**Creating your own components**

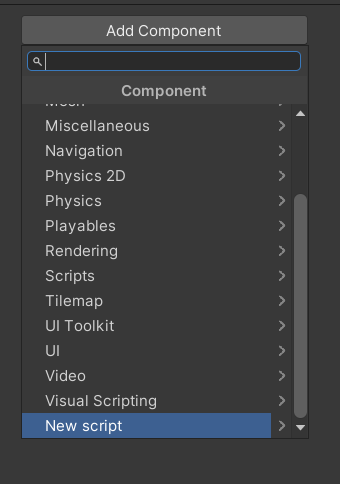
Для начала нужно перейти в *Edit -> Preferences -> External tools*

Тут мы проверим правильное подключение редактора кода с движком



Что бы создать наш компонент, необходимо выбрать объект

Я выберу куб

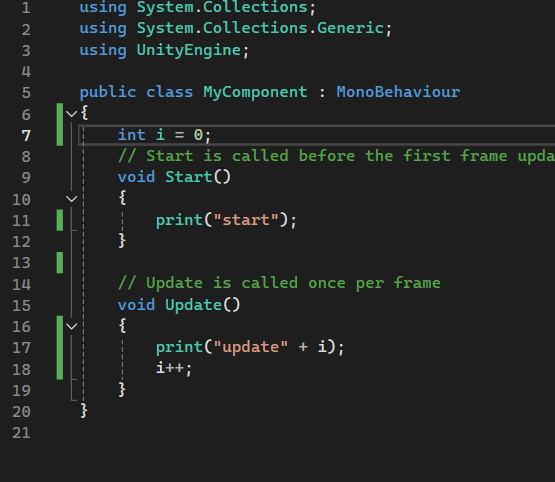


Этот скрипт мы сможем найти в наших папках, двойной клик по скрипту быстро нам покажет его расположение

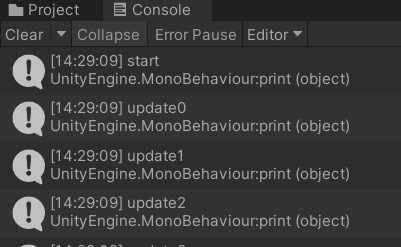
Далее открываем скрипт

* *Функция Start* будет запускаться в самом начале игры (самый первый кадр игры) это будет один раз в начале
* *Функция Update* - вызывается один раз за кадр (в Unity игры будут пытаться работать в 60 кадров/сек) это будет повторяться сново и сново

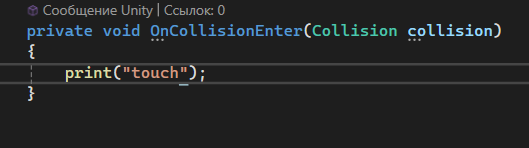
Небольшой пример, будем отлаживать наш код(debug)



В Unity мы можем просматривать эти сообщения в окне Console



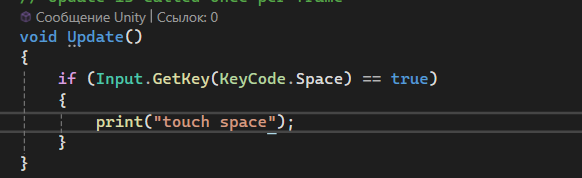
Далее рассмотрим функцию которая выполняется, когда физическое тело касается или сталкивается (onCollitionEnter).



Когда я буду запускать симуляцию, данная функция будет отправлять сообщение

**Conditions if**

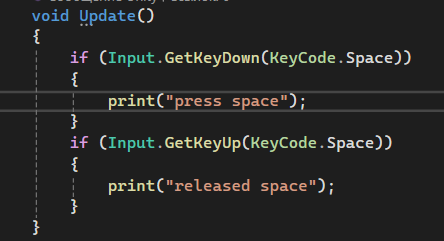
Сделаем условие внутри Update, так нам нужно будет проверять всегда это событие



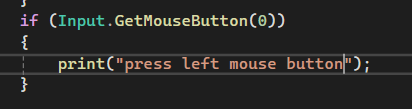
**Simple inputs**

Позволяет нам использовать клавиатуру/мышь для отправки сигналов.

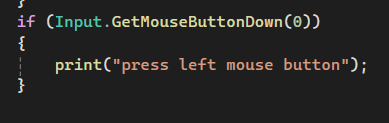
Будем отслеживать нажатие клавиши и когда мы будет отпускать клавишу, сообщения будут печаться лишь по одному разу, в момент нажатия и в момент когда отпускаем.



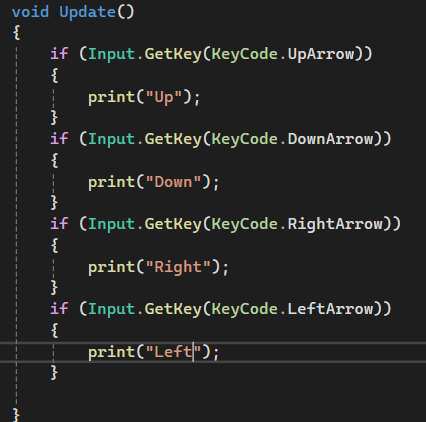
Так же можем прослушать нажатие кнопок мышки, будет выводиться куча сообщений



Тут будет выводиться только один раз сообщение, так как он отслеживает только нажатие



Далее получим доступ к клавишам перемешения

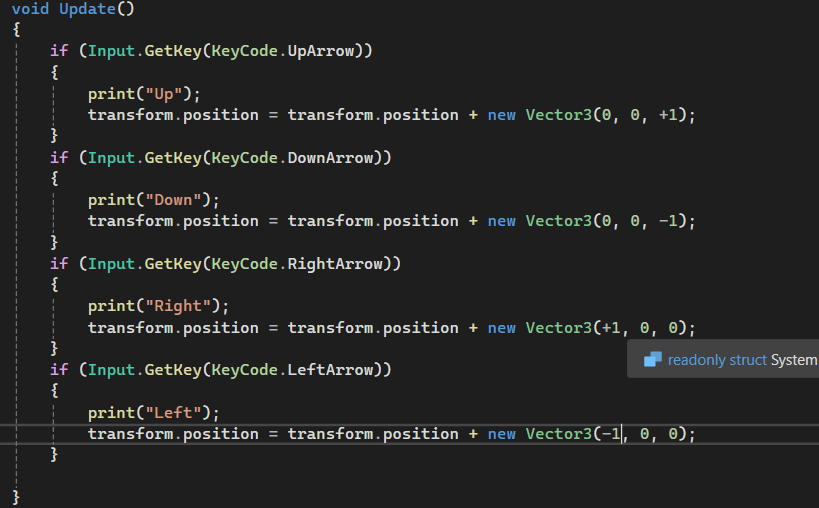


**Accessing Component**

Будем соединят наши собственные компоненты с другими компонентами, это позволит нам использовать код для доступа к информации и ее изменения

*Transform component*

Для начала что бы установить связь между нашим компонентом и компонентами объекта, для этого нам идеально подходит функция start, для того что бы все настроить, потому что мы собираемся настрить ссылки на другие компоненты. Но для *transform* нам беспокоиться не нужно так как это всегда доступно, этот компонент являеться обязательным для всех.

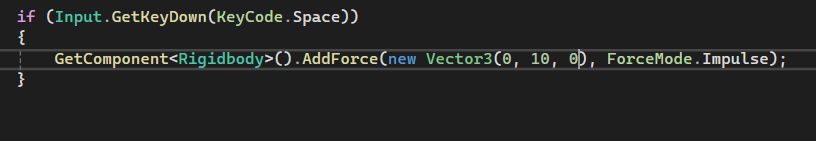


Доступ к другим компонентам объекта

Получам доступ к компоненту Rigibody c помощью



Данный кусок кода будет прослучишавть нажатие Space и поднимать наш предмет по Y, своего рода прыжок с силой 10

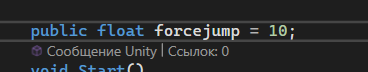


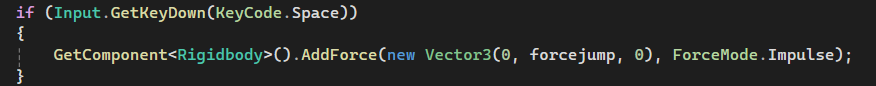
Можно получить к компоненту Collider

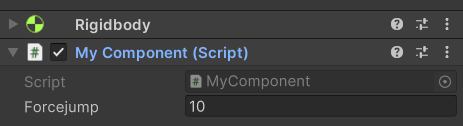


**Code Variables**

Не очень верно давать числовые значени, лучше их сохранять в переменные и использовать, так как переменную мы объявляем в начале всего кода и ее легче найти и поменять ей значение если, к примеру, нас не будет устраивать сила прыжка. Даем общий доступ, мы сможем изменять значение переменной непосретсвенно из движка не заходя в скрипт

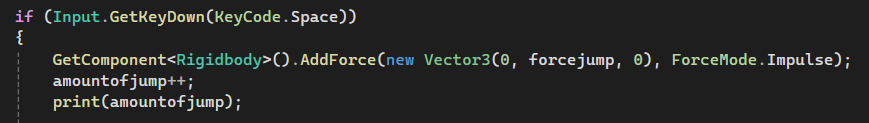






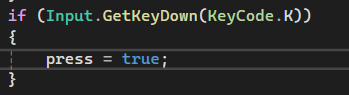
Так же будем делать подсчет прыжков



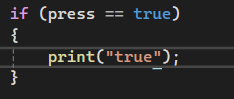


Будем использовать переменные типа bool



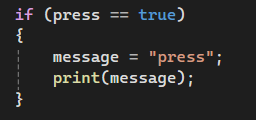




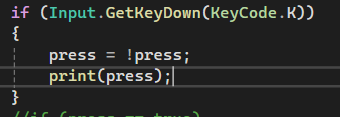


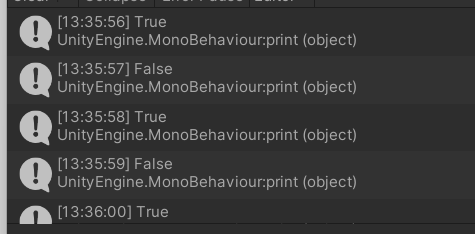
Строковую переменную тоже важно использовать





небольшая, интресная запись с bool





Так же наша переменая будет реагировать на нажатие клавиши



**Handling Errors**

К примеру если у нас есть какие то синтактические ошибки, то юнити нам см об это скажет в console и мы сможем через двойной клик получить доступ в то место где нужно что то подправить

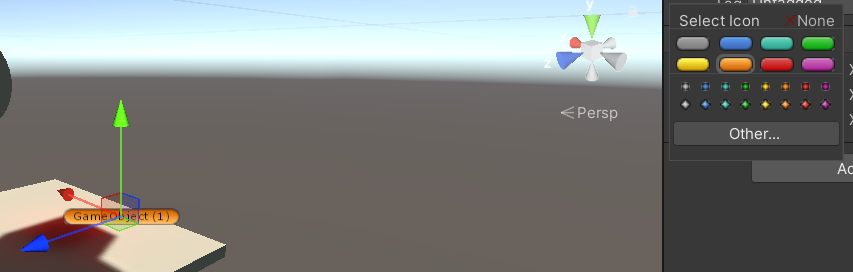


**Spawning and destroying objects**

Для этого создадим новый Script, я бы хотела создать пременную которая будет ссылаться на объект в scene, так же public что я свободно могла закинуть туда объект из сцены, обычно это добавляется на пустышку и после на скипт закидывается любой объект





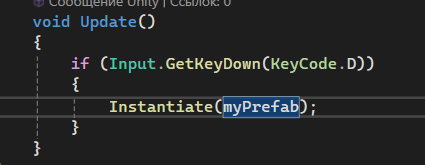




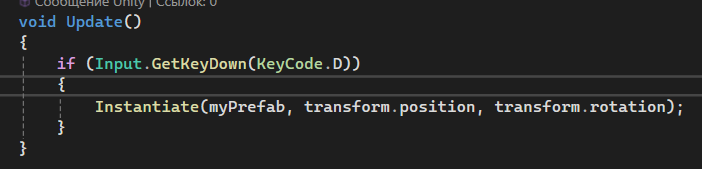
Теперь наш компонент ссылается на объект которго даже нет в сцене

Структура которая заставляет появляться объекты *instantiate*

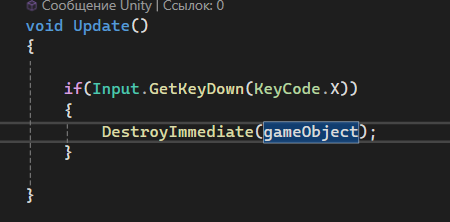
мы ссылаемся на нашу пременную myPrefab, выбирается случайное место положение и вражение при создании объекта.



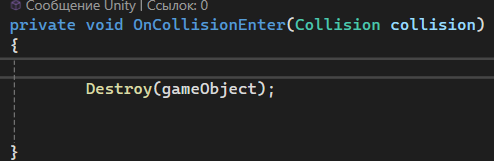
Но если мы будем получать компонент transform, то наш объект будет появляться в области того объекта куда был помещен скрипт (так как мы получали его компонент transform)



Но для того что бы удалить появившийся объект из сцены, нам нужен новый скрипт который будет на этом обхекте (myPrefab) на который мы ссылаемся в появлении



Но так же мы можем удалить объект если он будет с чем то сталкиваться, на объекте обязательно должен быть Collider (эту функцию мы не пишем в update, она существует отдельно), Что бы эта функция работала, объекту необходимо иметь Riggibody, сталкивается с другими объектами



Мы имеем пустышку которается на объект что бы его создать, а в свою очередь объект имеет скрипт уничтожения

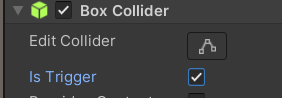
**Creating triggers**

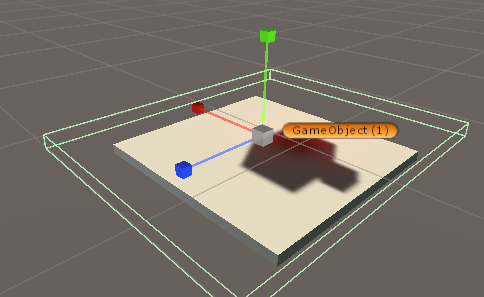
Это очень похоже на то, что происходит при Collition

Что бы эта функция работала, объекту нужно иметь Riggibody, нужно просто войти в триггерную зону

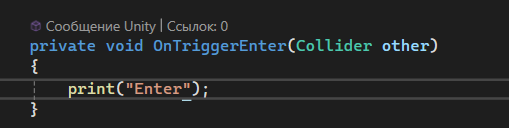
Для начала нам нужно создать *trigger zone*

На объекте(который будет триггерной зоной/областью) есть компонент Collider на нем нужно активировать *Is trigger ,* так когда герой будет появляться в этой зоне мы сможем с помощью кода это определять и делать разные действия (к примеру, скримеры), и если на этом объекте мы отключим компонент *Mesh Renderer*, то он будет не видем

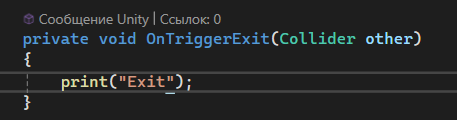




И далее мы помещаем следующий код на объект который будет входит в зону триггера



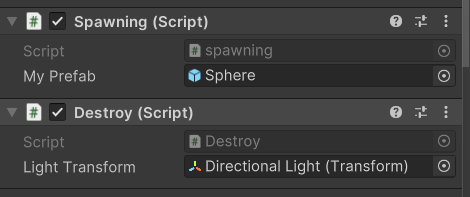
Далее функция выхода из зоны триггера



Сделаем небольшой пример, когда объект входит в зону триггера, то свет становится слабее

Для этого нужно зайти в скрипт на объекте который входит в триггерную зону и сделать переменную-ссылку на свет, куда потом в Unity мы забросим наш источник света





Далее пишим следующую логику , чьл будет происходить когда объект будет в ходить в триггерную зону

