C# basic

C# 简介

Unity中的Script基于C#

C# 是一个现代的、通用的、面向对象的编程语言

C# 的构想十分接近于传统高级语言 C 和 C++，是一门面向对象的编程语言，但是它与 Java 非常相似

C# 基本语法

C# 程序结构

一个 C# 程序主要包括以下部分:

- 命名空间声明 (Namespace declaration)
- 一个类(class)
- 一个Main方法
- Class方法与属性

和Java很像!

```
using System;
namespace HelloWorldApplication
{
   class HelloWorld
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Hello World");
         Console.ReadKey();
      }
   }
}
```

你可能注意到了 Unity的脚本中没有Main()方法

```
using UnityEngine;
using UnityEngine.UI;

public class LivesUI : MonoBehaviour {

        public Text livesText;

        // Update is called once per frame
        void Update () {
                livesText.text = PlayerStats.Lives.ToString() + " LIVES";
        }
}
```

这是因为绑定在游戏对象上的脚本都会继承（稍后谈到） `MonoBehaviour` 类，这个类中使用 `Start()` 和 `Update()` 等方法通过协程（在后续的课程中谈到）管理它的生命周期。可以把Unity引擎的启动看做是 `Main()` 方法，而游戏对象的生命周期则是由Unity引擎管理的。

C# 基本语法

和C++、Java很像!

变量

```
int i, j, k;
char ch = 'c';
float f, salary;
double d;
i = 10;
j = 20;
```

常量

```
const double pi = 3.14159;
const string s = "runoob";
```

判断

`if` `else` 三目运算符等方法同C++

循环

`for` `while` `do...while` 等方法同C++

`foreach` 方法：

```
int [] numbers = {1, 2, 3, 4, 5, 6};
foreach (var i in numbers) {
    Console.WriteLine(i);
}
```

方法/函数

和Java一样 你需要在Class中定义方法

```
using System;

namespace CalculatorApplication
{
   class NumberManipulator
   {
      public int FindMax(int num1, int num2)
      {
         return num1 > num2 ? num1 : num2;
      }
      static void Main(string[] args)
      {
         int a = 100, b = 200;
         NumberManipulator n = new NumberManipulator();

         //调用 FindMax 方法
         int ret = n.FindMax(a, b);
         Console.WriteLine("最大值是： {0}", ret );
         Console.ReadLine();
      }
   }
}
```

空类型

？用于对 int、double、bool 等无法直接赋值为 null 的数据类型进行 null 的赋值，意思是这个数据类型是 Nullable 类型的。

```
int i; //默认值0
int? ii; //默认值null
int? iii = 3; //iii可以为null
```

枚举

枚举列表中的每个符号代表一个整数值，一个比它前面的符号大的整数值。默认情况下，第一个枚举符号的值是 0.例如：

```
enum Day { Sun, Mon, tue, Wed, thu, Fri, Sat };
```

字符串

```
string str = "Hello World";
```

数组

```
int [] n = new int[10]; /* n 是一个带有 10 个整数的数组 */
```

C# OOP

C# 面向对象

类的定义、成员函数、构造/析构函数和C++ OOP类似。

特别的 Unity的脚本中不需要也不能为其定义构造函数，这是因为继承了 `MonoBehaviour` 类的脚本的生命周期由Unity引擎管理。

继承与派生

C# 只支持单一继承。 也就是说，类只能继承自一个类。

派生类继承了基类的成员变量和成员方法。使用 `public` `protected` `private` 等关键字来控制成员的访问权限。

访问级别

- public： 无限制;
- internal： 只允许在同一程序集内访问;
- protected： 只允许当前类和派生类访问;
- protected internal： 只允许同一程序集内的当前类和派生类访问;
- private： 只能在当前类中访问;

接口Interface

和Java中的Interface类似。接口中只包含成员的声明，没有包含成员的实现。

```
using System;

interface IMyInterface
{
        // 接口成员
    void MethodToImplement();
}

class InterfaceImplementer : IMyInterface
{
    static void Main()
    {
        InterfaceImplementer iImp = new InterfaceImplementer();
        iImp.MethodToImplement();
    }

    public void MethodToImplement()
    {
        Console.WriteLine("MethodToImplement() called.");
    }
}
```

Object 类

Object 类是类层次结构的根。每个类都派生自Object。所有对象（包括数组）都实现这个类的方法。通常不需要类来声明继承 Object， 因为继承是隐式的。

Object的一部分方法

方法名	
Equals	确定指定的对象是否等于当前对象。
Finalize	在垃圾回收将某一对象回收前允许该对象尝试释放资源并执行其他清理操作。
GetHashCode	用作特定类型的哈希函数。
GetType	获取当前实例的 Type。
ToString	返回表示当前对象的字符串。