the action (gun fire orientation or scene details for example)
- Using obstruction values, listener cones and emitter cones values in Wwise to manage all of this
- Use Mixing categories to choose what to filter using these parameters

DISTANCE

- Use of blending between different audio files over distance on top of attenuation curves
- Especially for sounds consisting in several layers, that are attenuated differently over distance: the emitter blends into a single asset over distance for easier management and control
- Specific emitters, items or UI feedbacks were also attenuated specifically depending on their needs to get unique frequencies underlined
- Again, good naming and tracking was key to keep track of all the variations for attenuation values

AMBIENCE

- Foundation layer of ambisonic loops around the player and emitters around the map
- Three RTPC drive these sounds: one for the player being inside or outside a closed space, a time of day, and a weather intensity
- Inside/outside: ambience sounds are filtered depending on the situation, and the curves can be offset by using the Map IDs parameters
- Time of day: the background and emitters both have two version for the day and night possibilities of each map. The game design was including a way to switch during gameplay to different times of day even if it is used in a more binary way in the final game
- Weather: rain, wind and thunder sounds can be blended in following these parameters. The rain also reacts to the inside/outside values to be changed and positioned above the player in the ambisonic bus

MOVEMENT

- Adding details in the interactions that are possible in the gameplay was important
- Depending on states and switch values (game stage, invisibility), additional layers can be heard on different actions (like footsteps, grabbing, etc)
- Parameters measuring the angle and velocity of the hands in space are also used to add foleys details to the player actions, like leather creaking or clothes rustling around the shoulders. Delta time comparison on these values trigger events
- Objects like a laser sword, physics impact or thrown objects are also using events for collision or switching on and off, but velocity, angle and elevation were used to drive sound blend and playback to follow more closely the actual physics
- Players are categorised by ID, to be able to mix different layers for the same actions. Using a gun in first person will play a low frequency layer for a kick, some actions will not be heard from other players when they are invisible, attenuation curves can be offset for foleys, etc

MUSIC STRUCTURE

- The initial request was having close to none
- For building ambience and motivation, we advised it was better to use a structure used by many FPS competitive games, like RB6, Apex Legends or Overwatch
- Music plays according to this structure, following the map the player is in, building up from the Lobby to the start of a game during loading, and playing pumping up music up until an introduction before players start the match
- Some additional music is playing during the capture of a flag, elevating tension with strings and rhythmic patterns. Tracks can add to that depending on parameters like the closeness of bullets impacts or pass-by, or the level of items and gun sound buses in Wwise to bring even more tension to this specific action

ADDITIONAL FEATURES

- Before the final release of the game, I want to take time to implement additional small things if possible
  - Adding the usage of Wwise motion to the final mix, to be able to drive controller's vibration depending on the audio level of some of the audio buses
  - Doppler effect on grenades: these items have a ticking sound before the explosion is triggered, and travel through air after being thrown. I want to implement good parameter values when tracking their 3D coordinates to add a proper doppler effect on this sound
  - Adding experimental binaural oscillations in drone sounds that play when a match starts or the game time is coming to an end. I want to test if these kind of oscillations in headphones can influence the mood of the players at the right frequencies