Global Audio Manager Manual

Baranger - Holsnyder

October 6, 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 within a *Resources* subfolder 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 section 7).

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 (actually the same ones as in the regular AudioClip inspector view), 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 to performance proportion 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.

On the right side of the toolbar, an array (or drop-down menu) of gaming platforms is represented. By selecting a platform, you can edit the import settings for it, independently from the others. Or make it follow the default pattern.

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. Modifying a setting for an AudioClip within a selection will modify the setting in the same manner for all selected clips.

3 UI

The UI Utility automatically references all the UI objects which are liable to play a sound on user interaction[1], and allows to define and control these sounds and their playback settings.

3.1 GAM UI Editor

Access : Menu Windows/Global Audio Manager/GAMUI

This window allows a global overview of all selectable objects within the project, or under a specific canvas. It is split horizontally into three sections.

3.1.1 Events-sounds assignments

The top section allows building a *setting*, which is basically a list of UI Events paired to Switches (see Switch section). All possible UI events are available.

This setting can in turn be applied (Apply setting button) to all UI objects referenced in the Objects Overview. On the contrary, the Clear objects of events button will delete all relevant[2] events on these objects.

[1] Components derived from the Selectable class.
[2] Meaning events which trigger a switch

3.1.2 Parent objects

The *parent objects* determine where to look for UI game objects. Canvases containing UI objects can be referenced using the + button, or all can be automatically added using the Add canvases in scene button.

3.1.3 Objects Overview

All referenced UI objects and all their relevant[3] UI events are shown here.

4 Music

4.1 GAMMusic window

Access : Menu Windows/Global Audio Manager/GAMMusic

Allows integration of dynamic music, from simple setups to complex interactive systems. Small musical cells are fed into GAMMusic (possibly with separate instruments), and a complete dynamic music can be built.

4.1.1 Parts

These elements are the reservoir side of music building, or the vertical assembling. They can be managed from the lower part of the window. All elements within this section can be activated and deactivated (and will respectively be taken in consideration during playback or ignored).

Cells: Music *cells* are the smallest building block of a music. They represent a short length of music, and either a slice of the whole music, or a slice of an instrumental track. Cells should be made in different versions[4], and during playback one of these will be picked randomly (in the manner of a simplified Switch) by the *track* container.

Track: these contain similar cells (the several versions mentioned above). They are in turn summed by the *part*. They can be faded in and out.

Part: A part contains several instruments, and will play them together (and each one will pick a cell version, if any). It represents the total vertical slice of music. These are the basic building blocks for *structuring* music.

The menu in the purple toolbar allows loading AudioClips with respect to folder structure : Selecting a folder containing directories and subdirectories following the above hierarchy will load them into the music. The folder structure should be : Music folder/Part/Track/Cell.wav.

4.1.2 Structures

Structures are the instantiation of elements of the reservoir side, also named horizontal assembling. They are managed from the top part of the window.
The menu in the purple toolbar allows switching structures or adding new ones.

Structure: a chain of parts. Each part in a chain can be looped (special instructions must then be sent to beak out of the loop). Each structure will be looped unless a follow-up is triggered.

4.1.3 Mixes

Mixes interact with both *parts* and *structures* and are the actual way of playing a music : a mix references a structure, and a setup of parts elements activations and fades. They can be managed from the left-hand side panel in the window.

The menu in the purple toolbar allows adding mixes.

[3] Meaning events which trigger a switch

[4] One can use humanized quantization options on MIDI files to achieve realism. An audio recording should be made several times to produce different versions.

4.1.4 GAMMusic file

All of the above should be **saved** into a special asset - a music file. The top-most purple toolbar allows all usual interactions on a file : opening, saving, saving as, creating a new one...

4.2 GAMMusic Component

5 Switch

5.1 Quick theory

The fundamental tool of the sound designer. The switch allows for variation on sound samples, and is frequently used on any sound that occurs more than once. Without this, the game inevitably sounds artificial and poor.

The switch is at its core an array of similar AudioClips (for example, several footstep sounds for the same character walking on a given material) ; when playing a sound, instead of directly calling an AudioClip, a switch containing the AudioClip and its variations should be called. The switch will then randomly choose one of its AudioClips and play it.

5.2 Switch creation and setup

In GAM, the switch is an asset, created in a similar manner to any other asset : right-click in the browser view. Doing this with folders or AudioClips selected will create a switch referencing the selected AudioClips and the ones under the selected directories.

The user has a great amount of control over the playback of a switch, which allows for a limitless sound variation potential.

- **Presets** include Music, Sound Effects and UI and setup the parameters of your switch to adequate values. This should be used carefully though, and requires checking, as all sounds won't react in the same manner to the possible changes.
- **Play On Start** matches the option on the AudioSource.
- **2D/3D** actually matches the spatial blend in an AudioSource, and will override it with the boundary values on the scale.
- **2D/3D**
- **Volume** is self-explanatory. It will scale the volume for all the contained AudioClips.
- **Loop Policy** handles the way the AudioClips will be played when the switch is triggered.
  - *One shot* Picks a clip and plays it.
  - *Repeat* will continuously play the contained AudioClips, picking one when one finished. This setting and the following one are useful for audio backgrounds.
  - *Repeat and exclude last element* will also continuously play the contained AudioClips, but will not play the same one twice in a row.
- **Fade** handles fade-ins and fade-outs. If ticked, fade-in and fade-out durations below will be applied to switch playback (at the beginning and at the end of the switch, respectively, whether playback is looped or one-shot).
- **Random pitch** randomizes the pitch of an AudioClip when it is played. The values displayed are in semitones. The randomization range is represented by the min-max slider. This can be especially useful for footsteps and other repetitive small impacts.
- **Element** is the name given to the referenced AudioClips, as they have slightly increased functionality : they are weighted. That's the slider on the left of each AudioClip line. Weight works like a normal, statistical/randomization weight : the higher it is, the more chances the AudioClip has of being picked at playback. A weight of 0 will prevent the AudioClip from being picked.
- **Add Audio from directory** allows to add all AudioClips in a directory.

5.3 Integration: Switch component

Since the switch extends the regular AudioClip/AudioSource functionality, it has its own component for integration : the **GAMSwitchComponent**, under Scripts. It allows referencing an existing switch and edit its settings (with the same interface as the own for the asset inspector). A Switch Component should have its own GameObject to avoid any AudioSource conflict with another Audio component.
The switch can handle AudioSource creation, but creating one for each switch component is recommended for more flexibility (the AudioSource still has some options the switch is missing, especially the tracks routing).

6 Animation

Access : Menu Windows/Global Audio Manager/GAMAnimation Or via GAMAnimation component.

Utilities allowing 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.
Two buttons on the left-hand side of the toolbar allow to switch views between Animation and Footsteps.

6.1 GAM Animation Component

6.2 Animation view

A global view of all Animation Clips present in the project. Mainly useful for 3D animations : Unity's 2D animation preview isn't handled well and can be completed with any sprite animation preview package on the asset store (beware though, some of these plugins may break the 3D animations preview). 3D animations require an avatar to generate a preview.

6.3 Footsteps view

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 : A garden with *birds* in the *trees* and a *fountain* in the center. Background audio : 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 : A garden with *birds* in the *trees* and a *fountain* in the center. 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.

Example: A garden with *birds* in the *trees* and a *fountain* in the center. 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 :