Sound System

This is a complete guide to using the **Sound System** tool in Unity. It is relevant for the version of the tool 0.2.1.0.

Link to the Sound System repository:

https://github.com/Linerichka/SoundSystem-With-Eazy-Sound-Manager

Link to the Eazy Sound Manager repository:

https://github.com/JackM36/Eazy-Sound-Manager

Link to the official Steam Audio website:

https://valvesoftware.github.io/steam-audio

Table of contents

Before you start

1. Download the latest version of the **Sound System** from the release section.

2. If you have not installed the plugin before, then follow step 3. If you already have the plugin installed and are using it in the project, skip step 3 and go to step 4.
3. Import the package into your project. In Unity, it looks like this: Assets – Import Package – Custom Package – choose the version you downloaded. Go to the next section of the manual.
4. If you already had the tool installed and it was used in the project, you need to make sure that the new version you want to install is compatible with your current version.

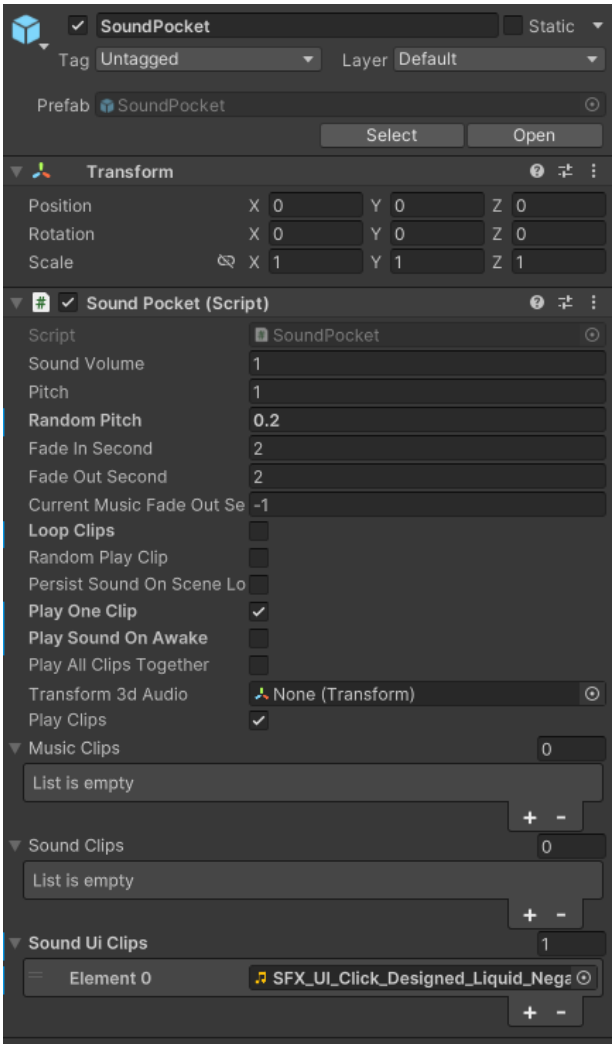
5. For older releases (up to and including version 0.0.1.0) this is indicated directly in the release itself in the section “Upgrade from the previous version:”. You can find out the version of the plugin that is used in the project using the file “Version.txt ”, which is on the way: “Assets\Lineri\SoundSystem\SoundSystem\Docs\Version\Version.txt ”.
6. For new releases, the augmented Semantic Versioning system is used, so when the release version is X.Y.Z.W, you can easily update if the version in your project is identical in X.Y. digits. That is, from version 1.0.0.0 you can upgrade to version 1.0.0.15 or version 1.0.2.0, but you cannot upgrade to version 1.1.0.0 or 2.0.0.0.

7. If after checking you are convinced that the versions are compatible, follow step **3**.

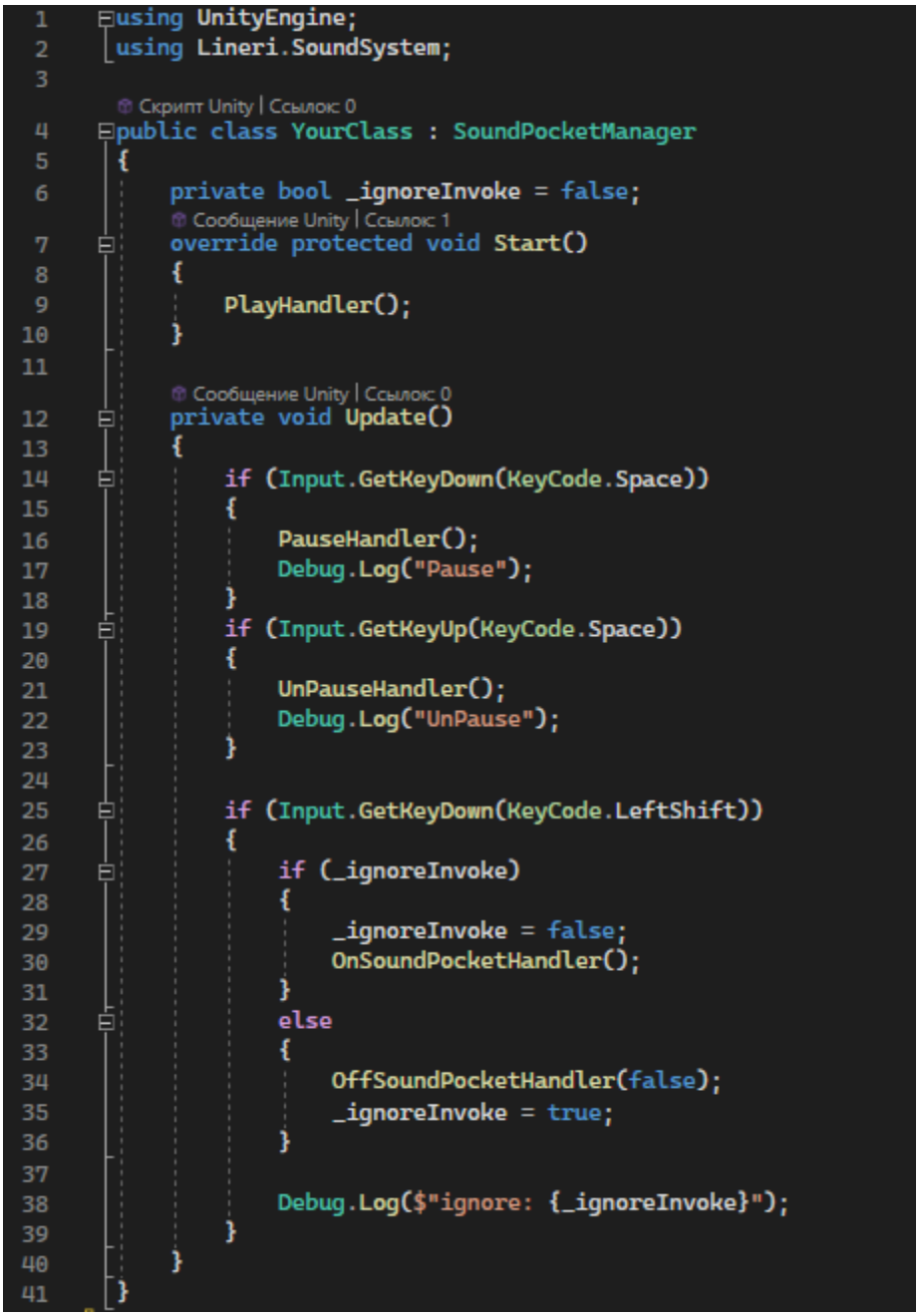
Working principle

Sound System – a simple and fast tool that is designed to share responsibilities when creating a sound scheme of the game, as well as significantly speed up this process. It is also worth noting that **Sound System** although it contains an **Eazy Sound Manager** plugin, it does not encourage working with it directly, because some elements of the **Eazy Sound Manager** can be changed. The separation of responsibilities is achieved by conveniently editing the settings of a group of clips directly from the inspector. Anyone without knowledge of the code can create a prefab with one or more "SoundPocket" classes, set the necessary settings and check their performance, or create simple sound schemes. If a more complex interaction is needed, this prefab can be transferred to a programmer who at this time will already create the necessary logic for your needs. That is, from the moment the task is set, the programmer and the designer can start working in parallel and not depend on each other's results.

As a result of the designer's work, a prefab with the necessary settings is obtained *SoundPocket*:



As a result of the programmer's work, we get a class that inherits from *SoundPocketManager* and implements the logic we need:



```
using UnityEngine;
using Lineri.SoundSystem;

Скрипт Unity | Ссылок: 0
public class YourClass : SoundPocketManager
{
    private bool _ignoreInvoke = false;
    Сообщение Unity | Ссылок: 1
    override protected void Start()
    {
        PlayHandler();
    }

    Сообщение Unity | Ссылок: 0
    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            PauseHandler();
            Debug.Log("Pause");
        }
        if (Input.GetKeyUp(KeyCode.Space))
        {
            UnPauseHandler();
            Debug.Log("UnPause");
        }

        if (Input.GetKeyDown(KeyCode.LeftShift))
        {
            if (_ignoreInvoke)
            {
                _ignoreInvoke = false;
                OnSoundPocketHandler();
            }
            else
            {
                OffSoundPocketHandler(false);
                _ignoreInvoke = true;
            }

            Debug.Log($"ignore: {_ignoreInvoke}");
        }
    }
}
```

And as a result, we get a customized prefab with sound, as well as a class that can fully control this sound the way we need it. Now it is enough to add this class to any object, in our case you will also need to add a prefab to the scene and specify a link to our object that we added from the prefab.

Getting started

1. To begin with, I'll add the “Sound” prefab to the stage, which is on the way: “Assets\Lineri\SoundSystem\SoundSystem\Prefabs\Sound.prefab ”.
2. After that, I have to select its child object “SoundPocket” and configure the *SoundPocket* class as I need.

I will also add my clip to the "Music Clips" section. If you are not sure what type of sound you need, choose “Sound Clips”, read the manual for more details **Eazy Sound Manager**.

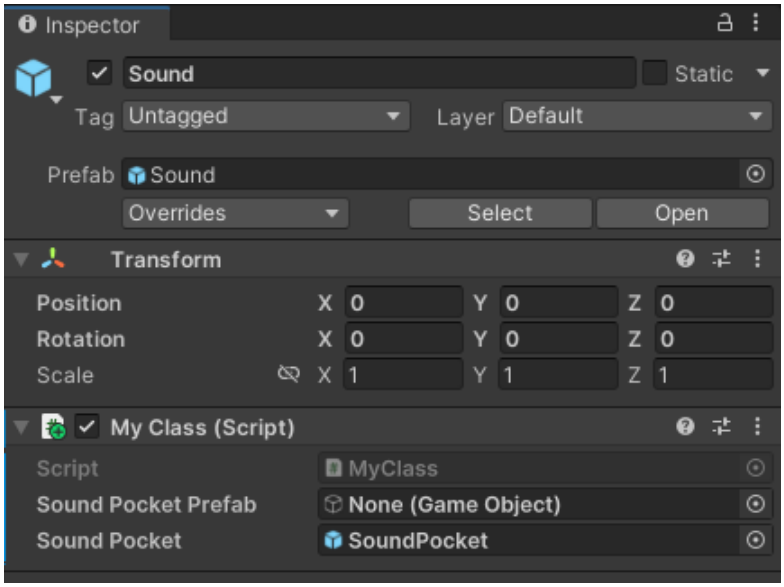
3. Now I will create a new class “MyClass”, connect the namespace “Lineri.SoundSystem” and will inherit it from SoundPocketManager.

```
using Lineri.SoundSystem;
using UnityEngine;

public class MyClass : SoundPocketManager
{

}
```

4. Since the *SoundPocket* instance that I will use has “Loop Clips” and “Play Sound On Awake” enabled, the sound will play all the time from the moment this scene starts, so I don't need to call the “PlayHandler()” method in my script.
5. I want to be able to stop and resume audio playback. To do this, I will write the following structure in the Update method:

```
private void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            PauseHandler();
        }
        if (Input.GetKeyUp(KeyCode.Space))
        {
            UnPauseHandler();
        }
    }
```

6. Now, when you press the space bar, the sound playback will stop, and when you release the space bar, the sound will continue playing again.
7. I will add my class to the “Sound” object and in the “Sound Pocket” field of my class, I will put a link to my previously configured “SoundPocket” object.



8. I start the scene for verification. Everything works, the sound plays the way I need it, when you press the space bar, playback stops, and when you release the space bar, it continues again.
9. You can use the other methods from the “Handlers” region of the *SoundPocketManager* class in the same way. The methods are described in detailed comments, which are constantly updated. You can also override methods in your class to create complex or non-standard logic.
10. This knowledge will be enough for you to use **SoundSystem**, but I also recommend reading the **Eazy Sound Manager** manual to have a more complete understanding of the capabilities of this bundle of tools.
11. In addition, I want to say that if you have any questions or have any suggestions for improving the **SoundSystem**, including improvements to the code itself or related documentation, as well as any other suggestions, you can always contact me on GitHub, describing your situation in detail.

Using Steam Audio

1. Import Steam Audio into your project.
2. Create an object and add the *SoundPocketSteamAudio* class to it. In the “Audio Source” field, place the desired AudioSource on which the clips will be played. Note that the AudioSource settings will be used to play the clip, except for the settings: Clip, Volume, Loop, Pitch.

The *SoundPocketSteamAudio* class is intended only for playing clips in a queue, it is not possible to play all clips at once. The AudioSource specified in the class field will not be destroyed, can be located on any of the objects and used by the Steam Audio toolkit. Also, when changing the scene, the sound will not continue even with the appropriate setting enabled, if the AudioSource is destroyed, the sound will not be played.