Eqo Ambience Creator Documentation

For support, questions or bug reports

Use Discord

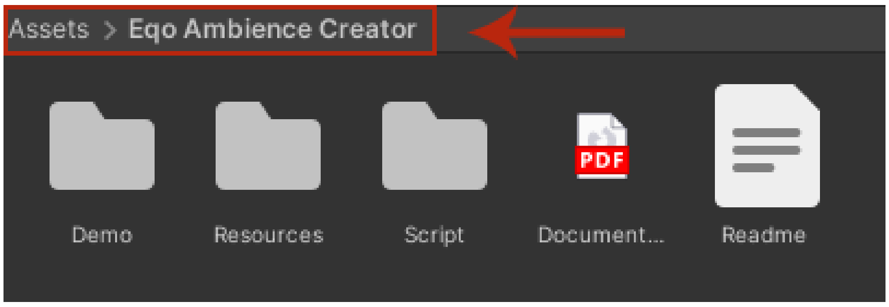
Email me: pathiralgames@gmail.com

INDEX

TAKE NOTE:

The Eqo asset folder must be placed in the main *Assets/* root and any movement to any other location will cause the system to not be able to read the audio files.

The system will automatically place itself in the correct location on import. So just don’t move it from there.



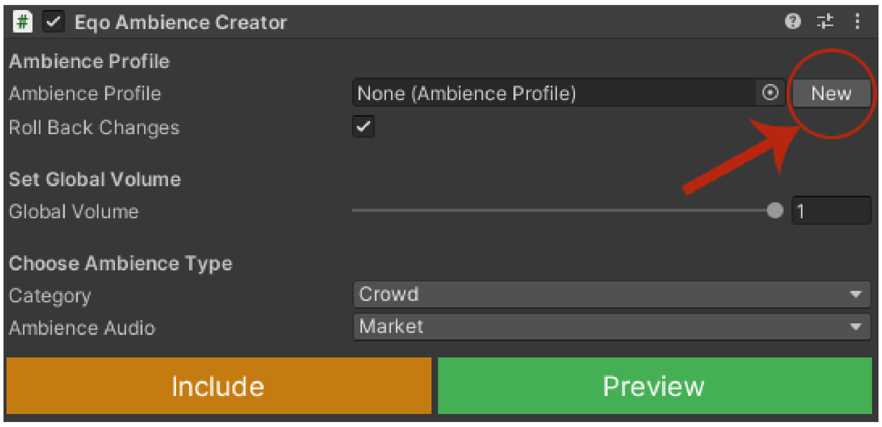
GETTING STARTED: (Video tutorial)

- Create an empty game object in your scene.

- Add component: **Eqo Ambience Creator**. To this empty game object.

- Now in order to start using the system, you need to create an ambience profile (or add an existing one).

- To create a profile simply click on the **New** button.



Eqo Ambience Creator

Ambience Profile

Ambience Profile: None (Ambience Profile)

Roll Back Changes: ✓

Set Global Volume

Global Volume: 1

Choose Ambience Type

Category: Crowd

Ambience Audio: Market

- It'll automatically create a new profile and set it in the **Ambience Profile** property.

- Now you can start adding audios to create your ambience.

- Choose the *category* you like from the **Category** dropdown.

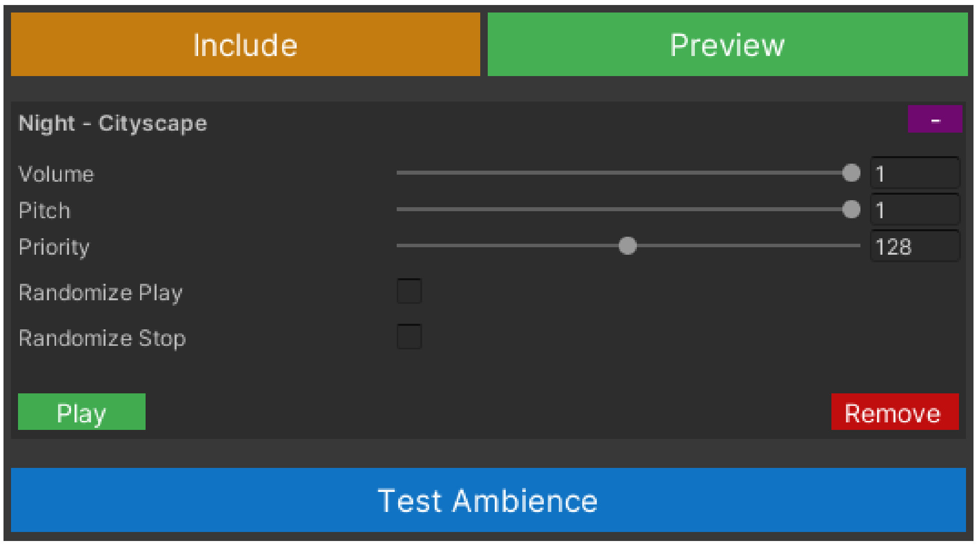
- Then select the *audio type* you like from the **Ambience Audio** dropdown. *The ambience audio dropdown is based on the category selected.*

- As seen in the picture above I have selected the category **Night** and the ambience audio **Cityscape**. Choose whatever you like.

- Now you can preview the audio selection before actually including it. If you click on the big green button **Preview**, the audio will play so you can make sure it's the type of audio you need.

- Now to add the selected audio. Simply press on the big orange button **Include**.

- You will find that the new audio has been added with a new audio block dedicated to that audio.



Night - Cityscape

Volume	1
Pitch	1
Priority	128
Randomize Play	☐
Randomize Stop	☐

- You will also find that a new blue button has appeared called **Test Ambience**. This will play all of the included audios together so you can set their volumes and properties in order to find the perfect mix.

- If you want to play an included audio individually simply click on it's green **Play** button.

- Clicking on the purple button in the top right corner will minimize the block. To save inspector space.

- After adding the second audio let's test the ambience in order to find the perfect mix.

- Click on **Test Ambience** button.

- All included audios will play now and you can set their properties as you wish until you reach the sweet spot of your desired ambience.

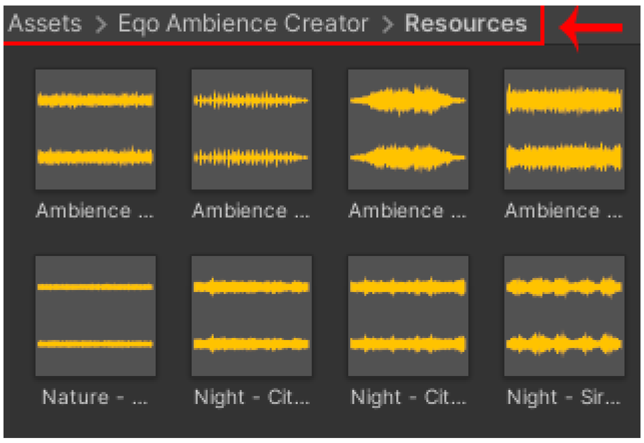
- On game start, all audios will play and you have created your ambience.

ADDING YOUR OWN AUDIOS:

Eqo gives you the freedom to use any audio of yours with the system. There are only two requirements. It's **location** and **naming**.

Location

In order for Eqo to read your audio you need to place it in the **Resources** folder inside the *Eqo Ambience Creator* folder. You will find the rest of the audios are there too.



Naming

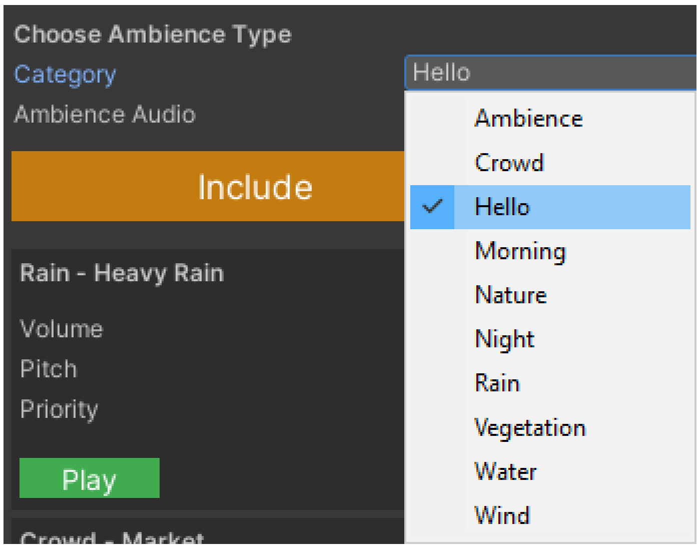
The second requirement to have eqo read your audio is the naming. You simply need to have a dash (–) between two words. The word(s) before the dash depicts the category name and the word(s) after the dash depicts the audio name. Here is an example:

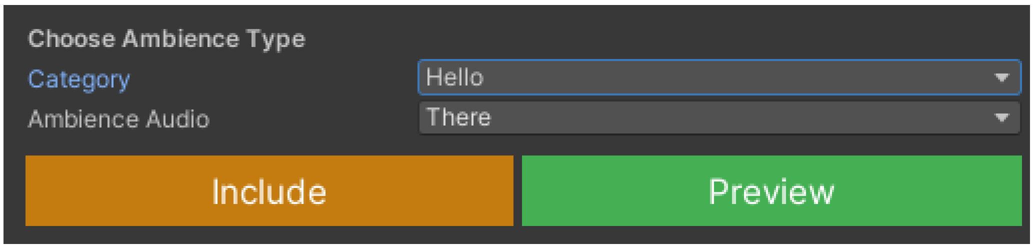
I added a new audio in the Resources folder and called it "Hello - There".

Hello - will be the category name the audio is to be found in.

There - is the actual name of the audio in the Ambience Audio dropdown.

Now when I go back to eqo's inspector. A new Category name called **Hello** has been added and when chosen, the audio dropdown detects **There** as an audio part of that category.





Category: Hello
Ambience Audio: There

And now you have added an audio that can be read as part of the system by eqo and can be included to your profiles. You can also add your new audio to an existing category by simply naming it as such. **The category and audio names don't need to be one word. Name them as you like.**

One important thing to note that: there must be a dash between two words but you don't need to have spaces. Eqo reads them all the same so all these examples will be read the same no problem:

Hello - There

Hello-There

Hello- There

Hello -There

Write it however you prefer. Spacing is not an issue.

APIs & Properties:

You get to have strong control over *eqo* using the APIs. Change properties, profiles, add/remove audios and more. These are all public methods & properties that can be called from external scripts.

EqoAmbienceCreator.instance – Access the Eqo component in your scene from any external script using this line of code. Giving you direct access to all of it's properties and functions.

ambienceProfile – Change the profile using this property. It takes a scriptable object of type *Ambience Profile*.

rollBackChanges – Set to true or false. Whether you want roll back changes to be applied or not.

globalVolume – Set from 0 to 1. This property controls the global volume of the ambience.

useFading – Set to use use fading or not on profile change & randomize stop.

fadeSpeed - The fading speed for any fade in/fade out operation.

PlayAll() – Plays all the audios.

StopAll(bool fadeOut=true) – Stops all audios. Passing **true** will make the audios fade out and then stop. It's **defaulted to true**. For an instant stop, pass **false**.

AddAudio(string fileName) – Adds the passed audio name to ambience. The file must be in the Resources folder inside Eqo Ambience Creator folder. **You only need to pass the file name without it's format**. So if you have an audio file called: "Water – Calm Sea.wav" simply pass ***AddAudio("Water – Calm Sea");***

AddAudio(AudioClip clip) – Adds the passed audio clip to ambience. Same as the above but you can pass the audio clip directly.

GetAudioIndexFromName(string name) – Returns an int array representing the indexes of all the included audios with the passed name.

Example:

```
// returns [0, 2] because I have two of these audios added

int[] indexes = eqoScript.GetAudioIndexFromName("Rain – Light Rain");


// remove all the light rain audios

for (int i=0; i<indexes.Length; i++) {

   eqoScript.RemoveAudio(indexes[i]);

}
```

RemoveAudio(int index, bool fadeOut=true) – Remove audio by it's index from ambience. If you pass **true** in the second parameter the audio will fade out then get removed. It's **defaulted to true.**

PlayAudio(int index, bool fadeIn=true) – Plays audio of passed index. If you pass **true** in the second parameter the audio will fade in. It's **defaulted to true.** For an **instant play, pass false** in the second parameter.

PlayAudio(string name, bool playAll=false) – Plays audio of passed name. If you have several audios with the same name that you want all to play. You can pass ***true*** as the second argument. Otherwise by default it'll only play the first one with the same name. *The audio name is how it's written in it's audio block. Example: "Rain – Light Rain".*

StopAudio(int index, bool fadeOut=true) - Stops the audio with the passed index. Passing true in the second parameter will make the audio fade out first. It's **defaulted to true**. For an **instant stop, pass false** as the second parameter.

StopAudio(string name, bool stopAll, bool fadeout=true) - Stops audio of passed name. If you have several audios with the same name that you want all to stop. You can pass ***true*** as the second argument. Otherwise by default it'll only stop the first one with the same name. *The audio name is how it's written in it's audio block. Example: "Rain – Light Rain".*

GetAudioSource(int index) – Returns the audio source of the passed index.

GetAudioSources() – Returns an AudioSource array with all audio sources of the included audios.

GetAudiosLength() – Returns an int of how many audios are included.

ChangeAudioProperty(int index, string propertyType, float value) – Use this method to change either the **volume**, **pitch** or **priority** of the audios.

Example:

eqoScript.ChangeAudioProperty(0, “volume”, 0);

eqoScript.ChangeAudioProperty(1, “pitch”, 0.5f);

eqoScript.ChangeAudioProperty(2, “priority”, 256);

For changing anything else get the audio source itself using GetAudioSource(int index) and then change what you need.

GetProfileData() – Returns a list of the profile data struct.

Example:

List<AmbienceProfile.AudiosData> profileData = eqoScript.GetProfileData();

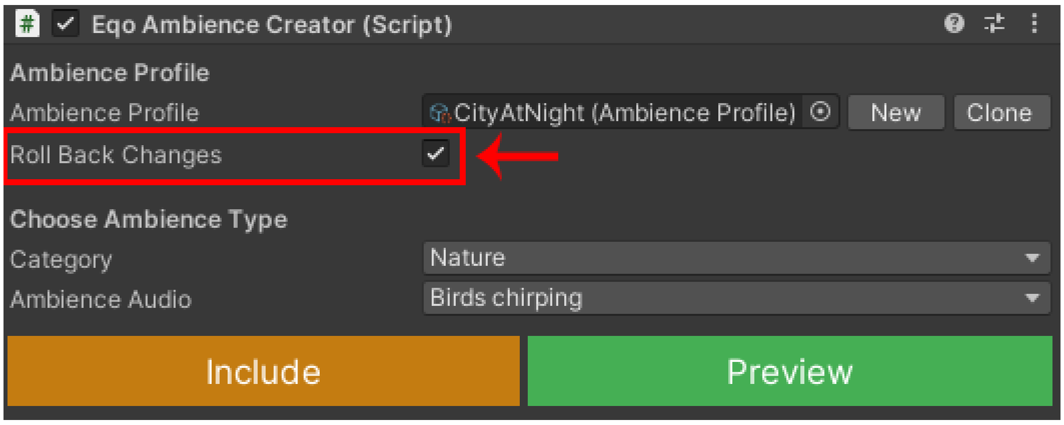
Each item in the return list is this struct:

```
public struct AudiosData {
        public string name;
        public AudioClip clip;
        public float volume;
        public float pitch;
        public int priority;
}
```

SaveProfileData() – Saves the current profile changes. So can rollback to these changes. If *Roll Back Changes* property is enabled then this function is run automatically on game start.

RollBackProfile() – Reset the profile data back to it’s original settings before the game started. In other words, rolls the profile data back to when *SaveProfileData()* has last been called.

Roll Back Changes: (video)


Eqo Ambience Creator (Script)

Ambience Profile

Ambience Profile: CityAtNight (Ambience Profile) New Clone

Roll Back Changes ✓

Choose Ambience Type

Category: Nature

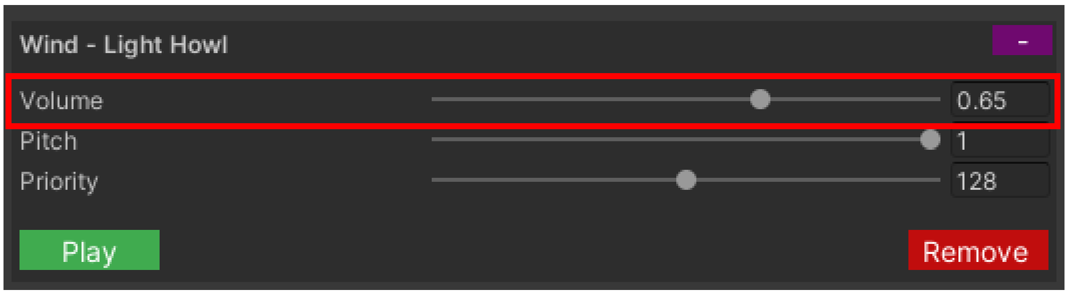
Ambience Audio: Birds chirping

This is a very important feature in Eqo where you can rollback any changes made to the profile and return it to it's original data it had before game start. Because the data is persistent, any change made to a profile even using the APIs within the game will change the profile permanently.

Example of Use

Have *Roll Back Changes* set to **true** and start the game. Then change any property of any audio block or even remove an entire audio block. On game end you will find the profile has been reset back to it's original data it had before game start and even each audio block has been reset back to it's original property values.

I set this audio block's *volume* to **0.65** before game start.

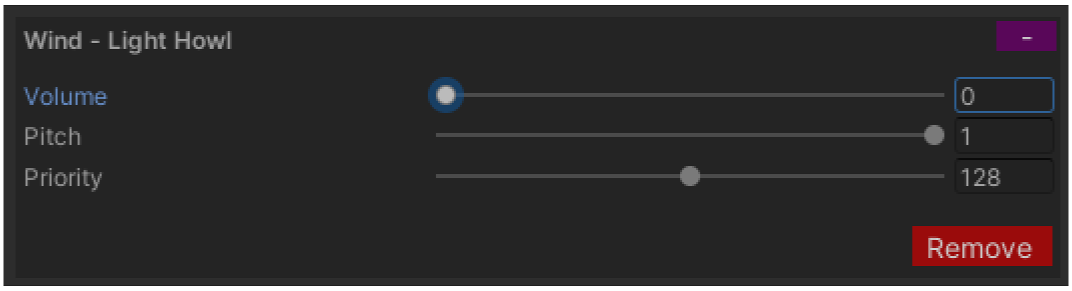

Wind - Light Howl

Volume 0.65

Pitch 1

Priority 128

Then I start the game and as the game is running, I turn the *volume* property down to **0**.

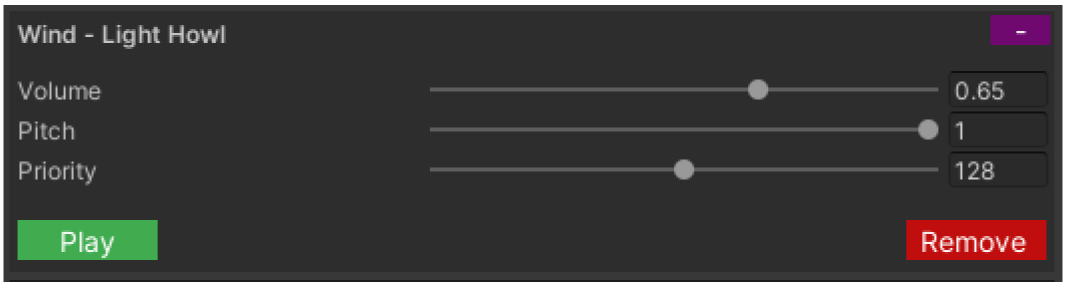

Wind - Light Howl

Volume 0

Pitch 1

Priority 128

Now I stop the game and the value is back to what it was before game start. This applies to all properties.


Wind - Light Howl

Volume 0.65

Pitch 1

Priority 128

The rollback of data isn't only on game/scene stop but also on profile swap. So if you swap the profile with another one programmatically, the rollback will be applied on the previous one which had changes. You can also manually trigger the Rollback using the ***RollBackProfile()*** API.

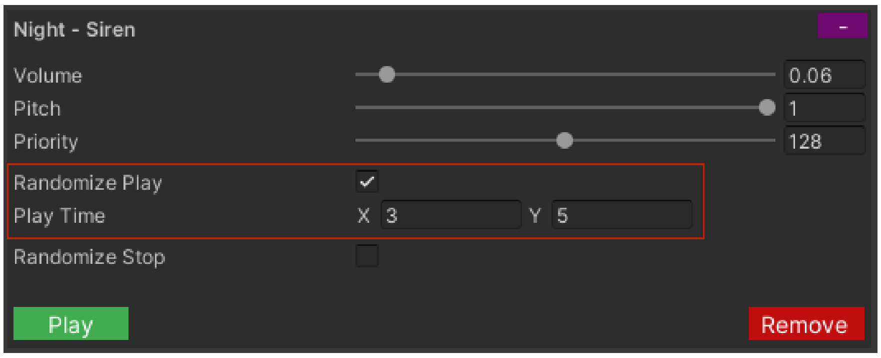
Important to know

The profile's data is persistent that means any change will cause it to have that change permanently. In Editor time, any change applied with rollback changes set to off will cause that data to stick. In game build even with rollback changes set to off the data will reset but only on scene unload or game exit.

Randomize Play:

This is a way to make each audio block play in random or certain durations. Let’s see how we can do this.

In each audio block there is a ***Randomize Play*** checkbox. Enable it.



Volume 0.06
Pitch 1
Priority 128
Randomize Play ✓
Play Time X 3 Y 5
Randomize Stop

When you enable it, another property will be drawn, called ***Play Time***. In this property set the range *(min & max)* in seconds to randomize a duration between these two values. So in other words, a random time will be generated between these two values. As you can see I set mine between **3** and **5** seconds.

For Fixed Time

If you want the time to be fixed then simply set the two inputs to the same value. So for example (2 & 2) this’ll make sure the audio plays every 2 seconds.

Back to Main Point

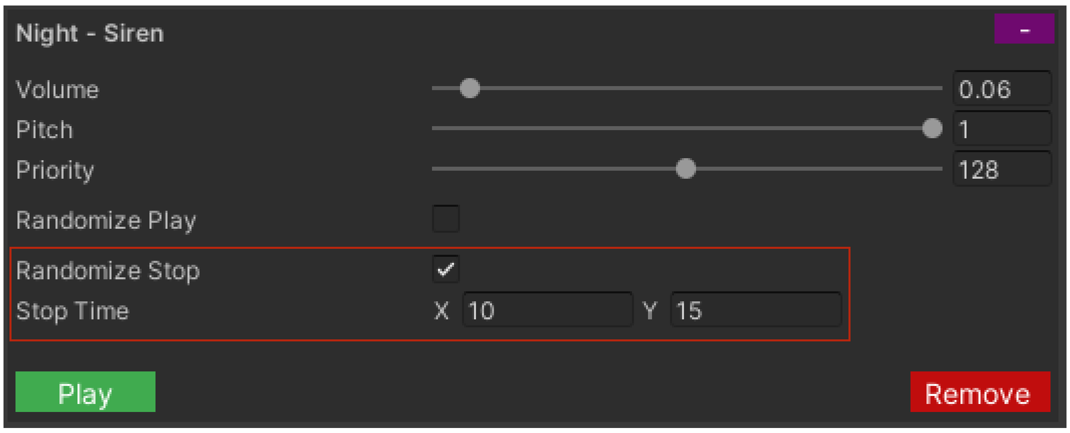
Now if you **Start** the game the audio block won’t play instantly until the back-end timer finishes, it’ll then play the audio and when it finishes, a new random time will be generated again and the timer will start again and so on.

One thing to note: the ***Test Ambience*** button does not consider the *Randomize Play* audios. Pressing the button will play all the audios no matter what. It doesn’t take into account any timers (timers only work on game run). This is obviously because Test Ambience is supposed to show you how all the audios together will sound at any point.

Randomize Stop:

This is a way to make the audios stop before actually finishing playing the entire clip.

In each audio block there is a ***Randomize Stop*** checkbox. Enable it.


Night - Siren

Volume 0.06
Pitch 1
Priority 128
Randomize Play
Randomize Stop ✓
Stop Time X 10 Y 15

When you enable it, another property will be drawn, called ***Stop Time***. In this property set the range *(min & max)* in seconds to randomize a stop duration between these two values. So in other words, a random time will be generated between these values. As you can see I set mine between **10** and **15** seconds.

For Fixed Time

If you want the value to be fixed then simply set the two inputs to the same value. So for example (2 & 2) this'll make sure the audio stops at the 2 second duration.

Back to Main Point

Now if you **Start** the game, you'll find that the audio stops when it's duration reaches that of the chosen randomized stop time. Not waiting for the entire clip to finish.

One thing to note: the ***Test Ambience*** button does not consider the *Randomize Stop* audios. Pressing the button will play all the audios till the end no matter what. It doesn't take into account any timers (timers only work on game run). This is obviously because Test Ambience is supposed to show you how all the audios together will sound at any point.