Государственное бюджетное профессиональное образовательное учреждение
«Южно-Уральский государственный колледж»

РАССМОТРЕНО :

На заседании ПЦК «Информационных технологий»

От _____протокол №___

Председатель ПЦК

_____/Н.А. Назарова/

**МЕТОДИЧЕСКАЯ РАЗРАБОТКА**

Комбинированный урок

Специальность 09.02.07 Информационные системы и программирование

Дисциплина: МДК 12.02 Программирование и разработка на платформе Unity

Разработал преподаватель Е.В. Фостаковская

Челябинск, 2023

# Методическая разработка учебного занятия

**Учебная дисциплина:** МДК 12.02 Программирование и разработка на платформе Unity

**Специальность:** 09.02.07 Информационные системы и программирование

**Квалификация:** программист

**Курс:** 3

**Тема занятия:** Добавление объектов (клонирование) на сцену программно.

**Вид занятия:** комбинированый

**Продолжительность занятия:** 90 минут

С целью овладения профессиональными компетенциями обучающихся в ходе освоения дисциплины должен

уметь:

- разрабатывать программные модули для пользовательского интерфейса, игровых уровней и объектов;
- правильно составлять программный код в соответствии с общепринятыми парадигмами;
- исправлять, возникающие в процессе написания, и в процессе сборки, ошибки;
- выполнять поставленные в соответствие с ТЗ задачи;

знать:

- принципы разработки программного кода;
- методы работы с вспомогательным специализированным программным обеспечением, сопровождающем разработку игры;
- особенности всесторонней работы с Unity;
- основные принципы и методы написания компьютерных программ на языке программирования высокого уровня;
- основные принципы систематизации информации к решению практических задач по программированию.

Формирование ОК 1: Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности

Формирование ОК 2: Планировать и реализовывать собственное профессиональное и личностное развитие.

Формирование ОК 3: Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

Формирование ОК 10: Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

Формирование ПК 12.6: Выполнять работу в Unity, Работать с компонентами, сценами, пользовательским интерфейсом

Формирование ПК 12.7: Осуществлять работу с функциями Unity

Формирование ПК 12.8: Производить инспектирование компонентов программного обеспечения на предмет соответствия стандартам кодирования

Формирование ПК 12.9: Разрабатывать программные модули в соответствии с техническим заданием

**Цели занятия:**

**Обучающая:** с целью формирования у обучающихся умений владения актуальными методами работы в профессиональной сфере и использования современного программного обеспечения студент должен:

знать систему Prefab Unity;

уметь создавать префаб из объекта, редактировать префаб, программно создавать экземпляр префаба на сцене

**Развивающая**: развивать умение обучающихся самостоятельно работать с по инструкциям для получения навыков работы в программе Unity. Развивать трудолюбие, ответственность, внимательность, коммуникативность.

**Воспитательная**: воспитание профессиональных качеств личности обучающегося, таких как аккуратность и самостоятельность при выполнении заданий. Формирование интереса к профессиональной деятельности по специальности.

**Ход урока:**

Цель работы

Сегодня мы должны научиться

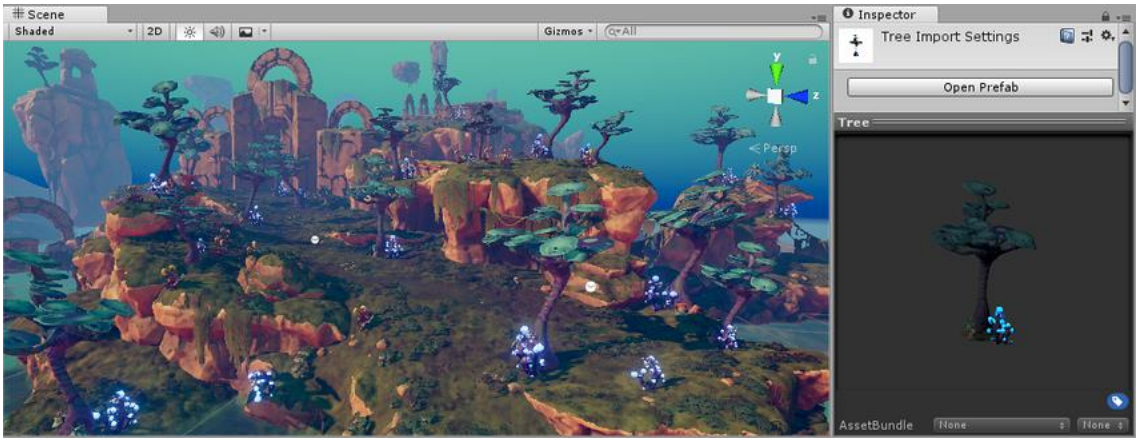
В ходе занятия мы должны познакомиться с новым материалом по теме: «Добавление объектов (клонирование) на сцену программно». И выполнить практическую работу по изучаемой теме.

***Система Prefab Unity***

Префаб (Prefab) в Unity - это шаблон объекта, который может быть использован для создания игровых объектов в сцене или в процессе игры. Префабы обычно используются для многократного использования ассетов, таких как 3D модели, материалы, анимации и скрипты.

Система Prefab Unity позволяет создавать, настраивать и хранить Игровой объект ( GameObject) в комплекте со всеми компонентами, значениями свойств и дочерними игровыми объектами как многоразовый объект. Prefab действует как шаблон, на основе которого вы можете создавать новые экземпляры Prefab на Сцене.

Если нужно повторно использовать игровой объект, настроенный определенным образом (например, неигровой персонаж (NPC), реквизит или часть декорации), в нескольких местах сцены или в нескольких сценах проекта, следует преобразовать его в Prefab. Это лучше, чем простое копирование и вставка GameObject, поскольку система Prefab позволяет автоматически синхронизировать все копии.



Любые изменения, которые вносятся в префаб, автоматически отражаются в экземплярах этого префаба, что позволяет легко вносить значительные изменения во весь проект без необходимости повторно вносить одно и то же редактирование в каждую копию актива.

Unity поддерживает иерархию префабов, что позволяет создавать сложные сцены с использованием иерархии родительских и дочерних объектов. Это обеспечивает большую гибкость и контроль над структурой сцены и поведением объектов в ней.

Можно вкладывать префабы в другие префабы, чтобы создавать сложные иерархии объектов, которые легко редактировать на нескольких уровнях.

Однако это не означает, что все экземпляры Prefab должны быть идентичными. Можно переопределить настройки отдельных экземпляров префаба, если хотите, чтобы некоторые экземпляры префаба отличались от других. Также можно создавать варианты префабов, которые позволяют группировать набор переопределений в значимый вариант префаба.

Можно использовать префабы, если хотите во время выполнения создавать экземпляры игровых объектов, которые не существовали в сцене в начале - например, чтобы бонусы, специальные эффекты, снаряды или неигровые персонажи появлялись в нужные моменты во время игры.

Некоторые распространенные примеры использования Prefab

- Экологические активы — например, определенный тип дерева, используемый несколько раз на уровне
- Неигровые персонажи (NPC) — например, робот определенного типа может появляться в вашей игре несколько раз на разных уровнях. Они могут отличаться (с использованием *переопределений* ) скоростью движения или звуком, который издают.
- Снаряды — например, пиратская пушка может создавать экземпляр префаба пушечного ядра каждый раз при выстреле.
- Главный герой игрока - префаб игрока может быть размещен в начальной точке каждого уровня (отдельных сцен) вашей игры.

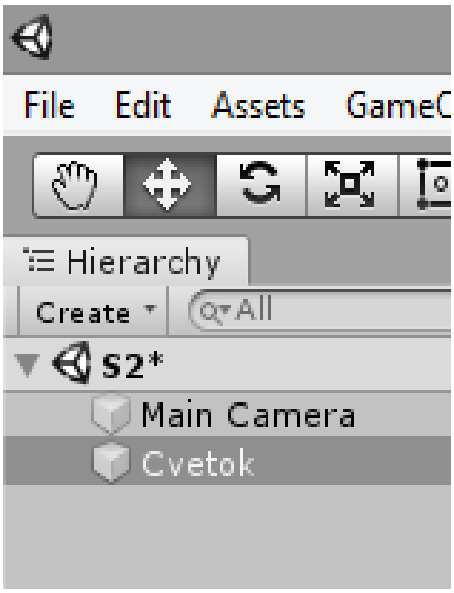
Создание префаба

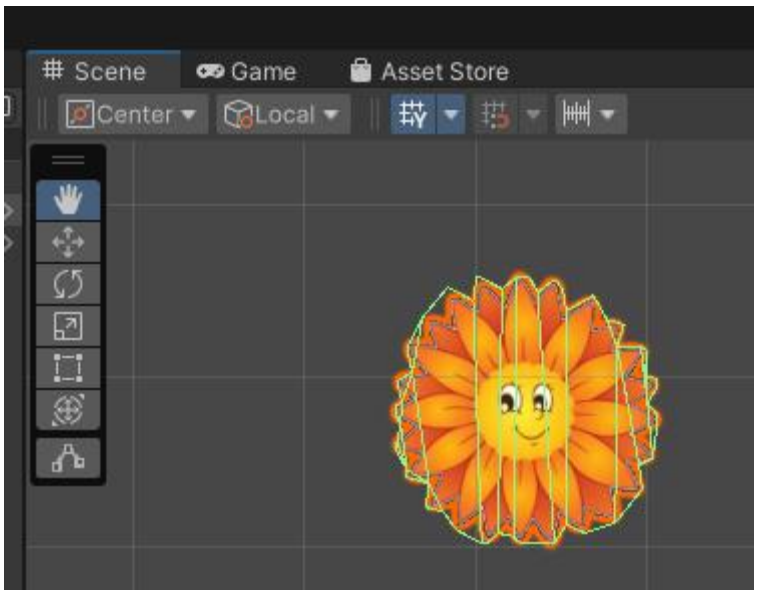
Создание префаба состоит из создания экземпляра объекта в сцене, настройки его свойств и сохранения его как префаба.

Пример:

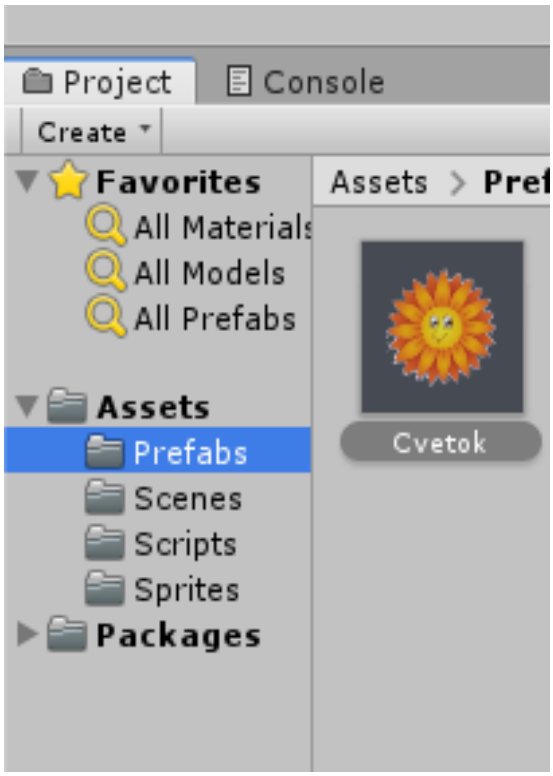
- Загрузите новый ассет изображение цветка (Assets – Import New Asset) в папку Sprites. Назовите его Cvetok.

- Перетащите его на сцену.






- Создайте папку Prefabs в папке Assets
- Теперь из цветка нужно сделать префаб. Для этого из окна иерархии перетащите объект Cvetok в папку Prefabs.




- Удалите со сцены цветок

***Создание экземпляра префаба программно***

Чтобы создать экземпляр префаба во время выполнения скрипта, коду нужна ссылка на этот префаб. Можно сделать эту ссылку, создав в своем коде общедоступную (public) переменную для хранения ссылки на Prefab. Открытая

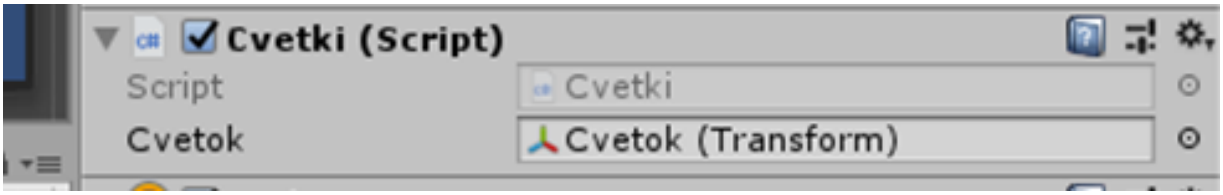
(public) переменная в коде отображается как назначаемое поле в Инспекторе. Затем нужно назначить в Инспекторе фактический префаб, который должен использоваться.

Пример: Создадим экземпляр префаба цветка на сцене программно. Для этого создадим скрипт, назовем его Cvetki. Откроем скрипт и в нем объявим переменную Cvetok

```
using UnityEngine;
public class InstantiationExample : MonoBehaviour
{
public GameObject Cvetok;
```

Сохраним скрипт и вернемся опять в Unity

В окно переменной Cvetok, там где отражается добавленный скрипт, перетащим пиктограмму префаба Cvetok из папки префабов.




Для создания экземпляра (клона) префаба в языке C# используется метод

Instantiate ( исходный объект , позиция Vector3 , вращение кватерниона );

аргументами которого являются название префаба, его координаты на сцене и угол вращения на сцене.

В приведенном примере сценария есть одна общедоступная переменная «Cvetok» , которая является ссылкой на Prefab. Он создает экземпляр этого префаба в методе **Start()** .

```
using UnityEngine;
public class InstantiationExample : MonoBehaviour
{
public GameObject Cvetok;
void Start() {
Instantiate (Cvetok, new Vector3(0, 0, 0), Quaternion.identity);
```

}

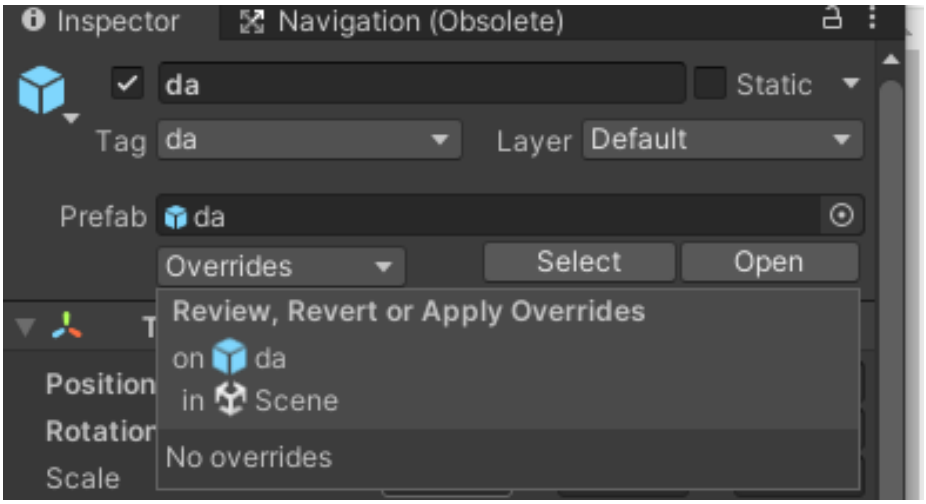
}

***Редактирование префаба из его экземпляров***

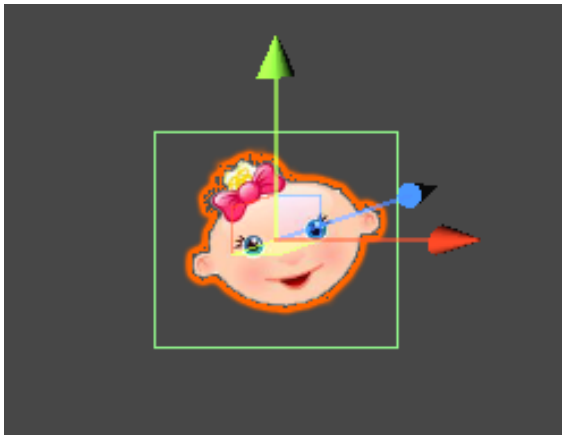
Инспектор экземпляра префаба содержит три кнопки, которые у обычных объектов отсутствуют: *Select* (Выделить), *Open* (Открыть) и Overrides (Переопределить).

Кнопка *Select* выделяет файл префаба, из которого был получен данный экземпляр. Это позволяет быстро найти префаб.

Кнопка *Open* позвоялет открыть и отредактировать оригинальный префаб, применяя изменения ко всем его экземплярам.




Также вы можете сохранить переопределённые свойства из экземпляра в сам оригинальный префаб с помощью кнопки Overrides . Это позволяет эффективно редактировать все экземпляры через любой из них, и это быстрый и правильный способ вносить глобальные изменения.



Теперь перейдем к практической работе.

**Практическая работа**

**Цель занятия**: создание объекта Кнопка, создание префаба, загрузка префаба на сцену.

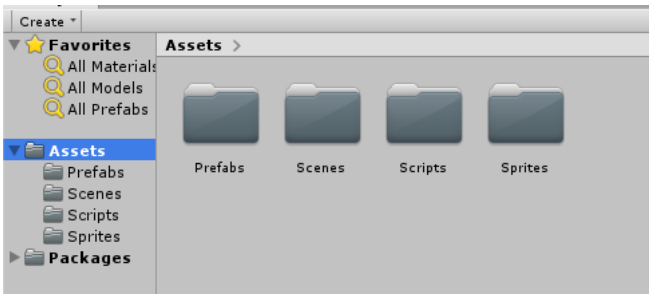
**Оборудование**: ПК

**Программное обеспечение:** Unity, Visual Studio.

**Задание:** Сделать программу, в которой будет на сцене две кнопки: Выход и Цветочки. При щелчке по кнопке Цветочки на сцену должны загружаться 10 экземпляров цветочка (из префаба) со случайными координатами.

**Выполнение**:

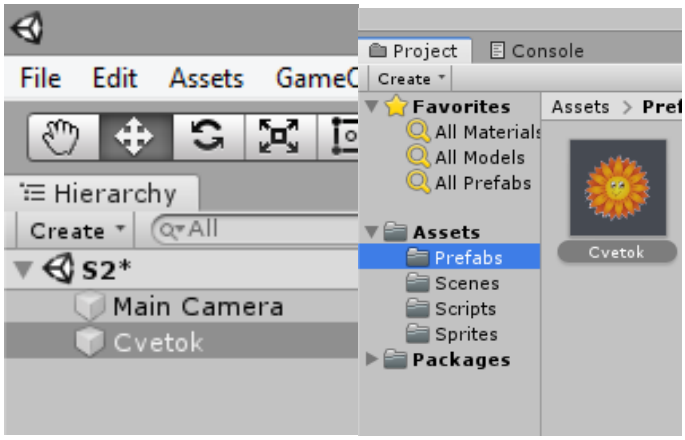
1. Откройте программу и создайте новый проект, который назовите своей фамилией (транслитом).
2. Во вкладке проекта Project в папке Assets 4 папки: Prefabs, Scenes, Scripts, Sprites, выполнив Assets – Create - Folder



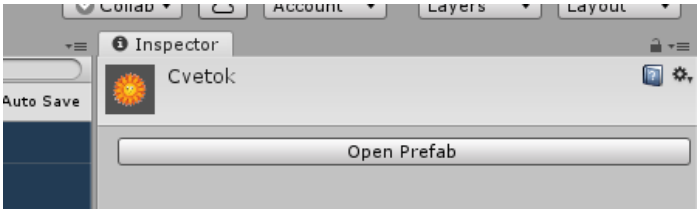

3. Создайте новую сцену в папке Scenes, выполнив Assets – Create – Scene
4. Двойным щелчком по пиктограмме, откройте созданную сцену


5. Загрузите новый ассет изображение цветка (Assets – Import New Asset) в папку Sprites. Назовите его словом Cvetok.

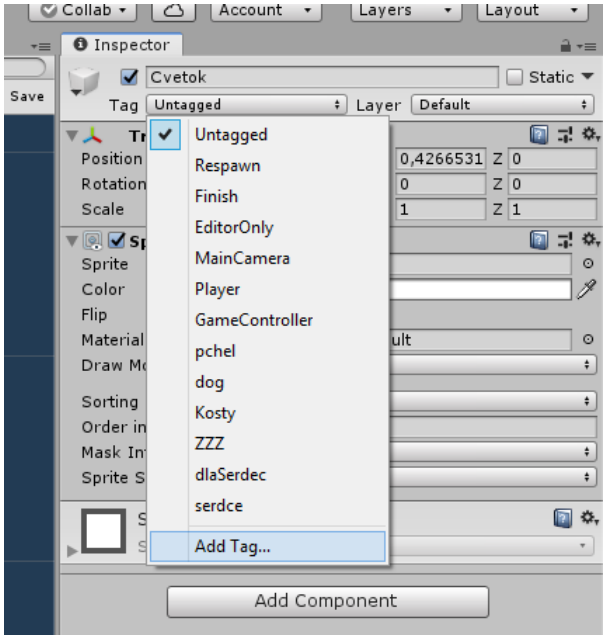
6. Перетащите его на сцену.
7. Теперь из этого цветка нужно сделать префаб. Для этого из окна иерархии перетащите объект Cvetok в папку Prefabs.

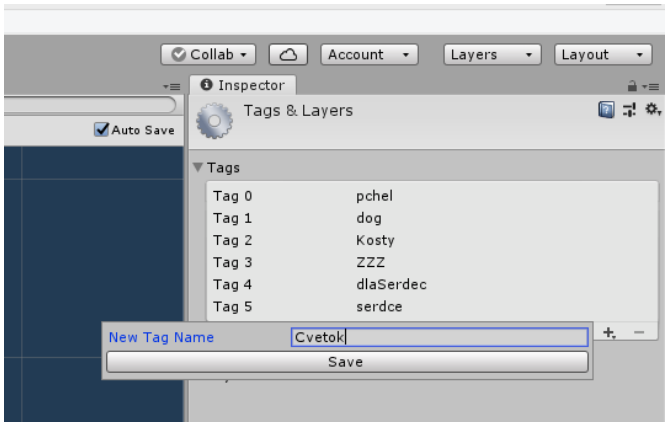



8. Удалите цветок со сцены.
9. В папке префабов щелкните по пиктограммке цветка и в окне инспектора откройте префаб.

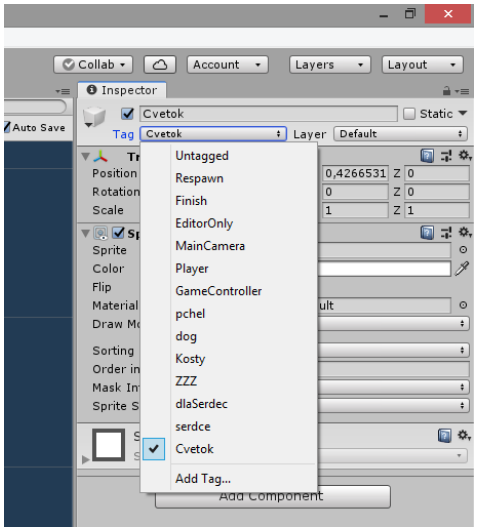



10. Добавьте в Tag название Cvetok. Add Tag, введите название и щелкните Save.





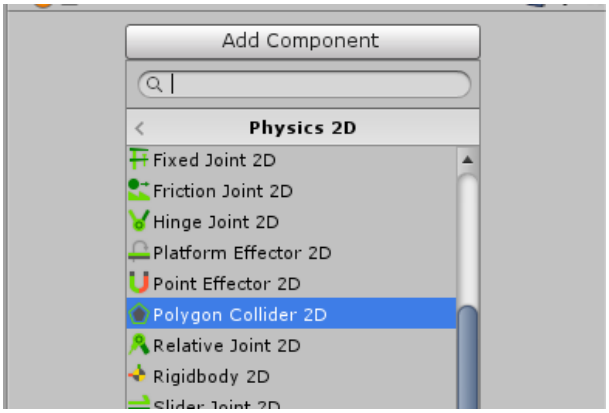

11. После сохранения опять откройте инспектор цветка и выберите из списка его название.



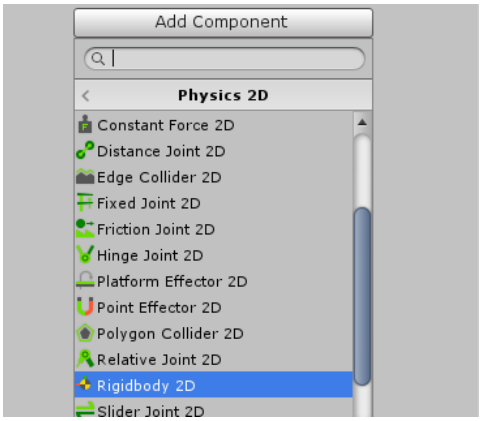

Для правильного создания префаба посмотрите видеофайл: Создание префаба.mov

12. Добавьте к свойствам еще два свойства, которые потребуются для взаимодействия объектов:

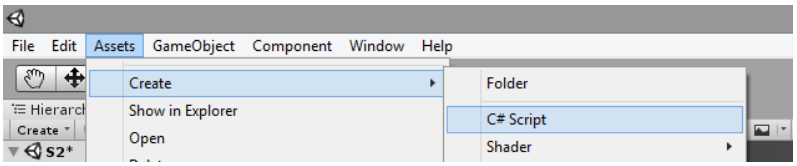
12.1. Poligon Collider 2D, нажав на кнопку Add Component и выбрав Physics 2D, затем Poligon Collider 2D




12.2. Rigibody 2D, нажав на кнопку Add Component и выбрав Physics 2D, затем Rigibody 2D




13. В свойстве Gravity Scale (Гравитация) поставьте значение 0, чтобы цветы не падали вниз на сцене.
14. Перейдите на сцену. На ней пока находится только камера.
15. Теперь в папке Scripts создайте новый скрипт и назовите его Cvetki.

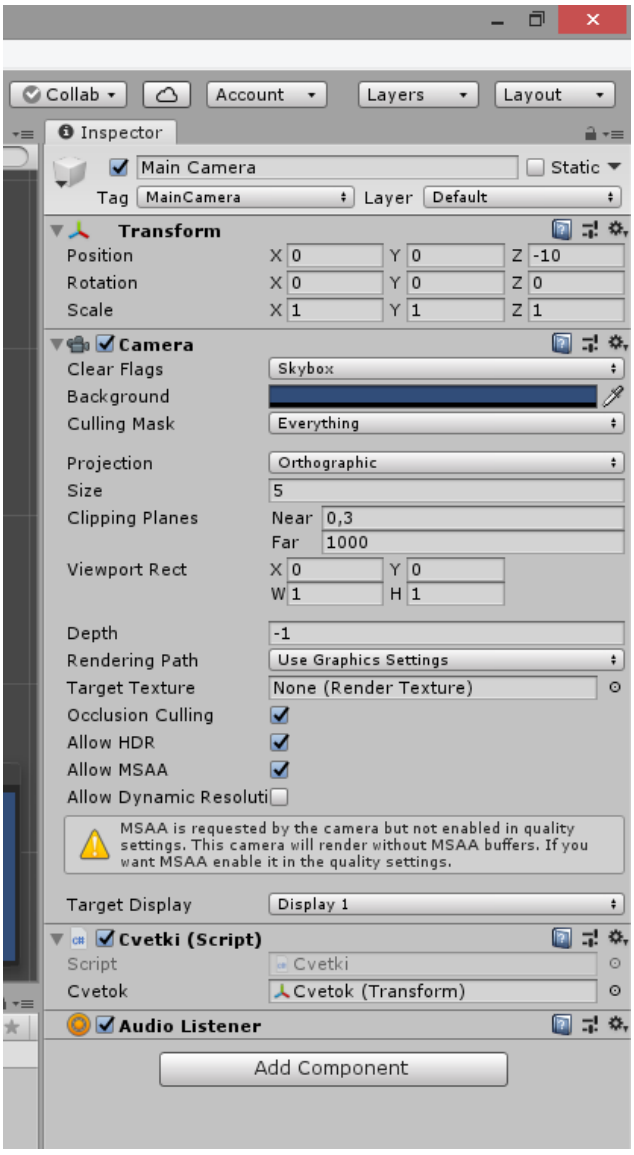



16. Создадим скрипт, при котором при нажатии на кнопку будут добавляться 10 цветков со случайными координатами.
17. Откройте скрипт, щелкнув на нем дважды, запустится Visual Studio. Объявите переменную Cvetok, сохраните изменения, перейдите в Юнити.

```
// Создание переменной цветок
   public Transform Cvetok;
```

18. Прикрепите скрипт к камере, перетащив его в окно инспектора.

19. А теперь в окно переменной Cvetok перетащите пиктограмку префаба цветок. Это делается для того, чтобы пребаф можно было добавить на сцену методом Instantiate.




Вы можете посмотреть видео Присвоение префаба переменной.mov, чтобы научиться переменной скрипта присваивать значение префаба.

20. Перейдите в скрипт. Создайте две кнопки: выхода из программы и добавления цветков на случайные координаты. Кнопки будут создаваться программно, с помощью класса GUI. Этот класс отвечает за элементы управления, такие как кнопки, переключатели, текстовые поля и пр. программно создать элемент управления можно только в обработчике событий OnGUI. Внутри кнопки, которая добавляет цветы на сцену, используем цикл for со счетчиком i. В структуре Vector3 пишутся кординаты X, Y, Z. Для координаты Z укажем 10,

чтобы цветы располагались дальше всех от камеры, т.е. были на заднем плане. Для генерации случайных координат X, Y используем метод Range класса Random. Весь код скрипта выглядит следующим образом:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Cvetki : MonoBehaviour
{
    // Создание переменной цветок
    public Transform Cvetok;

    void OnGUI()
    {
        //Глобальный цвет, подсвечивающий фон элементов GUI.
        GUI.backgroundColor = Color.yellow;
        //Цвет глобальной подсветки для GUI. Влияет на цвет фона и текста.
        GUI.color = Color.blue;
        // кнопка выхода
        if (GUI.Button(new Rect(10, 10, 100, 30), "Выход"))
        {
            Application.Quit();
        }
        // кнопка для случайного появления цветочков
        if (GUI.Button(new Rect(1000, 10, 100, 30), "Цветочки"))
        {
            for (int i = 0; i < 10; i++)
            {
                // добавление цветов на сцену
                Instantiate(Cvetok, new Vector3(Random.Range(-10.0f, 10.0f), Random.Range(-
5.0f, 5.0f), 10), Quaternion.identity);
            }
        }
    }

}
```

21. Сохраните скрипт и проверьте программу.

22. Сохраните проект.

**Контрольные вопросы:**

- Что такое Prefab?
- Расскажите алгоритм создания префаба из объекта?
- Как создать сценарий в программе Unity?
- Какой метод применяется для клонирования объектов из префаба на сцене?
- Как объявить общедоступную переменную в скрипте?
- Как прикрепить фактический префаб к объявленной в скрипте общедоступной переменной в среде разработке Unity?

**Методическое обеспечение занятия:**

Электронные издания (электронные ресурсы)

1. Корнилов, А. В. UNITY. Полное руководство. (+виртуальный DVD 10 Гб с Unity-проектами, примерами из книги и ассетами) / А. В. Корнилов. — 2-е изд. — Санкт-Петербург : Наука и Техника, 2021. — 496 с. — ISBN 978-5-94387-721-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/191482 (дата обращения: 28.09.2023). — Режим доступа: для авториз. пользователей.
2. Курбанисмаилов, З. М. Современные подходы в программировании при создании интерактивной анимации на C# и Unity : учебно-методическое пособие / З. М. Курбанисмаилов. — Москва : РТУ МИРЭА, 2021. — 142 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/176569 (дата обращения: 28.09.2023). — Режим доступа: для авториз. пользователей.
3. Ларкович, С. Н. UNITY на практике. Создаем 3D-игры и 3D-миры : учебное пособие / С. Н. Ларкович. — 2-е изд. перераб. и доп. — Санкт-Петербург : Наука и Техника, 2022. — 384 с. — ISBN 978-5-907592-02-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/297155 (дата обращения: 28.09.2023). — Режим доступа: для авториз. пользователей.
4. Руководство пользователя Unity 2022: официальная документация кроссплатформенной среды разработки компьютерных игр - docs.unity3d.com
5. Unity в действии. Мультиплатформенная разработка на C# / Джозеф Хокинг/ 2-е международное издание — СПб.: Питер, 2019. — 352 с.