Global Audio Manager Manual

Baranger - Holsnyder

October 2, 2015

Contents

Introduction

GAM is a set of tools designed to optimize and speed up audio integration.

1 Arborescence

A file hierarchy has been set up to allow for an optimized, faster audio integration in your Unity project. This hierarchy represents a conform audio structure, split into three domains : Voice, Sound Effects (SFX) and Music (see Generalities).

2 Audio Clip Manager

Access : Menu Windows/Global Audio Manager/GAMAudioClipManager

The Audio Clip Manager is a file browser designed specifically to allow a global view of all audio assets and optimize their management. All audio import settings associated with an AudioClip, along with platform-specific options, can be managed from here. This becomes indispensable when working on projects ranging from medium-size to maddening large.

AudioClips are split into 3 audio types (Music, SFX and Voices. See Generalities section for more info). This separation is extremely useful for giving type-specific values to audio import settings of whole batches of clips. All import settings should be type-specific to allow maximum quality and maximum performance at runtime. Clips corresponding to each type can be viewed separately, or all can be show together ; a fifth option allows to show all typeless clips. This can be managed with the 5 buttons on the left-hand side of the toolbar. This separation can be defined manually, by assigning singular or batches of AudioClips to a type (to select an array of clips, use the shift key or control(PC)/command(Mac) key), or automatically, provided the name of the type appears somewhere along the project-relative file path.

The browser view itself imitates the regular Unity browser view, each line representing an AudioClip, and columns its settings for various runtime and import parameters :

- To Mono : force a clip into a mono configuration. This transforms a multichannel audio clip into a mono clip at runtime. Such a reduction can bypass issues with left/right panning inside an audioclip, like a wrong panning.
- Background Load
- Preload :
- Load Type :
- Format : the audio codec used for the audioclip. Allows changes on the quality/file size scale. Recommended : Vorbis, with type-specific quality settings (see below).
- Quality : compression rate of a compressed audioclip. The higher the quality, the lower the compression. Recommended optimal compression with minimal quality loss : ˜80% for music, ˜40% for SFX and ˜70% for voices. These numbers can greatly vary depending on the considered sounds, their quality and the desired quality.
- Sample Rate
- Label : the type (Music/SFX/Voice) of the clip.

3 UI

The UI Utility automatically references all the UI objects which are liable to play a sound on user interaction, and allows to define and control these sounds and their play settings.

4 Music

Easy integration of a mono or stereo dynamic music. The example shows the structural slicing of an 8-bit version of the Tetris music. More advanced options can be used to create a multitrack randomized music with the same slicing method. The example shows the process of creating such a music from a gypsy-jazz version of the same Tetris music.

5 Switch

Fundamental tool of the sound designer. The Switch allows for variation on sound samples, and is frequently used on sounds which tend to be repetitive, like footsteps, or the sound of a water drop. Without this, the game can only sound artificial and poor.

6 Animation

Utility allowing for in-editor animation preview with sound and sound event addition and synchronization on the animation timeline. This utility allows for a classic *play switch* event (see section 5), or for a *play switch on surface* event, which automatically detects the surface under the animated object and plays a corresponding switch. The latter can typically be used for automatic surface-dependent footstep sounds on animated characters.

7 Generalities

Audio design in a video game is made up of several layers :

- Music.
- Environment Sound Effects :
  - Background : non-spatialized, 2D sounds.
  - Objects : spatialized, 3D sounds.
- Action SFX.
- UI SFX.
- Voices.

7.1 Non-spatialized sounds

Unity calls these 2D sounds. DO NOT mistake that for a 2D-game-exclusive instruction. Both 2D and 3D sounds should be used for any Unity project. A non-spatialized sound is played back directly by the sound engine, without any processing from Unity, exactly like if it were played by a regular media player. This setting is often (but not systematically) used for music, voices and background SFX.

Example : Jardin, arbres, oiseaux, fontaine. Background : Birds in the trees. More or less present depending on the time of day.

7.2 Fixed spatialized sounds

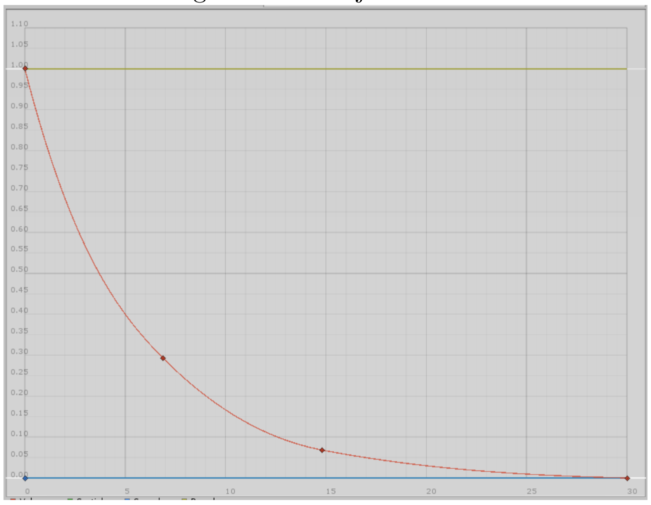
Fixed spatialized sounds are used for objects which don't move. Nevertheless, if the listener moves, their volume will change accordingly.

Example : Jardin, arbres, oiseaux, fontaine. Fixed spatialized sound : the fountain. It emits sound, and the player can get closer or farther. So the fountain will have an audio source, with a volume rolloff curve. For a classic, realistic rolloff, the curve should be an invert-type function :

7.3 Mobile spatialized sounds

A spatialized sound may be attached to a moving object.

Figure 1: 2D object rolloff.



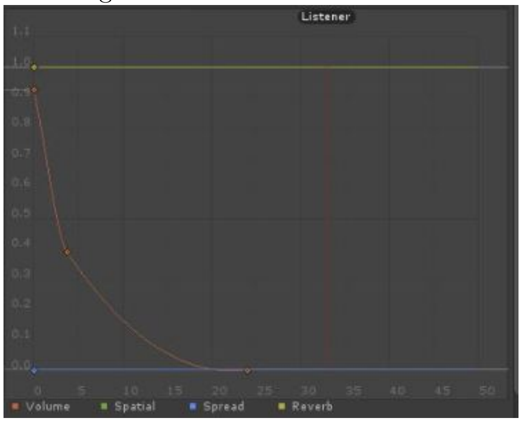
Example: Jardin, arbres, oiseaux, fontaine. A crow passes overhead and sticks around. To achieve realism, 3 elements are necessary :

A switch with 2 cawing sounds at the least.

A Doppler effect to add to the feeling of changing distance.

A volume rolloff of the type :

Figure 2: Invert function rolloff.


Listener
Volume
Spatial
Spread
Reverb