



For beginners

LESSON 1

MODULE 1

Unity – универсальный движок

Unity – комплексный кроссплатформенный движок, предоставляющий огромный набор инструментов, которые упрощают разработку игр.

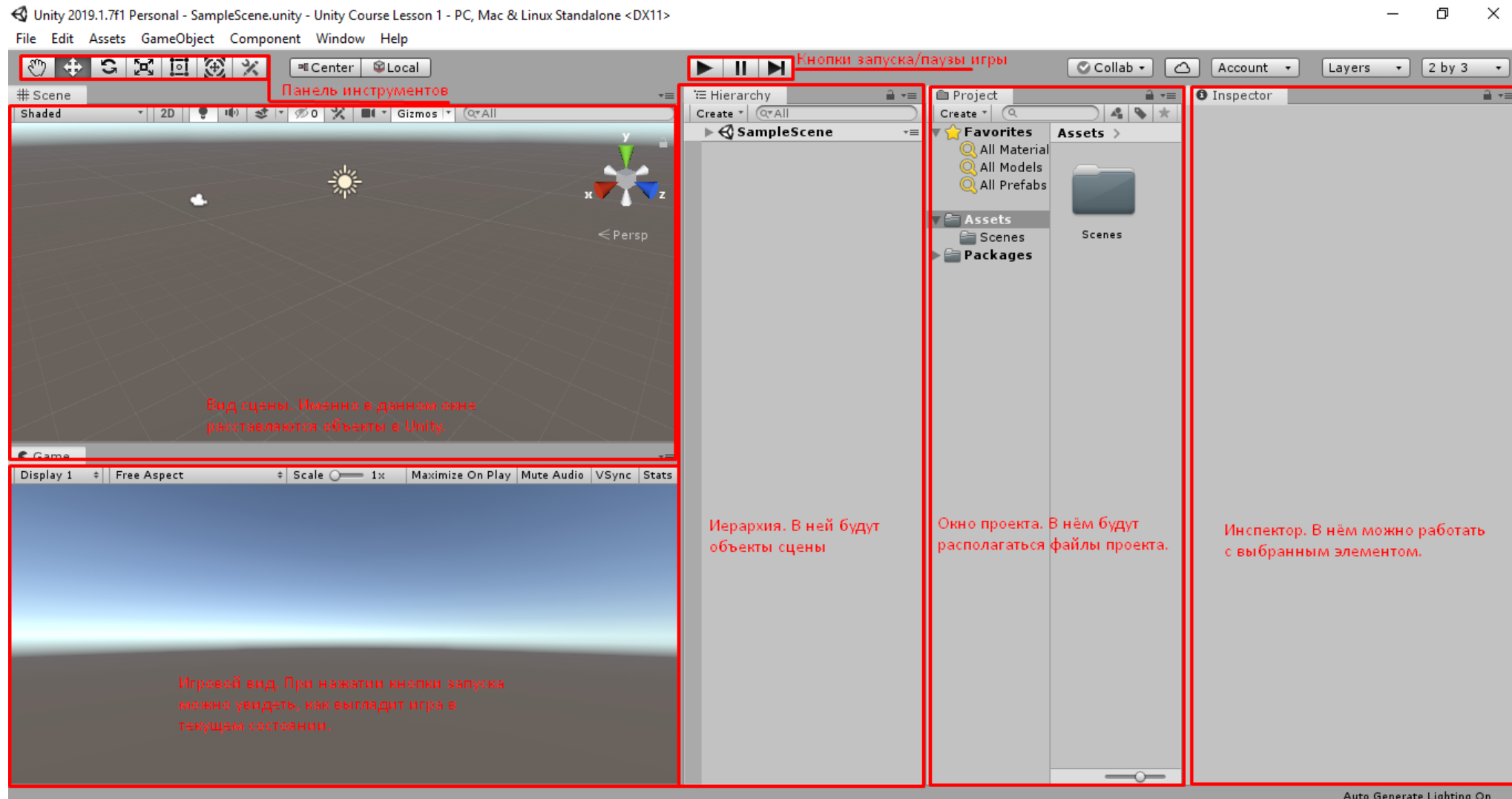
Unity предоставляет огромный набор инструментов, упрощающий разработку игры.

Язык, на котором мы будем писать игры – **C#**

Найти бесплатную персональную версию движка можно по [данной ссылке](#).

Обращаю внимание, что мы будем работать с расположением окон “2 by 3”.

Пользовательский интерфейс Unity

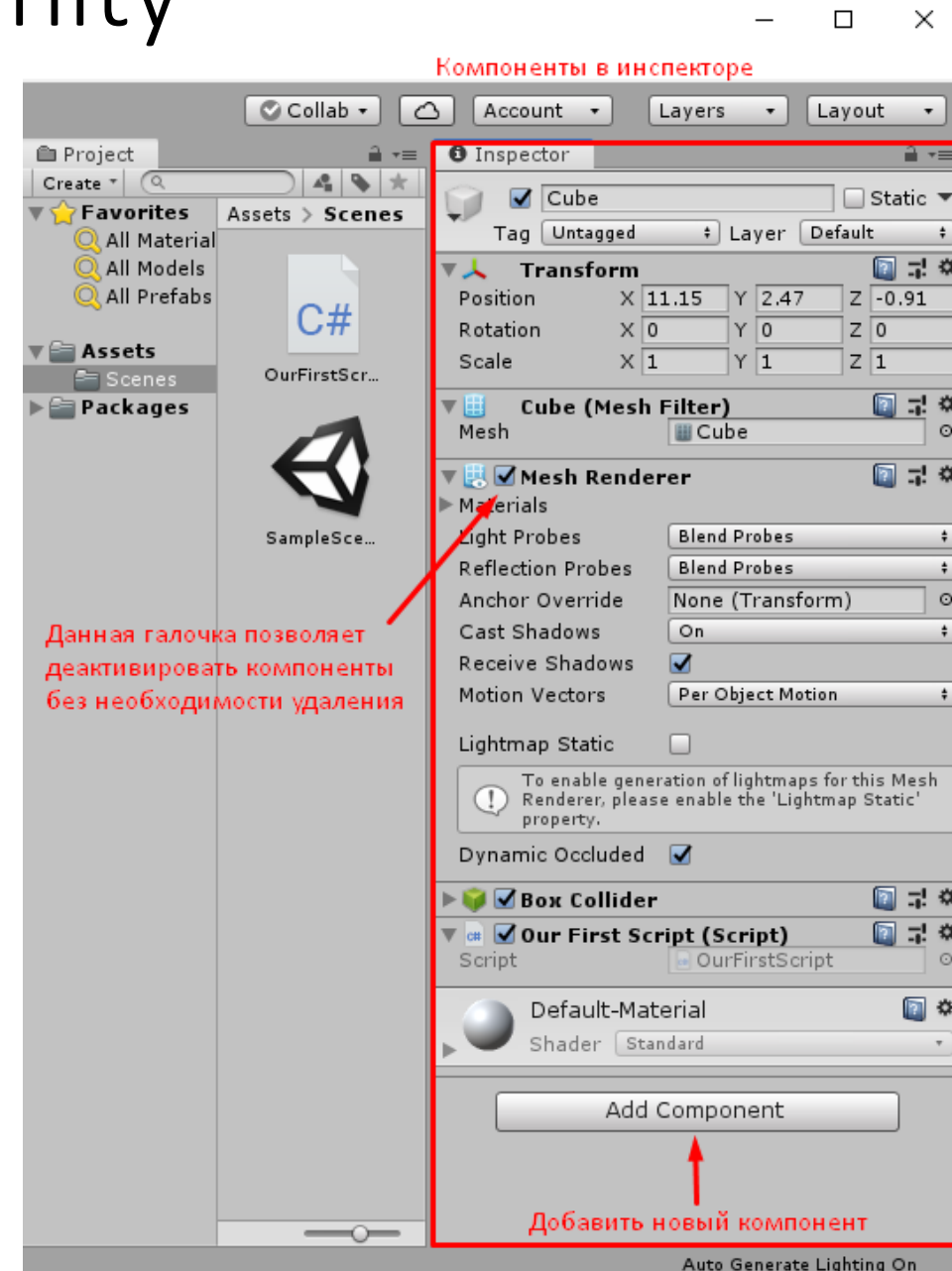


Компоненты в Unity

Компоненты в Unity – некоторые «наборы» функционала, которые можно прикреплять к игровым объектам в пространстве Unity. Данный движок предоставляет огромный набор компонентов, которые упрощают нашу разработку (например, Mesh Renderer отвечает за отрисовку объектов в пространстве, Transform определяет позицию объекта в пространстве и многое другое).

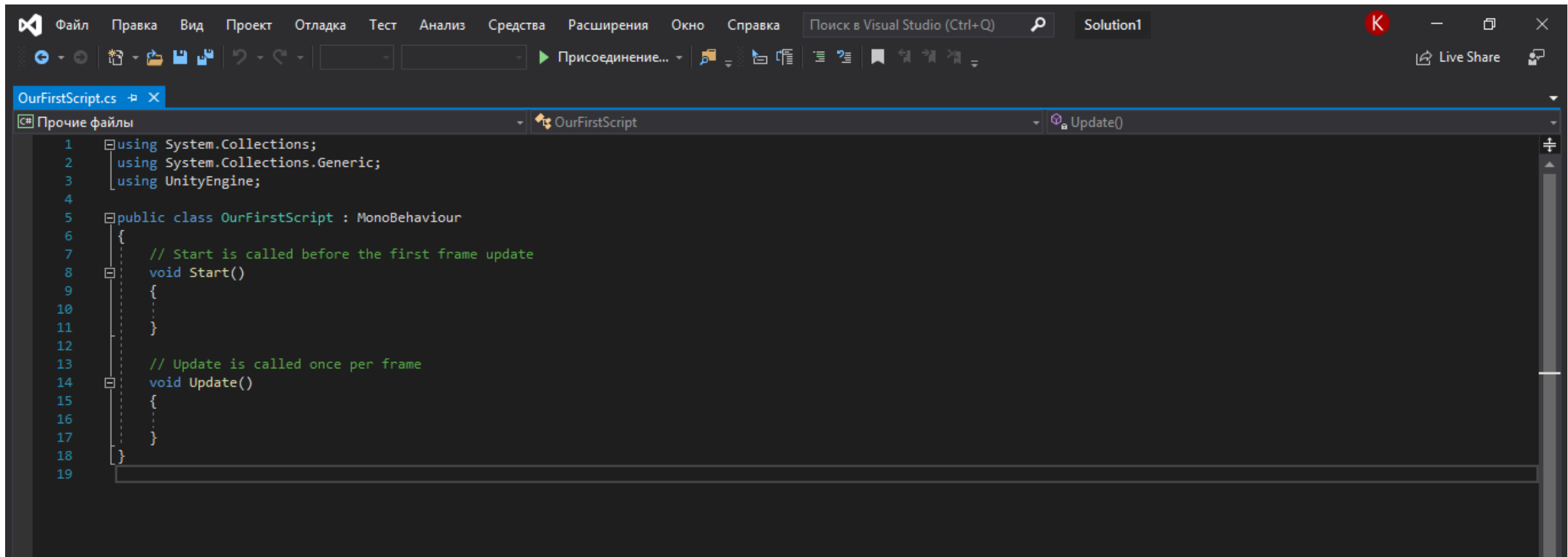
При помощи кнопки **Add Component** можно добавить любой необходимый компонент, причём компоненты, добавленные во время запуска игры исчезнут при остановке. Этим можно пользоваться, когда, например, необходимо протестировать физику.

Используя галочку можно также деактивировать компоненты. Например, так можно включать/отключать агрессивный режим монстров.



Скрипты в Unity - Компоненты

Всегда помните, что скрипты в Unity – это пользовательские компоненты. Их можно прикреплять к объектам подобно стандартным, являющимся частью Unity. При создании Unity-скрипта шаблон по умолчанию будет выглядеть так:

A screenshot of the Visual Studio IDE interface. The top menu bar includes options like 'Файл', 'Правка', 'Вид', 'Проект', 'Отладка', 'Тест', 'Анализ', 'Средства', 'Расширения', 'Окно', 'Справка', and a search bar 'Поиск в Visual Studio (Ctrl+Q)'. The toolbar below the menu contains icons for file operations and development tools. The main editor window displays a C# script named 'OurFirstScript.cs'. The script content is as follows:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class OurFirstScript : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
19
```

The script is shown with line numbers on the left. The 'OurFirstScript' namespace is visible in the Solution Explorer on the right side of the editor.

Структура Скрипта

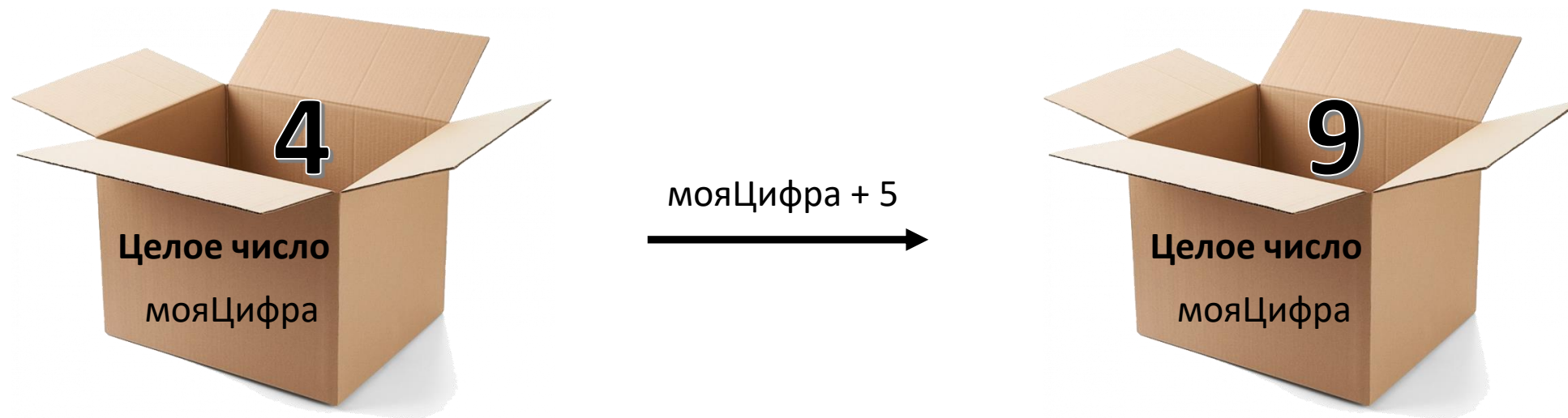
Скрипты, с которыми мы будем работать, представляют набор функций, которые можно активировать в определённый момент. Помните, что в идеале каждый Unity-скрипт будет отвечать за одну функцию. Например, скрипт перемещения персонажа состоит из параметров перемещения (скорость, направление и т. д.) и функций, описывающих перемещение (сместиться на столько-то единиц в указанном направлении, повернуться на столько-то единиц и т. д.).

Код наших скриптов будет располагаться после открывающихся фигурных скобок с обозначением класса (при создании файла Unity добавит их автоматически).

Внутри класса можно увидеть две функции по умолчанию – **Start()** и **Update()**. Первая выполняется в момент появления объекта, к которому прикреплён скрипт, на сцене; вторая – при каждом обновлении кадра (изображения на экране).

Переменные в C#

Переменные стоит воспринимать как некоторый контейнер для информации. В данном контейнере указывается возможность доступа, тип объекта и его уникальное название, а также в него помещается некоторое значение, с которым в дальнейшем можно совершать некоторые действия:



Типы Данных. Создание Переменных

Для начала, предлагаю вам ознакомиться с набором простых типов данных C#, которые мы будем использовать каждый день в наших скриптах. Не забываем при создании переменных указывать модификатор доступа.

public – переменную можно менять в инспекторе Unity, а также к ней можно обращаться из других скриптов.

int – целочисленное число.

float – десятичная дробь с точкой, на конце пишется суффикс **f**. На заметку: Unity использует именно этот тип для значений координат и большинства дробных значений.

bool – логический тип, имеет значения либо правда, либо ложь (**true/false**)

string – строка из символов, может содержать в себе пробелы

Vector3 – тип Unity, набор связанных друг с другом 3 координат типа float – x, y и z

Подробнее о Vector3 в документации [по данной ссылке](#).

Создаём Переменные

```
OurFirstScript.cs*  X
C# Прочие файлы OurFirstScript

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class OurFirstScript : MonoBehaviour
6  {
7      // После двух символов // можно (и нужно) писать комментарии в коде
8
9      /* А так выглядят комментарии на несколько строк
10      *
11      * Их можно использовать такжн для того, чтобы деактивировать строки кода
12      * так как код в комментариях не выполняется*/
13
14      // Несколько примеров создания переменных
15      // Формат объявления: <модификатор доступа> <тип> <имя> = <значение>;
16      // Или так: <модификатор доступа> <тип> <имя>; (это объявление без инициализации значения);
17
18      public int myInt = 4;
19      public string myString = "Hello I am an example!";
20      private float myFloat;
21      private bool myBool = true;
22
23      // Инициализация значения уже при активации объекта на сцене
24      void Start()
25      {
26          myFloat = 3.75f;
27      }
28  }
```

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

public class OurFirstScript : MonoBehaviour
{
    // Работаем с условиями и Vector3

    // Vector3 - объект Unity, используем ключевое слово new для
    // его создания и далее в круглых скобках передаём 3 координаты.

    public Vector3 myVector1 = new Vector3(5f, 6.25f, 3f);

    public float xCoord = 4.5f;
    public float yCoord = 1f;
    public float zCoord = 3f;

    private Vector3 myVector2;

    // Инициализация значения уже при активации объекта на сцене
    // Обратите внимание, что для значений Vector3
    // можно использовать другие переменные.
    // Хотя сам вектор приватный, мы можем менять координаты в инспекторе.
    void Start()
    {
        myVector2 = new Vector3(xCoord, yCoord, zCoord);
        transform.position = myVector2;
    }
}
```

Области Видимости

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 0 references
6 public class OurFirstScript : MonoBehaviour
7 {
8     // Работаем с областями видимости
9
10    // Данная переменная может быть использована в любой части
11    // скрипта, т. к. она принадлежит всему классу. К ней можно
12    // обращаться внутри любого метода.
13
14    public int powerfulInt = 5;
15
16    // Данная приватная переменная существует только
17    // внутри метода Start(). При попытке обратиться
18    0 references
19    void Start()
20    {
21        int limitedInt = 4;
22
23        // Парочка математических операторов
24        // Вычитаем/прибавляем 1; прибавляем 4 к себе.
25        limitedInt--;
26        limitedInt++;
27        limitedInt += 4;
28        ...
29    }
30
31    // Очень важный момент! Переменные методов
32    // могут иметь одинаковые названия, однако
33    // всегда необходимо помнить, что это абсолютно разные вещи!
34    0 references
35    void Example()
36    {
37        int limitedInt = 2;
38        limitedInt *= 4;
39    }
40 }
```

Крайне важно помнить о таком понятии, как область видимости в C#. **Область видимости** – часть кода, в пределах которой доступна конкретная переменная. Области видимости определяются фигурными скобками – вы можете объявлять свои переменные уже внутри класса (пример – `powerfulInt` на скриншоте). Такая переменная доступна в любой части скрипта. Помните, что у различных переменных в одной области видимости никогда не бывает одинакового имени.

В данном примере можно также увидеть 2 переменные с названием `limitedInt`, каждая из которых существует только внутри собственного метода. При попытке обратиться к `limitedInt` вне `Start()` или `Example()`, возникнет ошибка, так как в рамках класса такой переменной не существует.

Всегда помните, что `limitedInt` в `Start()` и `limitedInt` в `Example()` – 2 никак не связанные переменные. Для лучшего понимания рекомендую воспринимать фигурные скобки как начало/конец области видимости.

Также рекомендую в целом избегать повторяющихся названий переменных, это может путать.

Арифметика

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class OurFirstScript : MonoBehaviour
6  {
7      // Работаем с арифметическими действиями
8
9      // В C# с числами можно производить все арифметические действия
10     // Их порядок выполнения без скобок совпадает с математическим
11     // Кстати, переменные одного типа можно записать и в строку)
12
13     public int a = 12, b = 4, c = 10, d;
14
15     void Start()
16     {
17         d = (a + b - c) * b;
18         print(d);
19         // Выведет 24
20
21         d = (a + b) / c;
22         print(d);
23         // Выведет 1, т. к. возвращает целую часть от деления.
24
25         d = (a + b) % c;
26         print(d);
27         // Оператор % находит остаток от деления, в данном случае - 6
28     }
29
30 }
```

C# поддерживает набор стандартных операторов для выполнения различных арифметических действий.

Также можно складывать и строки – результатом суммы строк будет являться строка, содержащая обе предыдущие.

Unity Transform

Transform – крайне важный компонент в Unity. Он отвечает за позицию, угол поворота и габариты объектов сцены Unity. Запомните – каждый объект сцены содержит Transform. Мы только начинаем углубляться в Unity API, поэтому сейчас мы разберём лишь небольшую часть функционала Transform.

Запомните: вы можете обращаться к компоненту Transform любого игрового объекта при помощи ключевого слова **transform** (с маленькой буквы) и оператора «.».

Значения:

transform.position – позволяет напрямую задать координаты в пространстве. Принимает значения Vector3.

transform.eulerAngles – позволяет задать углы поворота по осям x, y и z. Принимает значения Vector3.

transform.localScale – позволяет менять габариты объекта по осям x, y и z. Принимает значения Vector3.

Unity Transform

Т. к. данные параметры принимают значения `Vector3`, значит с ними можно проводить операции сложения, вычитания, деления и умножения. Например:

```
transform.position += Vector3.forward * Time.deltaTime;
```

Методы:

transform.Translate(Vector3 направление); – перемещает объект по данному вектору.

transform.Rotate(Vector3 углыПоворота); – поворачивает объект по выбранным осям.

Примеры:

```
transform.Translate(new Vector3(2f, 3.5f, 1.25f));
```

```
transform.Rotate(new Vector3(60f, 0f, 0f));
```

Домашнее задание – обязательное!!!

- 1) Создайте скрипт, который при запуске игры будет задавать стартовые координаты куба в плоскости при запуске игры.
- 2) Доработайте предыдущий скрипт так, чтобы можно было менять стартовые координаты x, y и z в инспекторе.
- 3) Используя **Time.deltaTime** и **Vector3**, создайте скрипт, который позволит перемещать объект вперёд (**Vector3.forward**) со скоростью n единиц в секунду. Можете использовать как **transform.Translate()**, так и **transform.position**, в данном задании это не принципиально.
- 4) Доработайте предыдущий скрипт так, чтобы можно было свободно менять значения направления перемещение объекта в инспекторе.
- 5*) Доработайте скрипт так, чтобы при превышении определённых координат по x, y или z объект телепортировался на его стартовую позицию (в которой он появляется в момент запуска игры). Для этого используйте условие **if**
- 6**) Попробуйте создать переменную-таймер так, чтобы куб двигался вперёд n секунд, затем останавливался, затем снова двигался. *Подсказка: для этого следует использовать Time.deltaTime.*