Лабораторная работа №4. Работа со звуком, реализация панели управления (UI)

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.

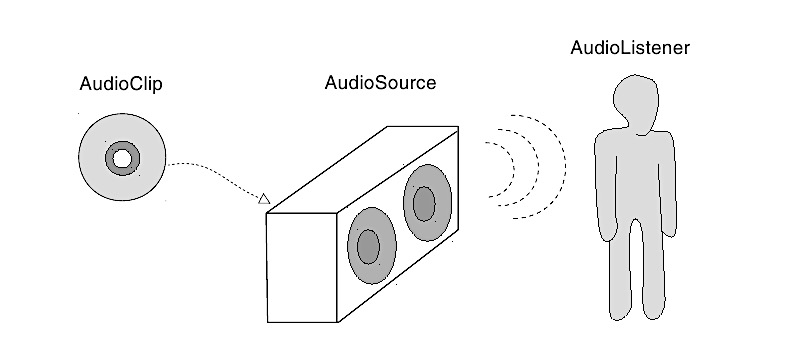
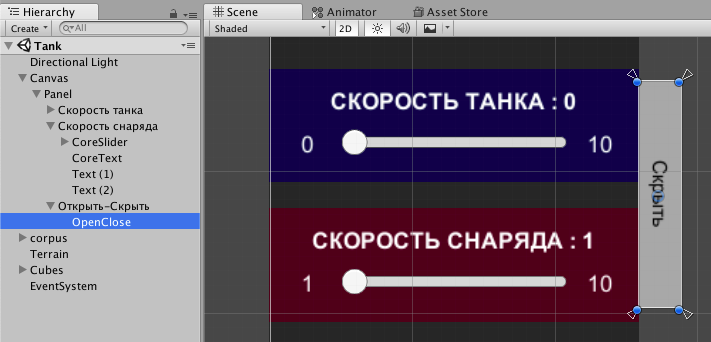


Рисунок – Принцип работы звука в Unity

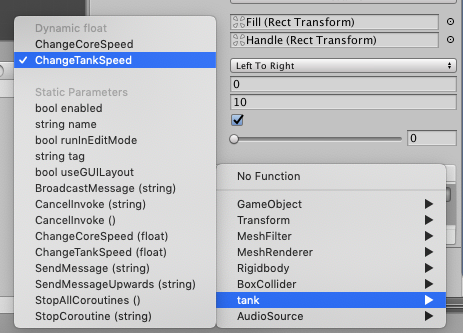
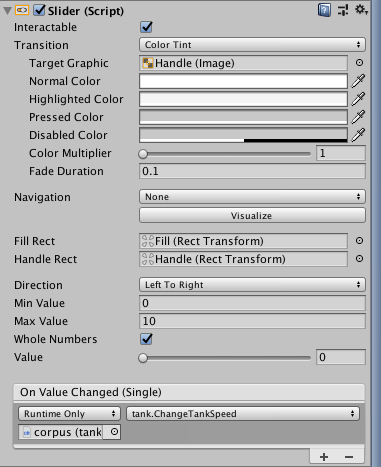
1. На источники звука (цели, снаряд, танк) поместить компонент **AudioSourse**. В инспекторе на всех компонентах отключить галочку **Play On Awake** (кроме снаряда, т.к. он должен произвести звук при его создании (выстреле), чтобы звук не проигрывался сразу при загрузке сцены. И поставить галочку **Loop** 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 танка и переменную, которая хранит состояние запущен ли звук танка (изначально не запущен) **bool isPlaying = false;**
4. В метод **Start ()** добавьте инициализацию **source\_tank = GetComponent<AudioSource>();**
5. Добавьте в метод **Update ()** условие если танк двигается и звук не включён: **if ((Input.GetAxis("Horizontal")!=0 || Input.GetAxis("Vertical")!=0) && !isPlaying)** в котором  
   включите проигрывание источника на танке **source\_tank.Play();** и отметьте, что звук включен **isPlaying = true;**
6. Создайте ещё условие на проверку остановлен ли танк и играет ли звук: **if (Input.GetAxis("Horizontal")==0 && Input.GetAxis("Vertical") == 0 && isPlaying)** в котором выключитепроигрывание звука на танке **source\_tank.Stop();** и отметьте, что звук выключен **isPlaying = false;**
7. Проверьте проигрывание звука про движении танка.
8. Для добавления звука смены цвета (скрипт **trigger.cs**) 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;**.
9. Аналогично проделайте для звука смены цвета спустя 3 секунды (переменная **clipOUT**).
10. В функции **OnCollisionEnter** в условии столкновения подставить звуковое сопровождение при столкновении с целью **GetComponent<AudioSource>().PlayOneShot(clipIN);**
11. Также в корутину, когда цвет через 3 секунды возвращается к белому добавить произведение запуска звука возврата цвета **GetComponent<AudioSource>().PlayOneShot(clipOUT);**
12. Сохраните скрипт и перетяните советующие аудио в инспектор компонента скрипта объектов-целей на сцене.



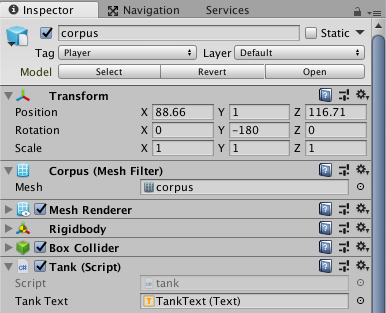
1. Проверьте проигрывание звука при смене цвета объектов-целей.
2. Создайте панель управления (**UI**) состоящую из **Canvas** (полотно), **Panel**, 2-ух слайдеров с подписями (**Text**: «скорость танка:» и «скорость снаряда») и одну кнопку для открывания/скрывания панели. Обратите внимание, что при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



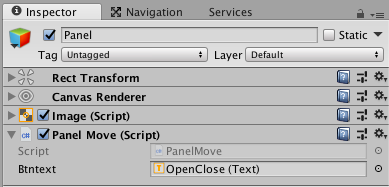
1. В скрипт на танке добавим метод с параметром нового значения скорости танка для смены скорости танка **public void ChangeTankSpeed (float value)** и назначим танку скорость, соответствующую значению слайдера **MoveSpeed = value/100** (деление на 100, так как танк слишком быстро двигается);
2. В инспекторе слайдера укажем свойства, соответствующие минимальному и максимальному значению скорости (например: 0 и 10). Также в инспекторе нужно указать начальное значение слайдера (0) и функцию, которая 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. Установите значение **Navigation** в **None**. Поставьте галочку **Whole Numbers** чтобы выводились только целые числа.

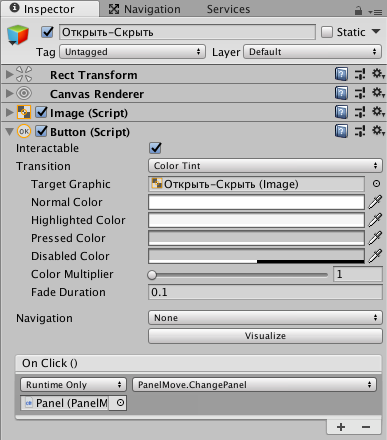
1. Аналогичные действия проделайте для манипуляции над скоростью снаряда, только задайте минимальные и максимальные значения в слайдере от 1 до 10.
2. Подключите директиву **UnityEngine.UI** в скрипт **tank.cs** и в скрипт на ядре **core.cs**, чтобы можно было взаимодействовать с пользовательским интерфейсом, а именно изменять текст.
3. В скрипте **tank.cs** объявите переменные типа **Text** для изменения текста слайдера танка. Проинициализируйте эту переменную в старте через поиск или через инспектор с помощью атрибута  **[SerializeField]**. После сохранения скрипта перетяните советующий текст в инспектор танка на сцене.



1. Добавьте в функцию изменения скорости танка для изменения текста следующее: **TankText.text = "Скорость танка:" + value;**.
2. Проделайте аналогичные действия для слайдера изменения скорости снаряда в скрипте **core.cs.** Но инициализацию текста для вывода значения скорости **CoreText = GameObject.Find( "CoreText" ).GetComponent<Text>();** проведите внутри метода **ChangeCoreSpeed().** Так как скрипт **core.cs** находится на префабе (объект **core**), 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.
3. Также в скрипт **core.cs** в метод **Start ()** добавьте присвоение значения из слайдера **coreSpeed = GameObject.Find("CoreSlider").GetComponent<Slider>().value;**
4. 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.
5. 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**.
6. Создайте перечисление состояние панелей через **enum states**, который содержит в себе следующие состояния: **open, close, opening, closing**.
7. Изначально установите состояние открытым **states state = states.open;**.
8. В функции **void Start()** проинициализируйте переменную трансформа с ссылкой на компонент **UIGameobject = gameObject.GetComponent<RectTransform>();.** Инициализируйте ширину панели **width = UIGameobject.sizeDelta.x;** и установите скорость её закрытия **speedPanel = 3;**.
9. В функции **void Update()** создайте условие нахождение панели в состоянии закрытия **state == states.closing**.
10. В данном условии возьмите текущую позицию по *x* **float x = UIGameobject.anchoredPosition.x;** и текущую позицию по *y* **float y = UIGameobject.anchoredPosition.y;**.
11. Так как нам необходимо закрывать панель по горизонтали (у вас может быть по вертикали, тогда необходимо работать с позицией по *y*), установим изменение значение *х* с заданной скоростью **x-=speedPanel;**. После этого запишите в переменную changeX на сколько сдвинулась панель с заданной скоростью **changeX += speedPanel;**.
12. По итогу примените новое положение панели **UIGameobject.anchoredPosition = new Vector2(x, y);**.
13. В этом же условии создайте новое условие, при котором если значение смещения панели больше её ширины **changeX > width** происходит изменение значение текста **btnText.text = "Открыть";** и указывается закрытое состояние панели **state = states.close;**.
14. В функции **void Update()** создайте ещё одно условие нахождение панели в состоянии открытия **state == states.opening**. В данном условии также возьмите текущие позиции по *x* и по *y*.
15. 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.
16. В этом же условии создайте новое условие, при котором если значение смещения панели больше её ширины **changeX > width** происходит изменение значение текста **btntext.text = "Скрыть";** и указывается открытое состояние панели **state = states.open;**.
17. Создайте публичный метод **void ChangePanel(),** который будет вызывать на кнопку. В нём пропишите условие, при котором если панель открыта, активно состояние закрытия **if (state == states.open) state = states.closing;**. И наоборот **if (state == states.close) state = states.opening;**. Инициализируйте обнулением **changeX = 0;.**
18. Разместите скрипт **PanelMove** на панель. На переменную **btntext** в инспекторе перетяните объект со сцены отвечающий за текст кнопки.



1. В событии **OnClick** кнопки открытия/закрытия панели перетянуть саму панель, указать данный скрипт и метод **ChangePanel**.



1. Запустить игру и проверьте работу всего реализованного кода.