Лабораторная работа №12. Работа со звуком, создание панели управления (UI) в Unity

Раздел I. Работа со звуком

1. Для добавления звуковых эффектов нужно записать или найти звуковые файлы, которые соответствуют звукам выстрела, передвижения танка, окрашивания цели при попадании в неё и возврата цвета цели к исходному цвету, после чего перенести их в папку Assets.

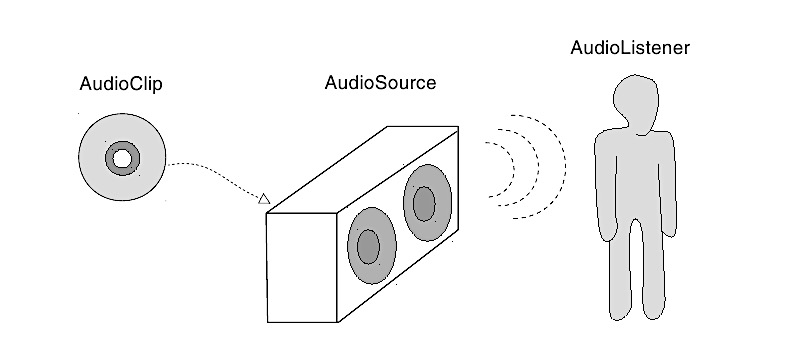


Рисунок 1 – Принцип работы звука в Unity

1. На источники звука (цели, снаряд, танк) поместить компонент **AudioSourse**. В инспекторе на всех компонентах отключить галочку **Play On Awake** (кроме снаряда, т.к. он должен произвести звук при его создании (выстреле), чтобы звук не проигрывался сразу при загрузке сцены). И поставить галочку **Loop** на танке, чтобы звук проигрывался циклически. Далее нужно добавить аудио клипы на соответствующие источники (кроме целей, на них клипы будут добавлены программно). Слушатель по умолчанию уже находится на камере.
2. Теперь нужно проиграть данные аудио клипы на соответствующих источниках при определенных событиях в игре. События: танк поехал, танк остановился, выстрел, смена цвета цели при попадании, смена цвета цели спустя 3 секунды после попадания.
3. Разберемся с движением и остановкой танка. Сперва объявите переменные для источника **AudioSource source\_tank;** в скрипте **«Tank»** и переменную, которая хранит состояние запущен ли звук танка (изначально не запущен) **bool isPlaying = false;**
4. В метод **Start ()** добавьте инициализацию переменной **source\_tank**

source\_tank = GetComponent<AudioSource>();

1. Добавьте в метод **Update ()** проверку через условный оператор **if** если танк двигается и звук не включён, то включите проигрывание источника звука на танке и отметьте через булевскую переменную, что звук включен:

if ((Input.GetAxis("Horizontal") != 0 || Input.GetAxis("Vertical") != 0) && !isPlaying) {  
            source\_tank.Play();   
            isPlaying = true;   
        }

1. Создайте также условие на проверку остановлен ли танк и играет ли звук, в котором выключитепроигрывание звука на танкеи отметьте, что звук выключен:

if (Input.GetAxis("Horizontal") == 0 && Input.GetAxis("Vertical") == 0 && isPlaying) {  
            source\_tank.Stop();  
            isPlaying = false;  
        }

1. Проверьте проигрывание звука при движении танка.
2. Для добавления звука смены цвета (скрипт **«Trigger»**) нужно указать, что это поле можно заполнять из инспектора с помощью атрибута **[SerializeField]** и на следующей строке объявить переменную для звука попадания в цель **AudioClip clipIN;**.
3. Аналогично проделайте для звука смены цвета спустя 3 секунды (переменная **clipOUT**).
4. В методе **OnCollisionEnter** в условии столкновения подставите звуковое сопровождение при столкновении с целью:

GetComponent<AudioSource>().PlayOneShot(clipIN);

1. Также в корутину **Scaler(),** после поставки переменной **scale** в значение **false** и перед удалением объекта добавите воспроизведение запуска звука **clipOUT.**
2. Измените метод **Destroy()** в корутине **Scaler(),** добавив второй параметр, который будет содержать длительность аудиоклипа в секундах, чтобы предотвратить удаление объекта-цели до проигрывания звука. Таким образом, следующий листинг представляет корутину **Scaler()**:

IEnumerator Scaler()  
    {  
        scale = true;  
        yield return new WaitForSeconds(2f);  
        scale = false;  
        GetComponent<AudioSource>().PlayOneShot(clipOUT);  
        Destroy(gameObject, clipOUT.length);  
    }

1. Сохраните скрипт и перетяните советующие аудио в инспектор компонента скрипта объектов-целей на сцене (рисунок 2).

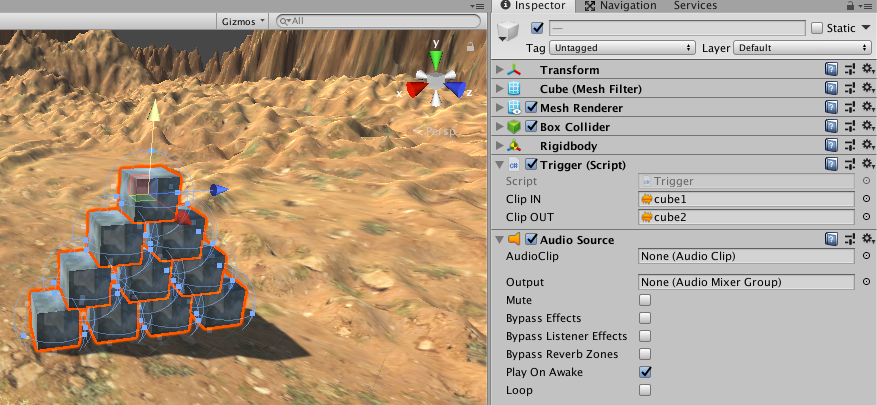


Рисунок 2 – Вкладка Inspector для объектов-целей

1. Проверьте проигрывание звука при смене цвета и удалении объектов-целей.

Раздел II. Работа с пользовательским интерфейсом (UI)

1. Создайте панель управления (**UI**) состоящую из **Canvas** (полотно), **Panel**, 2-ух слайдеров с подписями (**Text**: «скорость танка:» и «скорость снаряда») и одну кнопку для открывания/скрывания панели. Кнопку расположите за пределами панели (рисунок 3). Обратите внимание, что при создании **Canvas** для взаимодействия с UI элементами посредством мыши в иерархии объектов создаётся система событий **EventSystem**.

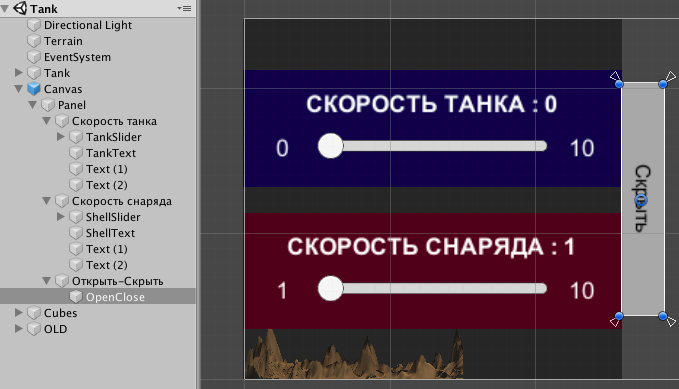


Рисунок 3 – Панель управления танком

1. В скрипт на танке **«Tank»** создайте новый публичный метод **ChangeTankSpeed (float value),** который принимает параметрзначения скорости танка для его изменения и назначьте танку скорость, соответствующую значению слайдера **MoveSpeed = value/100** (деление на 100, т.к. танк слишком быстро двигается). В объявлении переменную **MoveSpeed** поставьте в значение ноль.

public void ChangeTankSpeed (float value) { MoveSpeed = value / 100; }

1. В инспекторе слайдера укажите свойства, соответствующие минимальному и максимальному значению скорости (например: 0 и 10). Также в инспекторе нужно указать начальное значение слайдера (0) и метод, который будет вызываться при возникновении события смены значения слайдера (**ChangeTankSpeed**), для этого нужно указать объект, на котором находится скрипт, содержащий нужный метод (**corpus**), затем выбрать название скрипта (**Tank**) и только потом название метода, обязательно в списке **Dynamic float**, чтобы можно было передавать значение (рисунок 4). Установите значение **Navigation** в **None**. Поставьте галочку **Whole Numbers** чтобы выводились только целые числа.

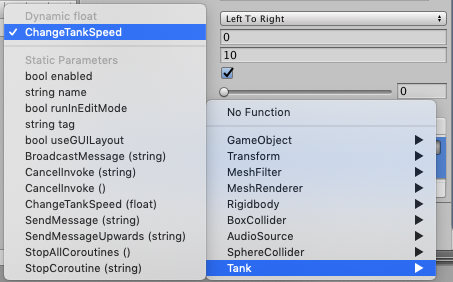
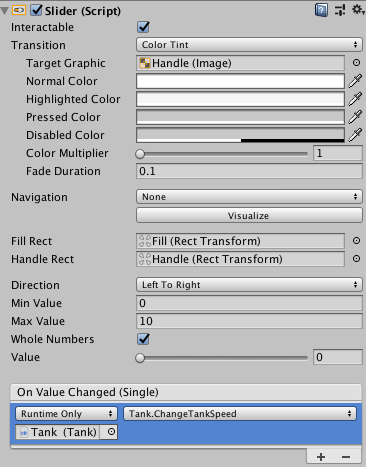
 

Рисунок 4 – Задание слайдеру метода

1. Аналогичные действия проделайте для манипуляции над скоростью снаряда, только задайте минимальные и максимальные значения в слайдере от 1 до 10.
2. Подключите директиву **UnityEngine.UI** в скрипт **«Tank»** и в скрипт на ядре **«Shell»**, чтобы можно было взаимодействовать с пользовательским интерфейсом, а именно изменять текст.
3. В скрипте **«Tank»** объявите две переменные (**tankText** и **shellText**) типа **Text** для изменения текста слайдера танка. Проинициализируйте эти переменные через инспектор с помощью атрибута  **[SerializeField]**. После сохранения скрипта перетяните советующий текст в инспектор танка на сцене (рисунок 5).

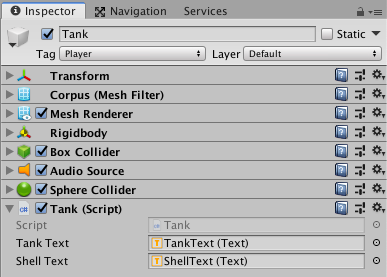


Рисунок 5 – Определение текстовых переменных

1. Добавьте в метод изменения скорости танка для изменения текста, таким образом листинг метода **ChangeTankSpeed()** будет следующим:

public void ChangeTankSpeed (float value)   
    {  
        MoveSpeed = value / 100;  
        tankText.text = "СКОРОСТЬ ТАНКА:" + value;  
    }

1. Также в скрипте **«Tank»** создайте публичный метод **ChangeShellSpeed().**

public void ChangeShellSpeed(float value)  
    {  
        shellText.text = "СКОРОСТЬ СНАРЯДА : " + value;  
    }

1. Так как скрипт **«Shell»** находится на префабе (объект **shell**), а не на объекте, который на сцене, то мы не можем проинициализировать его поля через префаб, поэтому для изменения текста слайдера приходится производить поиск нужного объекта-текста непосредственно перед его изменением. Поэтому сделайте получение значения скорости снаряда из слайдера, добавив в скрипте **«Shell»** в метод **Start()** следующий код:

shellSpeed = GameObject.Find("ShellSlider").GetComponent<Slider>().value;

1. Запустите проект и проверьте, что при запуске танк не может двигаться, так как его скорость равна нулю, и только при изменении слайдера его скорость изменится, а также измениться текст с выводом текущей скорости. Аналогично должен работать слайдер со скоростью полёта снаряда.
2. Создайте скрипт **PanelMove,** в котором подключите директиву **UnityEngine.UI**. В самом классе объявите переменную **RectTransform UIGameobject**, которая содержит ссылку на компонент, которая отвечает за положение объекта.
3. Проинициализируйте текстовую переменную **btnText** для кнопки с помощью **[SerializeField]**. Создайте переменные типа **float** для значения ширины панели **width**, для значения содержащее смещение панели **changeX** и для значения скорости закрытия панели **speedPanel**.
4. Создайте перечисление состояние панелей через **enum states**, который содержит в себе следующие состояния: **open, close, opening, closing**.
5. Изначально установите состояние открытым **states state = states.open;**.
6. В методе **void Start()** проинициализируйте переменную трансформа с ссылкой на компонент, инициализируйте ширину панели и установите скорость её закрытия:

void Start()  
    {  
        UIGameobject = gameObject.GetComponent<RectTransform>();   
        width = UIGameobject.sizeDelta.x;    
        speedPanel = 3;   
    }

1. В методе **void Update()** создайте условие нахождение панели в состоянии закрытия и в данном условии возьмите текущую позицию по X и текущую позицию по Y. Так как нам необходимо закрывать панель по горизонтали (у вас может быть по вертикали, тогда необходимо работать с позицией по Y), установите изменение значение X с заданной скоростью. После этого запишите в переменную **changeX** на сколько сдвинулась панель с заданной скоростью и по итогу примените новое положение панели.

if (state == states.closing)  
        {  
            float x = UIGameobject.anchoredPosition.x;  
            float y = UIGameobject.anchoredPosition.y;   
            x -= speedPanel;  
            changeX += speedPanel;  
            UIGameobject.anchoredPosition = new Vector2(x, y);        }

1. В этом же условии (если панель закрывается) создайте новое условие, при котором если значение смещения панели больше её ширины то изменяется текст и указывается закрытое состояние панели:

if (changeX > width)  
            {  
                btnText.text = "Открыть";  
                state = states.close;  
            }

1. В методе **void Update()** создайте ещё одно условие нахождение панели в состоянии открытия. В данном условии также возьмите текущие позиции по Х и по Y. Так как нам необходимо открывать панель по горизонтали, установим изменение значение Х с заданной скоростью **x+=speedPanel;**. И после этого запишите в переменную **changeX**на сколько сдвинулась панель с заданной скоростью**.** По итогу примените новое положение панели.
2. В этом же условии (если панель открывается) создайте новое условие, при котором если значение смещения панели больше её ширины происходит изменение значение текста на **"Скрыть"** и указывается открытое состояние панели.
3. Создайте публичный метод **ChangePanel(),** который будет вызываться при щелчке на кнопку. В нём пропишите условие, при котором если панель открыта, активно состояние закрытия и наоборот. Инициализируйте обнулением **changeX.**

public void ChangePanel()  
    {  
        if (state == states.open) state = states.closing;  
        if (state == states.close) state = states.opening;  
        changeX = 0;  
    }

1. Разместите скрипт **PanelMove** на панель. На переменную **btnText** в инспекторе перетяните объект со сцены отвечающий за текст кнопки.

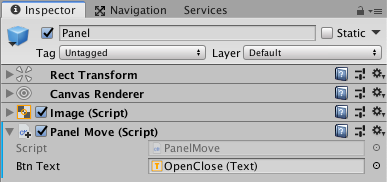


Рисунок 6 – Определение переменной btnText

1. В событии **OnClick** кнопки открытия/закрытия панели перетянуть саму панель, укажите данный скрипт и метод **ChangePanel()** (рисунок 7).

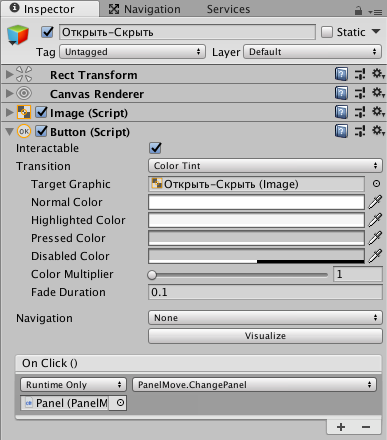


Рисунок 7 – Определение кнопки

1. Запустите игру и проверьте работу панели, изначально панель должна быть открыта, а при нажатии кнопки «Скрыть» начать плавно уезжать влево и текст должен измениться на «Открыть».