Sound System

Это полное руководство по использованию инструмента **Sound System** в Unity. Актуально для версии инструмента 0.1.0.1.

Qr код со ссылкой на репозиторий Sound System:

Текстовая ссылка на репозиторий Sound System:

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

Текстовая ссылка на репозиторий Eazy Sound Manager:

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

Содержание

Sound System 1
Содержание 2
Перед тем как начать 3
Принцип работы 4
Начало работы 6

Перед тем как начать

1. Скачайте последнею версию **Sound System** из раздела релиз.

2. Если вы не устанавливали плагин ранее, то выполните 3 шаг, в случае если же у вас уже установлен плагин и используется в проекте, пропустите шаг 3 и перейдите к 4 шагу.

3. Импортируйте пакет в свой проект. В Unity это выглядит так: Assets – Import Package – Custom Package – выбирайте версию которую вы скачали. Переходите к следующему разделу руководства.

4. В случае если у вас уже был установлен инструмент и он использовался в проекте, вам необходимо убедиться что новая версия, которую вы хотите установить, совместима с вашей текущей версии.

5. Для старых релизов (до версии 0.0.1.0 включительно) это указано непосредственно в самом релизе в разделе “Upgrade from the previous version:”. Узнать версию плагина которая используется в проекте можно с помощью файла “Version.txt”, который находится по пути: “Assets\Lineri\SoundSystem\SoundSystem\Docs\Version\Version.txt”.

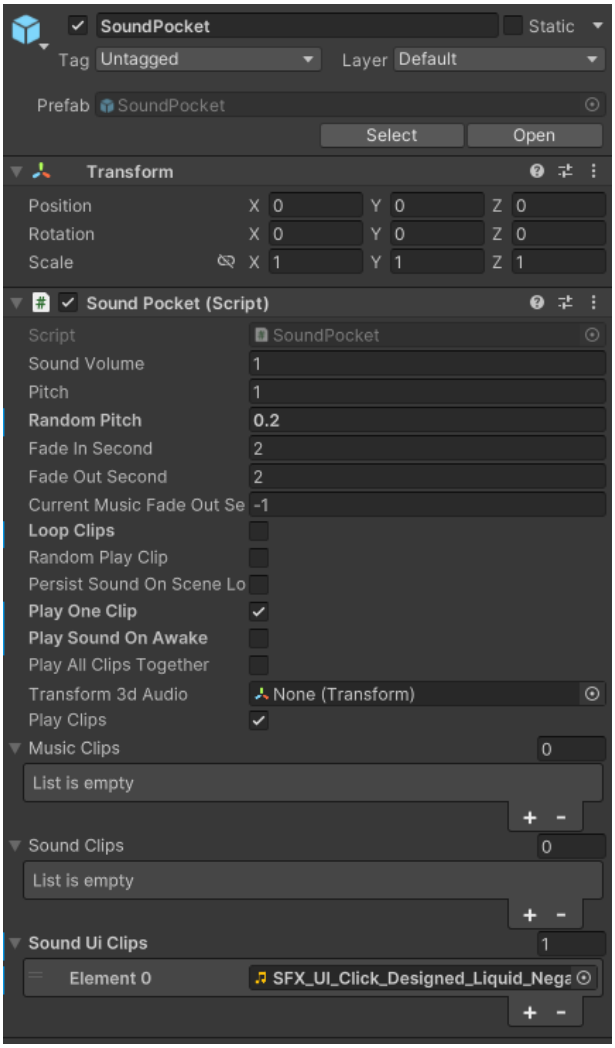
6. Для новых релизов используется дополненная система Semantic Versioning, поэтому когда версия релиза X.Y.Z.W вы сможете без проблем обновиться если версия в вашем проекте идентична разрядами X.Y. То есть с версии 1.0.0.0 можно обновиться на версию 1.0.0.15 или на версию 1.0.2.0, но нельзя обновиться на версию 1.1.0.0 или 2.0.0.0.

7. Если после проверки вы убеждены что версии совместимы, выполните шаг **3**.

Принцип работы

Sound System – простой и быстрый инструмент который призван разделить обязанности при создании звуковой схемы игры, а так же существенно ускорить этот процесс. Разделение обязанностей достигается при помощи удобного редактирования настроек группы клипов прямо из инспектора. Любой человек без знаний кода может создать префаб с одним или несколькими классами “SoundPocket”, установить необходимые настройки и проверить работоспособность или создать простые звуковые схемы. В случае если необходимо более сложное взаимодействие этот префаб можно передать программисту который в это время уже создаст необходимую логику под ваши нужды. То есть, с момента постановки задачи, программист и дизайнер могу начать работать параллельно и не завесить от результатов друг друга.

В результате работы дизайнера получается префаб с необходимыми настройками *SoundPocket*:



В результате работы программиста мы получаем класс который наследуется от *SoundPocketManager* и реализовывает необходимую нам логику:

```
using UnityEngine;
using Lineri.SoundSystem;

public class YourClass : SoundPocketManager
{
    private bool _ignoreInvoke = false;

    override protected void Start()
    {
        PlayHandler();
    }

    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}");
        }
    }
}
```

И в итоге мы получаем настроенный префаб со звуком, а так же класс который может полностью управлять этим звуком так как нам нужно. Теперь достаточно добавить этот класс к любому объекту, в нашем случае понадобится ещё добавить префаб на сцену и указать ссылку на наш объект который мы добавили из префаба.

Начало работы

1. Для начала я добавлю на сцену префаб “Sound”, который находится по пути: “Assets\Lineri\SoundSystem\SoundSystem\Prefabs\Sound.prefab ”.
2. После я должен выбрать его дочерний объект “SoundPocket” и настроить класс *SoundPocket* так как мне нужно.

Так же я добавлю свой клип в раздел “Music Clips”. Если вы не уверены какой тип звука вам нужен, выбирайте “Sound Clips”, для больших подробностей ознакомитесь с руководством **Eazy Sound Manager**.

3. Теперь я создам новый класс “MyClass”, подключу пространство имён “Lineri.SoundSystem” и унаследую его от SoundPocketManager.

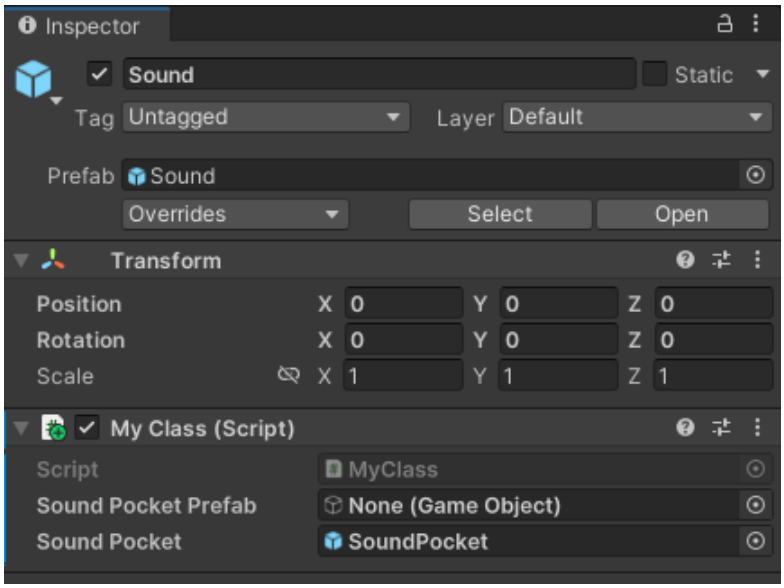
```
using Lineri.SoundSystem;
using UnityEngine;

public class MyClass : SoundPocketManager
{

}
```

4. Так как у экземпляра *SoundPocket* который я буду использовать включены “Loop Clips” и “Play Sound On Awake”, то звук будет играть всё время с момента запуска этой сцены, поэтому мне не нужно вызывать метод “PlayHandler()” в своём скрипте.
5. Я хочу иметь возможно останавливать и возобновлять воспроизведение звука. Для этого в методе Update я напишу такую структуру:

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

6. Теперь при нажатии на пробел воспроизведение звука будет останавливаться, а при отпускании пробела звук снова продолжит воспроизведения.

7. Я добавлю свой класс на объект “Sound” и в поле “Sound Pocket” моего класса, помещу ссылку на свой ранее настроенный объект “SoundPocket”.



8. Запускаю сцену для проверки. Всё работает, звук воспроизводится так как мне нужно, при нажатии на пробел воспроизведение останавливается, а при отпускании пробела снова продолжается.
9. Вы таким же образом можете использовать остальные методы из региона “Handlers” класса *SoundPocketManager*. Методы описаны развёрнутыми комментариями, которые постоянно обновляются. Вы так же можете переопределять методы в своём классе для создания сложной или нестандартной логики.
10. Этих знаний вам будет достаточно для использования **SoundSystem**, но я так же рекомендую ознакомится с руководством **Eazy Sound Manager**, чтобы иметь более полное представление возможностей данной связки инструментов.