

<b>Школа:</b>		
<b>Дата:</b>	<b>ФИО учителя:</b>	
<b>Класс:</b>	<b>Участвовали:</b>	<b>Не участвовали:</b>
<b>Тема урока:</b> Способы программирования взаимодействия между объектами сцены		
<b>Цели обучения, которые достигаются на данном уроке</b>	Ознакомить учащихся с Unity	
<b>Цели урока</b>	<b>Все учащиеся:</b> <ul style="list-style-type: none"> <li>Знать способы взаимодействия между объектами сцены</li> </ul> <b>Большинство учащихся:</b> <ul style="list-style-type: none"> <li>Распознать способы взаимодействия между объектами сцены</li> </ul> <b>Некоторые учащиеся:</b> <ul style="list-style-type: none"> <li>Создавать и использовать способы взаимодействия между объектами сцены</li> </ul>	
<b>Воспитание ценностей</b>	Воспитание уважения к мнению друг друга, ответственность, коммуникативные способности, критическое мышление	
<b>Предварительные знания</b>	Настройка параметров физики для объектов. Создание и использование prefabs	
<b>Межпредметные связи</b>	Информатика, математика	
<b>Запланированные этапы урока</b>	<b>Запланированная деятельность на уроке</b>	<b>Ресурсы</b>
<b>Начало урока</b> <b>_5_ мин</b>	<b>Организационный момент.</b> Приветствие учащихся. Создание положительного эмоционального фона. Настрой на рабочий лад. Показываю тему урока и цель урока, озвучиваю критерии для цели урока <b>Деление на подгруппы.</b> Стратегия «Нумерация»	Карточки с цифры
<b>Середина урока</b> <b>_30_ мин</b>	Пусть у нас в проекте есть два скрипта. Первый скрипт отвечает за начисление очков в игре, а второй за пользовательский интерфейс, который, отображает количество набранных очков на экране игры. Назовем оба скрипта менеджерами: ScoresManager и HUDManager. Каким же образом менеджеру, отвечающему за меню экрана можно получить текущее количество очков от менеджера, отвечающего за начисление очков? Предполагается, что в иерархии объектов(Hierarchy) сцены существуют два объекта, на один из которых назначен скрипт ScoresManager, а на другой скрипт HUDManager. Один из подходов, содержит следующий принцип: В скрипте UIManager определяем переменную типа ScoresManager: <pre> public class HUDManager : MonoBehaviour { </pre>	

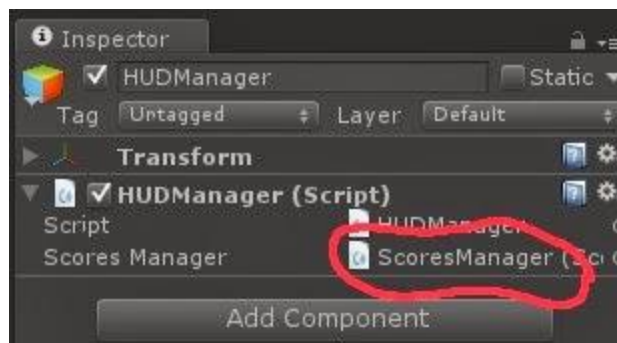
```
public ScoresManager ScoresManager;
```

```
}
```

Но переменную ScoresManager необходимо еще инициализировать экземпляром класса. Для этого выберем в иерархии объектов объект, на который назначен скрипт HUDManager и в настройках объекта увидим переменную ScoresManager со значением None.



Далее, из окна иерархии перетаскиваем объект, содержащий скрипт ScoresManager в область, где написано None и назначаем его объявленной переменной:




После чего, у нас появляется возможность из кода HUDManager обращаться к скрипту ScoresManager, таким образом:

```
public class HUDManager : MonoBehaviour
```

```
{
```

```
public ScoresManager ScoresManager;
```

```
public void Update ()
```

	<pre> {     ShowScores(ScoresManager.Scores); } } </pre> <p>Все просто, но игра, не ограничивается одними набранными очками, HUD может отображать текущие жизни игрока, меню доступных действия игрока, информацию о уровне и многое другое. Игра может насчитывать в себе десятки и сотни различных скриптов, которым нужно получать информацию друг от друга. Чтобы получить в одном скрипте данные из другого скрипта нам каждый раз придется описывать переменную в одном скрипте и назначать (перетаскивать вручную) ее с помощью редактора, что само по себе нудная работа, которую легко можно забыть сделать и потом долго искать какая из переменных не инициализирована. Если мы захотим что-то отредактировать, переименовать скрипт, то все старые инициализации в иерархии объектов, связанные с переименованным скриптом, сбросятся и придется их назначать снова. В то же время, такой механизм не работает для префабов (prefab) — динамического создания объектов из шаблона. Если какому-либо префабу нужно обращаться к менеджеру, расположенному в иерархии объектов, то вы не сможете назначить самому префабу элемент из иерархии, а придется сначала создать объект из префаба и после этого программно присвоить экземпляр менеджера переменной только что созданного объекта. Не нужная работа, не нужный код, дополнительная связанность.</p>	
<p><b>Конец урока</b> <b>_5_ мин</b></p>	<p><b>Рефлексия:</b> Метод «Рефлексивные карточки» Было интересно... Я понял, что... У меня получилось... Расскажу дома, что...</p>	<p>«Рефлексивные стикеры»</p> 
<p><b>Дифференциация – каким образом Вы планируете оказать больше поддержки? Какие задачи Вы планируете поставить перед более способными учащимися?</b></p>	<p><b>Оценивание – как Вы планируете проверить уровень усвоения материала учащимися?</b></p>	<p><b>Охрана здоровья и соблюдение техники безопасности</b></p>