# C#语言

## 重载和重写的区别

1. 所处位置不同 重载在同类中 重写在父子类中
2. 定义方式不同 重载方法名相同 参数列表不同 重写方法名和参数列表都相同
3. 调用方式不同 重载使用相同对象以不同参数调用 重写用不同对象以相同参数调用
4. 多态时机不同 重载时编译时多态 重写是运行时多态

## 面向对象的三大特点

封装、继承、多态

1、封装： 封装是将数据和行为相结合，通过行为约束代码修改数据的程度，增强数据的安全性，属性是C#封装实现的最好体现。

2、继承： 提高代码重用度，增强软件可维护性的重要手段，符合开闭原则。

3、多态性： 多态性是指同名的方法在不同环境下，自适应的反应出不同得表现，是方法动态展示的重要手段。

## 简述值类型和引用类型有什么区别

1.值类型存储在内存栈中，引用类型数据存储在内存堆中，而内存单元中存放的

是堆中存放的地址。

2.值类型存取快，引用类型存取慢。

3.值类型表示实际数据，引用类型表示指向存储在内存堆中的数据的指针和引用。

4.栈的内存是自动释放的，堆内存是.NET 中会由 GC 来自动释放。

5.值类型继承自 System.ValueType,引用类型继承自 System.Object。

## 请简述private，public，protected，internal的区别

public：对任何类和成员都公开，无限制访问

private：仅对该类公开

protected：对该类和其派生类公开

internal：只能在包含该类的程序集中访问该类

protected internal：protected + internal

## C#中所有引用类型的基类是什么

引用类型的基类是 System.Object 值类型的基类是 System.ValueType

同时，值类型也隐式继承自 System.Object

## 请简述 ArrayList 和 List<Int>的主要区别

ArrayList 不带泛型 数据类型丢失

List<T> 带泛型 数据类型不丢失

ArrayList需要装箱拆箱 List<T>不需要

## 请简述 GC（垃圾回收）产生的原因，并描述如何避免？

GC 为了避免内存溢出而产生的回收机制

避免：1）减少 new 产生对象的次数

2）使用公用的对象（静态成员）

3）将 String 换为 StringBuilder

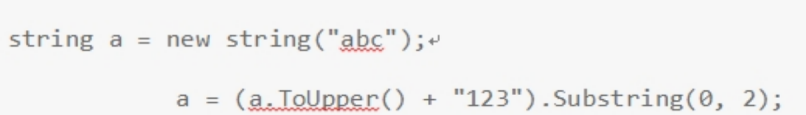
## 请描述 Interface 与抽象类之间的不同

1.接口不是类 不能实例化 抽象类可以间接实例化

2.接口是完全抽象 抽象类为部分抽象

3.接口可以多继承 抽象类是单继承

## 下列代码在运行中会产生几个临时对象？



其实在 C#中第一行是会出错的（Java 中倒是可行）。应该这样初始化：

string b = new string(new char[]{'a','b','c'});

忽略错误的话：

1."abc"

2.a.ToUpper()

3."123"

4.a.ToUpper()+"123"

## 下列代码在运行中会发生什么问题？如何避免？



会产生运行时错误，因为 foreach 是只读的。不能一边遍历一边修改。

## 请简述关键字 Sealed 用在类声明和函数声明时的作用

类声明时可防止其他类继承此类，在方法中声明则可防止派生类重写此方法。

## 反射的实现原理？

可以在加载程序运行时，动态获取和加载程序集，并且可以获取到程序集的信息

反射即在运行期动态获取类、对象、方法、对象数据等的一种重要手段

主要使用的类库：System.Reflection

核心类：

1.Assembly描述了程序集

2.Type描述了类这种类型

3.ConstructorInfo描述了构造函数

4.MethodInfo描述了所有的方法

5.FieldInfo描述了类的字段

6.PropertyInfo描述类的属性

通过以上核心类可在运行时动态获取程序集中的类，并执行类构造产生类对象，动态获取对象的字段或属性值，更可以动态执行类方法和实例方法等。

## .Net 与 Mono 的关系？

.Net是一个语言平台，Mono为.Net提供集成开发环境，集成并实现了.NET的编译器、CLR 和基础类库，使得.Net既可以运行在windows也可以运行于 linux，Unix，Mac OS 等。

## 在类的构造函数前加上 static 会报什么错?为什么?

构造函数格式为 public+类名如果加上 static 会报错（静态构造函数不能有访问修饰符）

原因：静态构造函数不允许访问修饰符，也不接受任何参数；无论创建多少类型的对象，静态构造函数只执行一次；运行库创建类实例或者首次访问静态成员之前，运行库调用静态构造函数；静态构造函数执行先于任何实例级别的构造函数；显然也就无法使用 this 和 base 来调用构造函数。

## C# String 类型比 stringBuilder 类型的优势是什么?

1. 在处理字符串时：String是创建一个string对象，在创建对象时需要分配内存空间。StringBuilder是在原来的内存中修改，不需要分配内存空间。
2. 从内存优化方面来说，频繁对字符串的操作使用StringBuilder更好。
3. 从功能上来说String仍然比StringBuilder更强。
4. String主要用于公共API,通用性好，读取性能高，占用内存小。
5. StringBuilder主要用于拼接String,修改性能好。
6. String是不可变的，所以天然线程同步。
7. StringBuilder可变，非线程同步。

如果是处理字符串的话，用 string 中的方法每次都需要创建一个新的字符串对象并且分配新的内存地址，而 stringBuilder 是在原来的内存里对字符串进行修改，所以在字符串处理

方面还是建议用 stringBuilder 这样比较节约内存。但是 string 类的方法和功能仍然还是比 stringBuilder 类要强。

string 类由于具有不可变性（即对一个 string 对象进行任何更改时，其实都是创建另外一个 string 类的对象），所以当需要频繁的对一个 string 类对象进行更改的时候，建议使用StringBuilder 类，StringBuilder 类的原理是首先在内存中开辟一定大小的内存空间，当对此 StringBuilder 类对象进行更改时， 如果内存空间大小不够， 会对此内存空间进行扩充，而不是重新创建一个对象，这样如果对一个字符串对象进行频繁操作的时候，不会造成过多的内存浪费，其实本质上并没有很大区别，都是用来存储和操作字符串的，唯一的区别就在于性能上。

String 主要用于公共 API，通用性好、用途广泛、读取性能高、占用内存小。

StringBuilder 主要用于拼接 String，修改性能好。

不过现在的编译器已经把 String 的 + 操作优化成 StringBuilder 了， 所以一般用String 就可以了

String 是不可变的，所以天然线程同步。

StringBuilder 可变，非线程同步。

## C# 函数 Func(string a, string b)用 Lambda 表达式怎么写?

(a,b) => {};

## 数列 1,1,2,3,5,8,13...第 n 位数是多少?用 C#递归算法实现

public int CountNumber(int num) {

if (num == 1 || num == 2) {

return 1;

} else {

return CountNumber(num -1) + CountNumber(num-2);

}

}

## 冒泡排序（手写代码）

public static void BubblingSort(int[]array) {

for (int i = 0; i < array.Length; i++){

for (int j = array.Length - 1; j > 0; j--){

if (array[j] < array[i]) {

int temp = array[j];

array[j] = array[j-1];

array[ j - 1] = temp;

}

}

}

}

## C#中有哪些常用的容器类，各有什么特点。

List,HashTable,Dictionary,Stack,Queue

List：索引泛型容器 访问速度快 修改速度慢

HashTable/Dictionary：散列表格式 查询效率高 空间占用较大

Stack：后进先出

Queue: 先进先出

## C#中常规容器和泛型容器有什么区别，哪种效率高？

不带泛型的容器需要装箱和拆箱操作 速度慢 所以泛型容器效率更高 数据类型更安全

## 有哪些常见的数值类？

简单值类型--包括 整数类型、实数类型、字符类型、布尔类型

复合值类型--包括 结构类型、枚举类型

## C#中委托和接口有什么区别？各用在什么场合？

接口（interface）是约束类应该具备的功能集合，约束了类应该具备的功能，使类从千变万化的具体逻辑中解脱出来，便于类的管理和扩展，同时又合理解决了类的单继承问题。

C#中的委托是约束方法集合的一个类，可以便捷的使用委托对这个方法集合进行操作。

在以下情况中使用接口：

1.无法使用继承的场合

2.完全抽象的场合

3.多人协作的场合

以上等等

在以下情况中使用委托：多用于事件处理中

## C#中unsafe关键字是用来做什么的？什么场合下使用？

非托管代码才需要这个关键字 一般用在带指针操作的场合

## C#中ref和out关键字有什么区别？

ref修饰参数，表示进行引用传递，out修饰参数也表示进行引用传递，但传递的引用只为带回返回值 ref又进又出 out不进只出

## For，foreach，Enumerator.MoveNext的使用，与内存消耗情况

for 循环可以通过索引依次进行遍历，foreach和Enumerator.MoveNext通过迭代的方式进行遍历。内存消耗上本质上并没有太大的区别。但是在Unity中的Update中，一般不推荐使用foreach 因为会遗留内存垃圾。

## 函数中多次使用string的+=处理，会产生大量内存垃圾（垃圾碎片），有什么好的方法可以解决。

通过StringBuilder那进行append，这样可以减少内存垃圾

## 当需要频繁创建使用某个对象时，有什么好的程序设计方案来节省内存？

设计单例模式进行创建对象或者使用对象池

## JIT 和AOT区别

Just-In-Time - 实时编译

执行慢 安装快 占空间小一点

Ahead-Of-Time - 预先编译

执行快 安装慢 占内存占外存大

## 给定一个存放参数的数组，重新排列数组

void SortArray(Array arr){Array.Sort(arr);}

## Foreach循环迭代时，若把其中的某个元素删除，程序报错，怎么找到那个元素？以及具体怎么处理这种情况？(注：Try.....Catch捕捉异常，发送信息不可行)

foreach 不能进行元素的删除，因为迭代器会锁定迭代的集合，解决方法：记录找到索引或者key值，迭代结束后再进行删除。

## GameObject a=new GameObject() GameObject b=a 实例化出来了A，将A赋给B，现在将B删除，问A还存在吗？

存在，b删除只是将它在栈中的内存删除，而A对象本身是在堆中，所以A还存在

## 你拥有A块钱，一瓶水B块钱，每瓶水可以得到一个瓶盖，每C个瓶盖可以换一瓶水请写出函数求解上面题目，上面题目ABC为参数

public static int Buy(int a,int b,int c) {

return a/b + ForCap(c,a/b);

}

public static int ForCap(int c,int d) {

if (d<c) {

return 0;

} else {

return d/c + ForCap(c,d/c + d%c);

}

}

## 有一排开关，第一个人把所有的开关打开，第二个人按2的倍数的开关，第三个人按3的倍数的开关，以此类推，现在又n个开关，k个人，写函数求最后等两者的开关，输入参数n和k

static void Main(string[] args) {

int n = int.Parse(Console.ReadLine());

int k = int.Parse(Console.ReadLine());

Function(100,100);

}

static void Function(int n, int k) {

int i, j = 0;

bool[] a = new bool[1000]; //初始false：关灯，true:开灯

for (i = 1; i <= k; i++) //k个人

for (j = 1; j <= n; j++) //n个灯

if (j % i == 0)

a[j] = !a[j]; //取反,false变true,原来开变关，关变开

for (i = 1; i <= n; i++) //最后输出a[i]的值就可以了

if (a[i]) //灯亮着

Console.WriteLine(i);

}

## 数制转换，将任意整数转换成8进制形式

static void Main(string[] args) {

int n;

n =int.Parse(Console.ReadLine());

Console.WriteLine("输入的10进制为：{0}",n);

Console.Write("转换为8进制数为: ");

d2o(n);

}

static void d2o(int n) {

if (n > 7) {

d2o(n / 8);

}

Console.Write(n%8);

}

## 找出200以内的素数。

static void Main(string[] args) {

int count = 0;

for (int i = 1; i < 200; i++) { //外层循环：要判断的数

for (int j = 2; j <=i; j++){

if (i % j == 0&& i!=j) {

break;

}

if (j == i ) { //结束的条件:最后一个数还没有被整除

count++;

Console.WriteLine(i);

}

}

}

Console.WriteLine(count);

}

## 打印杨辉三角形

public static void YHSJ(){

int [][]a= new int[7][] ;

a[0] = new int[1]; //a[0][0]=1;

a[1] = new int[2] ;

for (int i = 0; i < 7; i++) {

a[i] = new int[i+1] ;

a[i][0] =1;

a[i][i]=1;

if(i>1) { //求出中间的数据

for(int j=1;j<i; j++){

a[i][j]= a[i-1][j-1]+a[i-1][j];

}

}

}

for (int i=0; i<a.Length; i++) {

for (int k = 0; k < a.Length-1-i; k++) {

Console.Write("");

}

for(int j=0;j<a[i].Length; j++ ) {

Console.Write(a[i][j] + "");

}

Console.WriteLine();

}

}

## 中国有句俗话“三天打鱼两天晒网”，某人从2000年1月1日起开始“三天打鱼两天晒网”，问这个人在今后的某天中“打鱼”还是”晒网”

public static void Compute(){

Console.WriteLine ((DateTime.Now - DateTime.Parse("2000-01-01")).Days%5<3?"打鱼":"晒网");

}

## 假设当前市场价一只鸡10元，一只鸭12元5角。请写一个函数ShowPrice，输入参数分别为鸡和鸭的个数（非负整型），功能为显示出总价钱，精确到分。例如调用ShowPrice（5,10）后输出175.00。请注意程序的可读性和易于维护性。

static void ShowPrice(int num\_chicken, int num\_duck) {

float totalPrice = 0.00f;

float price\_chicken = 10f;

float price\_duck = 12.5f;

totalPrice = num\_chicken \* price\_chicken + num\_duck \* price\_duck;

Console.WriteLine("总价钱为：{0:0.00}", totalPrice);

}

## 请写一个函数，用于返回n!（阶乘）结果末尾连续0的个数，如GetZeroCount（5）返回1,因为5！ = 120，末尾连续1个0

static void Main(string[] args) {

int fac = Factorial(5);

Console.WriteLine(CountZero(fac));

}

public static int Factorial(int n) {

if (n == 1) {

return 1;

} else {

return n \* jiecheng(n - 1);

}

}

//求连续的0的个数

public static int CountZero(int num) {

int result = 0; // 最后的结果

String numStr = num.ToString();

for (int i = numStr.Length - 1; i >= 0; i--) {

if (numStr[i] == '0') {

result ++;

} else {

break;

}

}

return result;

}

# Unity编辑器基础

## 请描述游戏动画有几种，以及其原理。

主要有关节动画、单一网格模型动画(关键帧动画)、骨骼动画。

关节动画把角色分成若干独立部分，一个部分对应一个网格模型，部分的动画连接成一个整体的动画，角色比较灵活 Quake2 中使用了这种动画。

单一网络模型动画由一个完整的网格模型构成， 在动画序列的关键帧里记录各个顶点的原位置及其改变量，然后插值运算实现动画效果，角色动画较真实。

骨骼动画，广泛应用的动画方式，集成了以上两个方式的优点，骨骼按角色特点组成一定的层次结构，由关节相连，可做相对运动，皮肤作为单一网格蒙在骨骼之外，决定角色的外观。皮肤网格每一个顶点都会受到骨骼的影响，从而实现完美的动画。（骨骼动画是由关节动画发展而来的，如今基本都使用骨骼动画来实现角色动画）

## 物体发生碰撞的必要条件

物体 A 必须带有（collider+rigidbody）或者 CharacterController，另一个物体也必须至少带有 collider

## GUI与UGUI的优点和缺点

GUI不方便控制，UGUI所见即所得，方便控制。GUI使用在生命周期函数OnGUI中使用，脚本来书写控制。UGUI使用Canvas画布和事件系统。UGUI适应屏幕上比GUI简单。

## 一个场景放置多个carmera并同时处于活动状态，会发生什么

实际看到的画面由多个 camera 的画面组成，由 depth、Clear Flag、Culling Mask 都会影响最终合成效果。

## 使用过哪些第三方插件？

一、界面制作 推荐：NGUI

二、2D 游戏制作 推荐：2D Toolkit //[ˈtu:lkɪt] 工具包，工具箱

三、可视化编程 推荐：PlayMaker

四、插值插件 推荐：iTween,HOTween

五、路径搜寻 推荐：Simple Path

六、美术及动画制作 推荐：RageSpline,Smooth Moves

七、画面增强 推荐：Bitmap2Material,Strumpy Shader Editor

八、摄像机管理 推荐：Security Camera

九、资源包 推荐：Nature Pack

十、造路插件 EasyRoads3D

## U3D 中用于记录节点空间几何信息的组件名称，及其父类名称

Transform 父类是 Component

## 为何大家都在移动设备上寻求 U3D 原生 GUI 的替代方案

不美观，OnGUI 很耗费时间，使用不方便 ，DrawCall

## 请简述如何在不同分辨率下保持 UI 的一致性

NGUI 很好的解决了这一点，屏幕分辨率的自适应性，原理就是计算出屏幕的宽高比跟原来的预设的屏幕分辨率求出一个对比值，然后修改摄像机的 size。

## 为什么 dynamic font 在 unicode 环境下优于 static font

Unicode 是国际组织制定的可以容纳世界上所有文字和符号的字符编码方案。使用动态字体时，Unity 将不会预先生成一个与所有字体的字符纹理。当需要支持亚洲语言或者较大的字体的时候，若使用正常纹理，则字体的纹理将非常大。

## Render 的作用？描述 MeshRender 和 SkinnedMeshRender 的关系与不同

Mesh 就是指模型的网格（同名组件是用于调整网格属性的），MeshFilter 一般是用于获得模型网格的组件，而 MeshRender 是用于把网格渲染出来的组件

## 简述 SkinnedMesh 的实现原理

Skinned Mesh 中文一般称作骨骼蒙皮动画.这种动画中包含骨骼（ Bone ）和蒙皮 (Skinned Mesh) 两个部分， Bone 的层次结构和关节动画类似， Mesh 则和关节动画不同：关节动画中是使用多个分散的 Mesh, 而 Skinned Mesh 中 Mesh 是一个整体，也就是说只有一个 Mesh, 实际上如果没有骨骼让 Mesh 运动变形， Mesh 就和静态模型一样了。Skinned Mesh 技术的精华在于蒙皮，所谓的皮并不是模型的贴图。而是 Mesh 本身，蒙皮是指将 Mesh 中的顶点附着（绑定）在骨骼之上，而且每个顶点可以被多个骨骼所控制，这样在关节处的顶点由于同时受到父子骨骼的拉扯而改变位置就消除了裂缝。可以理解为具有蒙皮信息的 Mesh 或可当做皮肤用的 Mesh ，这个皮肤就是 Mesh 。而为了有皮肤功能， Mesh 还需要蒙皮信息，即 Skin 数据，没有 Skin 数据就是一个普通的静态 Mesh 了。 Skin 数据决定顶点如何绑定到骨骼上。顶点的 Skin 数据包括顶点受哪些骨骼影响以及这些骨骼影响该顶点时的权重 (weight) ，另外对于每块骨骼还需要骨骼偏移矩阵 (BoneOffsetMatrix) 用来将顶点从 Mesh 空间变换到骨骼空间。

## Prefab 的作用？如何在移动环境的设备下恰当地使用它？

在游戏运行时实例化，prefab 相当于一个模版，对你已有的素材、脚本、参数做一个默认配置，以便于以后修改，同时 prefab 打包的内容简化了导出的操作，便于团队的交流。

## 如何销毁一个 UnityEngine.Object 及其子类

Destory

## 为什么Unity3D 中会发生在组件上出现数据丢失的情况？

组件上绑定的对象被删除了

## MeshCollider 和其他 Collider 的一个主要不同点？

Meshcollider 是基于顶点的。建议还是用 boxcollider，boxcollider 本身是基于算法，没有面的概念。

## 当一个细小的高速物体撞向另一个较大的物体时，会出现什么情况？如何避免？

穿透（碰撞检测失败）(碰撞体变大，FixedUpdate, 代码限制)

## MeshRender 中 material 和 sharedmaterial 的区别？

修改 sharedMaterial 将改变所有物体使用这个材质的外观，并且也改变储存在工程里的材质设置。

不推荐修改由 sharedMaterial 返回的材质。如果你想修改渲染器的材质，使用 material替代。

## 用 u3d 实现 2d 游戏，有几种方式？

1.利用引擎自带的 GUI和2D系统

2.把摄像机设为 Orthographic，用面片作为 2d 元素

3.利用第三方插件：NGUI、2dToolkit

## u3d 中碰撞器和触发器的区别？

collider 碰撞器会有碰撞的效果，IsTrigger = false, 可以调用OnCollisionEnter/Stay/Exit 函数trigger 触发器没有碰撞效果, isTrigger = true,可以调用 OnTriggerEnter/stay/exit 函数

## CharacterController 和 Rigidbody 的区别

Rigidbody 具有完全真实物理的特性， 而 CharacterController 可以说是受限的 Rigidbody，具有一定的物理效果但不是完全真实的。

## 什么叫做链条关节

Hinge Joint ，他可以模拟两个物体间用一根链条连接在一起的情况，能保持两个物体在一个固定距离内部相互移动而不产生作用力，但是达到固定距离后就会产生拉力。 （简单说就是弹簧）

## unity3d 提供了几种光源，分别是什么

平行光：Directional Light

聚光灯：Spot Light

点光源：Point Light

区域光源：Area Light（只用于烘培）

## u3d 下如何安全的在不同工程迁移 asset 数据

方法 1，可以把 assets 目录和 Library 目录一起迁移，

方法 2，导出包

方法 3，用 unity 带的 assets server 功能

## Unity3d 中的碰撞器和触发器的区别？

触发器只是碰撞器身上的一个属性，碰撞器是触发器的载体。

碰撞器有碰撞的效果，IsTrigger=false，可以调用OnCollisionEnter/Stay/Exit 函数；

触发器没有碰撞效果，IsTrigger=true，可以调用OnTriggerEnter/Stay/Exit 函数。

a.如果不想让碰撞检测影响物体移动但是又想检测到碰撞这时用到触发器（Trigger）

b.触发器用来检测一个物件是否经过空间中的某个区域

## 动画层(Animation Layers)的作用是什么？

动画层作为一个具有层级动画编辑概念的工具， 可以用来制作和处理任何类型的动画

## Material 和 Physic Material 区别？

PhysicMaterial 物理材质：物理材质描述，如何处理物体碰撞（摩擦，弹性）。Material 材质（材质类）为了获得一个对象使用的材质，可以使用 Renderer.materia 属性

## 什么是导航网格（NavMesh）？

一种用于实现自动寻路的网格

## 请简述 NGUI 中 Panel 和 Anchor 的作用

Anchor 包含 UIAnchor 脚本。 UIAnchor 的功能是把对象锚定在屏幕的边缘 （左上， 左中，左下，上，中，下，右上，右中，右下），或缩放物体使其匹配屏幕的尺寸

Panel 对象有 UIPanel 脚本，UIPanel 是一个容器，它将包含所有 UI 小部件，并负责将所包含的部件组合优化，以减少绘制命令的调用。

## Unity 摄像机有几种工作方式，分别是什么？

perspective透视摄像机和orthographic正交摄像机

## LayerMask.NameToLayer()这个方法有什么作用？

LayerMask 的使用是按位操作的，LayerMask.NameToLayer(“Players”) 返回该 Layer的编号。

## NGUI与UGUI的优点和缺点

1) NGUI还保留着图集，需要进行图集的维护。而UGUI没有图集的概念，可以充分利用资源，避免重复资源。

2) UGUI出现了锚点的概念，更方便屏幕自适应。

3) NGUI支持图文混排，UGUI暂未发现支持此功能。

4) UGUI没有 UIWrap 来循环 scrollview 内容。

5) UGUI暂时没有集成Tween组件。

## 是否熟悉UGUI

熟悉，之前在一些项目用UGUI做过界面，和NGUI很相似。

## 会写插件么？

了解过，但在公司这块用的比较少，没怎么写过，不过我能很快学会。

## UGUI和NGUI的区别？为什么不使用NGUI？

1、uGUI的Canvas 有世界坐标和屏幕坐标，NGUI有2D和3D区别。

2、uGUI的Image可以使用material。

3、UGUI通过Mask来裁剪，而NGUI通过Panel的Clip

4、NGUI的渲染前后顺序是通过Widget的Depth，而UGUI渲染顺序根据Hierarchy的顺序，越下面渲染在顶层。

1) 、UGUI 不需要绑定Colliders，UI可以自动拦截事件。

2) 、UGUI的Anchor是相对父对象，没有提供高级选项，个人感觉uGUI的Anchor操作起来比NGUI更方便

7、UGUI没有Atlas一说，使用Sprite Packer。

8、UGUI的Navgation在Scene中能可视化。

9、UGUI的事件需要实现事件系统的接口，但写起来也算简单。

10、NGUI功能更丰富一些

之所以不用NGUI是因为UGUI是Unity官方推出的，慢慢会成为制作UI的主要工具，配套的插件也越来越多，但是具体使用NGUI还是UGUI还要看公司这边，因为这两个我都用过一段时间。

## UI用的是NGUI还是UGUI？

两个都用过。之前用的NGUI后来用了UGUI。

## Unity3d中static batching和dynamic batching 各有什么用？

Dynamic Batching不需要任何操作，只要共享材质(即使是不同的Mesh模型也可以)，就会自动被合并。可以自由移动旋转。但有以下使用要求：

(1) 模型文件共计点数不超过900。（重复使用同一个Mesh不计）

(2) 单个物体可以不超过300点，Shader可以有法线UV。 但如果Shader使用了 UV0 UV1两套UV,或者Tangent切线的话，单个物体只能不超过180点

(3)游戏对象使用相同模型和材质时，只有相同缩放(即xyz等比缩放,浮点尾数可以有细微差)的会被合并。如

（1，1，1）与（1，1，1）

（2，2，2）与（2，2，2）

（0.5，0.5，0.5）与 （0.5，0.5，0.5）

（2,2,2）与（2,2，2.0001）

(4)场景烘焙：烘焙后同材质将不会被烘焙。lightmap 有隐藏的材质参数：offset/scale, 所以使用lightmap的物体不会被合并

(5)Shader不能使用多Pass:多Pass的Shader会破坏Dynamic Batching

Static Batching

原理：运行游戏后将一组游戏对象的多个模型会被动态合并为1个。这组游戏对象所有使用同一材质的在一个DrawCall来完成。这些游戏对象运行后无法移动缩放旋转。但是Drawcall一定是最大化合并的，并且不受动态合并的诸多限制（见下文详述）。

注意：即使物体都使用了同样的模型，在batch后每一个物体都会创建一份模型对应的geometry,在新的Combined Mesh里。所以过多的batch会增加内存占用。例如场景里的树群就不适合Static Batch，而适合动态合并。

实现方法:

(1)MeshRenderer勾选Batching Static： 勾了即可

(2)代码中使用UnityEngine.StaticBatchingUtility实现(可以在任何平台调用):

1)将所有要合并的静态物体(不须勾Batching Static)放入统一一个root

2)StaticBatchingUtility.Combine(root); 之后就合并好了！

区别：勾选Batching Static：完全自动合并,在MeshFilter里显示的是 Combined Mesh(root:scene)。合并后不能移动

StaticBatchingUtility：合并到一个游戏对象下。合并后可以移动父节点游戏对象"

## Unity3d中Awake和Start 谁先执行，update和fixedUpdate 有什么区别？

Awake先执行。Update是在每次渲染新的一帧的时候才会调用，FixedUpdate，是在固定的时间间隔执行，不受游戏帧率（fps）的影响 ，FixedUpdate的时间间隔可以在项目设置中更改，Edit->Project Setting->time 找到Fixed timestep。就可以修改了

## Unity引擎中以下对Mesh Renderer组件描述正确的是哪一项？ C

A．Mesh Renderer组件决定了场景中游戏对象的位置，旋转和缩放

B．为场景中的某一游戏对象增添物理的特性，需要为该游戏对象添加Mesh Renderer组件

C．Mesh Renderer组件从Mesh Filter组件中获得网格信息，病根据物体的Transform组件所定义的位置进行渲染

D．Mesh Renderer是从网格资源中获取网格信息的组件

## 哪种实时光源是 Unity 中没有的？D

A:点光源

B:方向光

C:聚光灯

D:日光灯

## 如何在 Unity 中创建地形系统？A

A：Terrain->Create Terrain

B：Component->Create Terrain

C：Asset->Create Terrain

D：Windows->Create Terrain

## 以下哪种操作步骤可以在场景中添加“Wind Zone”？B

A：Terrain -> Wind Zone

B：GameObject -> Create Other -> Wind Zone

C：Component -> Physics -> Wind Zone

D：Assets -> Create -> Wind Zone

## 在 Unity 编辑器中创建一个 Directional Light，以下步骤正确的是？ B

A：Edit -> Rendering Setting -> Directional Light

B：GameObject -> Create Other -> Directional Light

C：Component -> Rendering -> Directional Light

D：Assets -> Directional Light

## 下列哪一项不属于 Camera 中的“Clear Flags”？D

A：Skybox

B：Solid Color

C：Depth Only

D：Background

## 以下哪种脚本语言是 Unity 编辑器所不支持的？ D

A：JavaScript

B：C#

C：Boo

D：Perl

## 对于 Prefab，以下说法错误的是？ D

A：Prefab 资源可以在项目中多次重复使用

B：由 Prefab 实例出的 GameObject，其在 Hierarchy 视图中表现为蓝色

C：Prefab 上的组件信息一经改变，其实例出的 GameObject 也会自动改变

D：实例出的 GameObject 上的组件信息一经改变，其对应的 Prefab 也会自动改变

## 下面哪种做法可以打开 Unity 的 Asset Store？ A

A：Windows -> Asset Store

B：Edit -> Asset Store

C：File -> Asset Store

D：Assets -> Asset Store

## 在哪个面板中可以修改物体的空间属性，如位置、朝向、大小等？B

A：Project

B：Inspector

C：Hierarchy

D：Toolbar

## 如何为一个 Asset 资源设定一个 Label，从而能够方便准确的搜索到? D

A：在 Project 窗口中选中一个 Asset,右键->Create->Label

B：在 Project 窗口中选中一个 Asset,右键->Add Label

C：在 Project 窗口中选中一个 Asset,在 Inspector 窗口中点击添加 Label 的图标

D：在 Project 窗口中选中一个 Asset,在 Inspector 窗口中点击按钮“Add Label”

## Mecanim 系统中，Body Mask 的作用是？D

A:指定身体的某一部分是否参与骨骼动画

B:指定身体的某一部分是否参与物理模拟

C:指定身体的某一部分是否可以输出骨骼信息

D:指定身体的某一部分是否参与渲染

## 以下哪种操作步骤可以打开 Unity 编辑器的 Lightmapping 视图？ C

A：File --> Lightmapping

B：Assets --> Lightmapping

C：Windows --> Lightmapping

D：Component --> Lightmapping

## 下列关于光照贴图，说法错误的是？ C

A：使用光照贴图比使用实时光源渲染要快

B：可以降低游戏内存消耗

C：可以增加场景真实感

D：多个物体可以使用同一张光照贴图

## 如何为物体添加光照贴图所使用的 UV？ B

A：不用添加，任何时候都会自动生成

B：更改物体导入设置，勾选“Generate Lightmap UVs”

C：更改物体导入设置，勾选“Swap UVs”

D：更改物体导入设置，在 UVs 选项中选择“Use Lightmaps”

## 在哪个模块下可以修改 Render Path？ A

A：Camera

B：Light

C：Render Settings

D：Project Settings->Quality

## 以下哪项技术不是目前 Unity 所支持的 Occlusion Culling 技术？ D

A：PVS only

B：PVS and dynamic objects

C：Automatic Portal Generation

D：Dynamic Only

## 关于 Vector3 的 API，以下说法正确的是？ C

A：Vector3.normalize 可以获取一个三维向量的法线向量；

B：Vector3.magnitude 可以获取一个三维向量的长度；

C：Vector3.forward 与 Vector3（0,0,1）是一样的意思；

D：Vector3.Dot（向量 A，向量 B）是用来计算向量 A 与向量 B 的叉积

## 下列那些选项不是网格层属性的固有选项? A

A：Default

B：Walkable

C：Not Walkable

D：Jump

## 什么是导航网格（NavMesh）？ D

A：一种用于描述相机轨迹的网格

B：一种被优化过的物体网格

C：一种用于物理碰撞的网格

D：一种用于实现自动寻路的网格

## 以下哪一个选项不属于Unity引擎所支持的视频格式文件 D

A．后缀名为mov的文件

B．后缀名为mpg的文件

C．后缀名为avi的文件

D. 后缀名为swf的文件

## Unity引擎使用的是左手坐标系还是右手坐标系？ A

A．左手坐标系

B．右手坐标系

C．可以通过Project Setting 切换左右手坐标系

D．可以通过Reference切换左右手坐标系

## 以下哪组摄像机中Normalized View Port Rect的数值设置可以使得摄像机显示的画面位于1280\*720分辨率的屏幕画面的右上角 D

A. x = 640, Y = -360 , W = 360,H = 360

B. x =640, Y = -360 , W = 360,H = 360

C. x =0.5, Y = 0 , W = 0.5,H = 0.5

C. x =0.5, Y = 0.5 , W = 0.5,H = 0.5

## 以下哪个组件是任何GameObject必备的组件 B

A.Mesh Renderer

B.Trtansform

C.Game Object

D.Main camera

## 在Unity编辑器中，停止对Game视图进行预览播放的快捷键操作是以下哪一项 A

A．CTRL/CMD + P

B．CTRL/CMD + Shift +P

C．CTRL/CMD + Alt +P

D．CTRL/CMD + S

## 在Unity引擎中，Depth属性值最大的摄像机会比Depth属性值小的摄像机更晚绘制么？ A

A．是

B．否

## 下列选项中有关Animator的说法错误的是 D

A.Animasstor是Unity引擎内置组件

B.任何一个具有动画状态机功能的GameObject都需要一个Animator组件

C.它主要用于角色行为的设置，包括StateMachines，混合Blend trees以及通过脚本控制的事件

D.Animator同Animation组件的用法是相同的

## 下列哪个视图主要用于显示和编辑所选游戏对象或资源的相关属性 C

A．Scene

B．Project

C．Inspector

D．Hierarchy

## 简述Prefab的使用？Editor下动态创建Prefab的方式 "u3d里动态创建对象，需要使用prefab而创建的时候 MonoBehaviour.Instantiate (GameObject orignal) 需要一个作为原型的对象。三种方式获得prefab对象。

方式一：使用脚本的public字段

直接在Project视图里找到做好的prefab，将其拖拽到指定脚本的指定public GameObject 字段。

方式二：Resource类

1、在Assets目录下的任意位置创建一个名为resources的文件夹，将做好的prefab放到这个文件夹下，path形式如下: Assets\....\resources\prefabName.prefab

2、在代码里使用Resource.Load 或 LoadAll 函数，获得原型对象。指定prefab时不需要指定扩展名（.prefab），形式如下：GameObject prototype = Resource.Load(“prefabName”) as GameObject; 可以有任意数量的resources文件夹，怀疑是Resource类初始化的时候会搜集所有resources文件夹里的文件名。

方式三：加载到场景

一般我们制作Perfab的时候，都是在Hierarchy视图里创建GameObject，然后再搭建Prefab。事后根据需要删除这个原始的GameObject。因此我们可以保留这个GameObject，然后在场景加载后Find这个对象（代码方式），或者使用脚本public字段（编辑器方式）

## NGUI的自适应性是？如果此时屏幕比例变化，屏幕出现黑边怎么办？（注：改变NGUI和UGUI的Fixed size with Screen不可行）

NGUI根目录的UIRoot组件自带了根据高度自适应分辨率的功能。

Scaling Style属性可选择三种不同的缩放策略。

PixelPerfect 完美像素：直接显示设定好的像素。当屏幕高度低于minimum Height时按比例缩小，当屏幕高度大于maximum Height时按比例扩大。

FixedSize 按比例缩放：在设定好的基础上，直接按比例缩放。

FixedSizeOnMobiles 合体版，android和ios为FixedSize方式，其它按照PixelPerfect方式。

# Unity脚本基础

## Unity3D中的协程（coroutine）和C#线程之间的区别是什么？

（1）多线程程序同时运行多个线程 ，而在任一指定时刻只有一个协程在运行，并且这个正在运行的协同程序只在必要时才被挂起。

（2）除主线程之外的线程无法访问 Unity3D 的对象、组件、方法。

（3）Unity3d 没有多线程的概念，不过 unity 也给我们提供了 StartCoroutine（协同程序）和LoadLevelAsync（异步加载关卡）后台加载场景的方法。

（4）StartCoroutine 为什么叫协同程序呢，所谓协同，就是当你在 StartCoroutine 的函数体里处理一段代码时，利用 yield 语句等待执行结果，这期间不影响主程序的继续执行，可以协同工作。而 LoadLevelAsync 则允许你在后台加载新资源和场景，所以再利用协同，你就可以前台用 loading 条或动画提示玩家游戏未卡死，同时后台协同处理加载的事宜

## 简述 Unity3D 支持的作为脚本的语言的名称

Unity 的脚本语言基于 Mono 的.Net 平台上运行，可以使用.NET 库，这也为 XML、数据库、正则表达式等问题提供了很好的解决方案。

Unity 里的脚本都会经过编译，他们的运行速度也很快。这三种语言实际上的功能和运行速度是一样的，区别主要体现在语言特性上。

JavaScript：和网页中常用的 JavaScript 不一样，它编译后的运行速度很快，语法方面也会有不少区别。

C#

Boo：可以看做是 Python 语言的变种，又糅合了 Ruby 和 C#的特性，它是静态类型语言

## Unity3D 是否支持写成多线程程序？如果支持的话需要注意什么？

仅能从主线程中访问 Unity3D 的组件，对象和 Unity3D 系统调用

## 支持：如果同时你要处理很多事情或者与 Unity 的对象互动小可以用 thread,否则使用coroutine。

注意：C#中有 lock 这个关键字,以确保只有一个线程可以在特定时间内访问特定的对象

## OnEnable、Awake、Start 运行时的发生顺序？哪些可能在同一个对象周期中反复的发生？

Awake -》OnEnable-》Start

OnEnable 在同一周期中可以反复地发生

## 请简述 OnBecameVisible 及 OnBecameInvisible 的发生时机，以及这一对回调函数的意义？

当物体是否可见切换之时。可以用于只需要在物体可见时才进行的计算。

## Unity3D 如何获知场景中需要加载的数据？

Resource.Load

AssetBundle.Load

## 物体发生碰撞时，有几个阶段，分别对应的函数

三个阶段，OnCollisionEnter/Stay/Exit 函数

## u3d 中，几种施加力的方式，描述出来

rigidbody.AddForce/AddForceAtPosition，都是 rigidbody 的成员函数

## 物体自旋转使用的函数叫什么

transform.Rotate

## 物体绕某点旋转使用函数叫什么

transform.RotateAround

## u3d 提供了一个用于保存读取数据的类，（playerPrefs），请列出保存读取整形数据的函数

PlayerPrefs.SetInt 与 PlayerPrefs.GetInt

## unity3d 从唤醒到销毁有一段生命周期，请列出系统自己调用的几个重要方法。

Awake –>OnEnable –> Start –> Update –> FixedUpdate –> LateUpdate –>OnGUI –> Reset –> OnDisable –> OnDestroy

## 物理更新一般在哪个系统函数里？

FixedUpdate，每固定帧绘制时执行一次，和 update 不同的是 FixedUpdate 是渲染帧执行，如果你的渲染效率低下的时候 FixedUpdate 调用次数就会跟着下降。FixedUpdate 比较适用于物理引擎的计算，因为是跟每帧渲染有关。Update 就比较适合做控制。

## 移动相机动作在哪个函数里，为什么在这个函数里。

LateUpdate,，是在所有 update 结束后才调，比较适合用于命令脚本的执行。官网上例子是摄像机的跟随，都是在所有 update 操作完才跟进摄像机，不然就有可能出现摄像机已经推进了，但是视角里还未有角色的空帧出现。

## 为什么 u3d 会出现组件上数据丢失的情况

一般是组件上绑定的物体对象被删除了

## 什么是协同程序？

在主线程运行时同时开启另一段逻辑处理，来协助当前程序的执行。换句话说，开启协程就是开启一个线程。可以用来控制运动、序列以及对象的行为。

## 反向旋转动画的方法是什么？

反转动画，讲动画的速度调到-1，碰撞时，被碰撞物体与碰撞物体有collider 组件，碰撞物体有刚体组件，或角色碰撞得包含角色组件 OR 改变 animation.speed

## 用代码实现第三角色控制器

public class Player : MonoBehaviour {

public Transform \_cameraTrans;

private Vector3 \_cameraOffset;

void Awake() {

\_cameraOffset = transform.position - \_cameraTrans.position;

}

void Update() {

\_cameraTrans.position = transform.position - \_cameraOffset;

}

}

## 实现吊机吊物体的功能

吊机吊物体需要节点挂接和坐标系转换

## 获取、增加、删除组件的命令分别是什么

获取：GetComponent

增加：AddComponent

删除：Destroy

## Animation.CrossFade 命令作用是：C

1. 动画放大 B.动画转换 C.动画的淡入为其他动画

## Application.loadLevel 命令为：A

1. 加载关卡 B.异步加载关卡 C.加载动作

## 调试记录到控制台的命令是什么？

Debug.Log();

## 编辑器类存放路径是什么？

工程目录下的 Assets/Editor 文件夹下。

## 使用原生 GUI 创建一个可以拖动的窗口命令是什么？

GUI.DragWindow();

## localPosition 与 Position 的使用区别？

localPosition：自身位置，相对于父级的变换的位置。 Position：在世界坐标 transform的位置

## 含义Mathf.Round, Mathf.Clamp, Mathf.Lerp

Mathf.Round 四舍五入

Mathf.Clamp 限制

Mathf.Lerp 插值

## 写一个计时器工具，从整点开始计时，格式为：00:00:00

private float timer = 0f;

private int h = 0;

private int m = 0;

private int s = 0;

private string timeStr = string.Empty;

void Update () {

timer += Time.deltaTime;

if (timer >= 1f) {s++; timer = 0;}

if (s >= 60) {m++;s = 0;}

if (m >= 60) {h++;m = 0;}

if (h >= 99) {h = 0;}

}

void OnGUI(){

timeStr = string.Format ("{0:D2}:{1:D2}:{2:D2}", h, m, s);

GUI.Label (new Rect (10, 10, 100, 200), timeStr);

}

## 写出 Animation 的五个方法

AddClip 添加剪辑、Blend 混合、Play 播放、Stop 停止、Sample 采样

## 用鼠标实现在场景中拖动物体，用鼠标滚轮实现缩放(用一个 Cube 即可)。

在场景中添加一个Plan，Camera，Directional Light，Cube。添加两个脚本scrollerScirpt（挂在Camera），CubeDragScript（挂在Cube上）。

1.鼠标滚轮实现缩放：将摄像机的镜头拉近或者拉远，调整摄像机的视角就可以实现，主要实现代码如下：

void Update () {

//鼠标滚轮的效果

if (Input.GetAxis("Mouse ScrollWheel") < 0) {

if (Camera.main.fieldOfView <= 100)

Camera.main.fieldOfView += 2;

if (Camera.main.orthographicSize <= 20)

Camera.main.orthographicSize += 0.5F;

}

//Zoom in

if (Input.GetAxis("Mouse ScrollWheel") > 0) {

if (Camera.main.fieldOfView > 2)

Camera.main.fieldOfView -= 2;

if (Camera.main.orthographicSize >= 1)

Camera.main.orthographicSize -= 0.5F;

}

}

2.鼠标实现在场景中拖动物体：

　　解决思路就是将世界坐标转换成屏幕坐标，然后计算物体与鼠标之间移动量，循环鼠标被按下操作，得到鼠标的当前位置，加上计算好的移动量，将新的坐标赋值给物理就行了。主要是开启一个协同程序（Corountine）来处理

主要代码如下：

// Use this for initialization

void Start () {

StartCoroutine(OnMouseDown());

}

IEnumerator OnMouseDown() {

//将物体由世界坐标系转换为屏幕坐标系

Vector3 screenSpace = Camera.main.WorldToScreenPoint(transform.position);

//完成两个步骤 1.由于鼠标的坐标系是2维，需要转换成3维的世界坐标系

//2.只有3维坐标情况下才能来计算鼠标位置与物理的距离，offset即是距离

//将鼠标屏幕坐标转为三维坐标，再算出物体位置与鼠标之间的距离

Vector3 offset = transform.position - Camera.main.ScreenToWorldPoint(new Vector3(Input.mousePosition.x, Input.mousePosition.y, screenSpace.z));

while (Input.GetMouseButton(0)) {

//得到现在鼠标的2维坐标系位置

Vector3 curScreenSpace = new Vector3(Input.mousePosition.x, Input.mousePosition.y, screenSpace.z);

//将当前鼠标的2维位置转换成3维位置，再加上鼠标的移动量

Vector3 curPosition = Camera.main.ScreenToWorldPoint(curScreenSpace) + offset;

//curPosition就是物体应该的移动向量赋给transform的position属性

transform.position = curPosition;

yield return new WaitForFixedUpdate(); //这个很重要，循环执行

}

}

## NGUI Button 怎样接受用户点击并调用函数,具体方法名称是什么

1、主要是在UICamera脚本中用射线判断点击的物体并通过SendMessage调用OnClick() OnPress()等函数，可以说NGUI的按钮是通过发消息这个方式调用的。具体方法名称是OnClick()

2、void Awake () {

//获取需要监听的按钮对象

GameObject button = GameObject.Find("UI Root/Button3");

//设置这个按钮的监听，指向本类的ButtonClick方法中。

UIEventListener.Get(button).onClick = OnButton3Click;

}

private void OnButton3Click(GameObject button) {

Debug.Log("我是按钮3被点击了");

}

## <愤怒的小鸟>给予初速度以后,怎么让小鸟受到重力和空气阻力的影响而绘制抛物线轨迹,说出具体的计算方法.

Vector3 v 代表初速度 v'代表现在的速度， 假设小鸟是沿的 z 轴也就是 transform.forward方向运动的质量为 1，那么 v'=v-new Vector3(0,g\*t,f\*t)，transform.Translate(v')做的就是抛物线运动（g 为重力加速度不要用现实中的需要自己调试，f 为阻力也要自己调试设置，t 为时间）

## 当游戏中需要频繁创建一个物体时，我们需要怎样做能够节省内存？

1、使用预制物体对象 Prefab

2、使用对象池技术，不使用时关闭，使用时打开

## 碰撞检测需要物体具备什么属性？

能检测碰撞发生的方式有两种，一种是利用碰撞器，另一种则是利用触发器

【Physics.OverlapSphere 相交球检测碰撞，碰撞检测需要包围盒】

## Vector3.forward与Vector3(0,0,1)是一样的意思对吗？

对

## 下哪个函数在游戏进入新场景后会被马上调用？

MonoBehaviour.OnLevelWasLoaded

## itween 插件的作用是什么，itween 作用于世界坐标还是局部坐标，请列举出 3 个其常用方法？

iTween 是一个动画库，作者创建它的目的就是最小的投入实现最大的产出。让你做开发更轻松，用它可以轻松实现各种动画，晃动，旋转，移动，褪色，上色，控制音频等等“方法：

a、MoveTo 物体移动；

b、ColorTo：随着时间改变对象的颜色组；

c、LookTo：随时间旋转物体让其脸部朝向所提供的 Vector3 或 Transform 位置；

## U3D 中用于记录节点空间几何信息的组件名称，及其父类名称

Transform 父类是 Component

## NGUI:把子控件放在父控件里面，如何上下边距都是10。

给子控件设置上下两个锚点为10

## 如何使子控件居中,如果使用UGUI怎么实现

设置子控件锚点为中心

## 去掉敏感字的程序（手写程序）

String s = "你是坏蛋";

s.Replace("坏蛋", "\*\*");

## unity3D从唤醒到销毁有一段生命周期，列出系统自己调用的重要方法。

Awake ()

OnEnable ()

Start()

FixedUpdate()

OnTriggerXXX(Collider other)

OnCollisionXXX (Collision collisionInfo)

Update()

LateUpdate ()

OnGUI()

OnDisable ()

OnDestroy ()

## Unity3d中resources目录一般用来放些什么，打包的时候会有什么影响？

resource一般用来放置一些需要动态加载的资源，打包程序的时候会将Resource目录下的全部文件都加密压缩打进包内，这样再想使用assetbundle方式打小包的话就不行了

## 空间内一物体绕球面固定点（0、1、2）按照固定速度speed环绕运动。

public float Speed = 1;

void Update () {

transform.RotateAround (new Vector3(0,1,2),Vector3.up,Speed);

}

## 以下关于WWW.LoadFromCacheOrDownload描述正确的是： C

A．可被用于将Text Assets自动缓存到本地磁盘

B．可被用于将Resource自动缓存到本地磁盘

C．可被用于将Assets Bundles自动缓存到本地磁盘

D．可被用于将任意的Unity资源文件自动缓存到本地磁盘

## 以下哪个函数在游戏进入新场景后会被马上调用？ C

A：MonoBehaviour.OnSceneWasLoaded

B：MonoBehaviour.OnSceneEnter

C：MonoBehaviour.OnLevelWasLoaded

D：MonoBehaviour.OnLevelEnter

## 关于MonoBehavior.LateUpdate函数描述错误的是： B

A．当MonoBehavior类型应用后，每帧调用一次

B．常被用于处理RigidBody的更新

C．在所有Update函数执行后才能被调用

D．常被用于实现跟随相机效果，且目标物体的位置已经在Update函数中被更新

## 某个GameObject有一个名为MyScript的脚本，该脚本中有一个名为DoSomething的函数，则如何在该GameObject的另一个脚本中国调用该函数？ A

A．GetComponent<MyScript>().DoSomething()

B．GetComponent<Script>("MyScript").DoSomething()

C．GetComponent< MyScript >().Call("DoSomething")

D．GetComponent<Script>("MyScript"). Call("DoSomething")

## 启用MipMaps对内存的影响是？ A

A．增加约33%的内存

B．减少约33%的内存

C．增加约25%的内存

D．减少约25%的内存

## 以下关于MonoBehaviour.OnGUI()的描述的是： D

A．如果MonoBehaviour没有被启用，则OnGUI函数不会被调用

B．用于绘制和处理GUI events

C．每帧可能会被绘制多次，每次对应于一个GUI event

D．每帧被调用一次

## 采用Input.mousePosition来获取鼠标在屏幕上的位置，以下表述正确的是： B

A．左上角为原点(0, 0)，右下角为(Screen.Width, Screen.Height)

B．左下角为原点(0, 0)，右上角为(Screen.Height, Screen.Width)

C．左下角为原点(0, 0)，右上角为(Screen.Width, Screen.Height)

D．左上角为原点(0, 0)，右下角为(Screen.Height, Screen.Height)

## 正确排列出下面Unity脚本自带的函数执行顺序Start，Awake，Update，OnUpdate，OnEnable，FixedUpdate，OnGUI，LateUpdate、OnDisable、OnDestory

Awake-->OnEnable-->Start-->Update-->FixedUpdate-->LateUpdate-->OnGUI-->OnDisable -->OnDestory

## Unity中销毁GameObject的方式，简述Destroy和DestroyImmediate的区别

Destroy销毁场景中的物体，但是内存中它还是存在的，只有当内存不够时，机制才会将它销毁并释放内存。而DestroyImmediate会立即将对象销毁，并且将内存释放。

## Unity中如何派发事件（消息）

在脚本里的Update函数里调用EventDispatcher.Instance().OnTick();就可以了

## ScriptObject的作用和使用方式

ScriptObject类型经常使用于存储一些Unity本身不可以打包的一些object，比如字符串，一些类对象，用这个类型的子类型可以用BuildPipeline打包成assetbundle包共后续使用，非常方便。

## 如何检测物体是否被其他对象遮挡

使用射线进行检测

## 写一个角色控制器，鼠标控制屏幕晃动，鼠标控制开枪。

public class Player : MonoBehaviour {

public GameObject \_prefabBullet;

private float \_angleSpeed = 120f;

void Update() {

float eularY = Input.GetAxis("Mouse X") \* \_angleSpeed \* Time.deltaTime;

transform.Rotate(new Vector3(0, eularY, 0));

if (Input.GetMouseButtonDown(0)) {

Instantiate(\_prefabBullet, transform.position, transform.rotation);

}

}

}

## 敌人AI，有各种状态，实现各种状态之间的切换。

通过状态机来实现各种状态之间的切换

## 1.写一个角色控制器，鼠标控制屏幕晃动，鼠标控制开枪。

**2.敌人AI，有各种状态，实现各种状态之间的切换。**

**3.敌人会和主角对抗，敌人被打到之后，会闪一次红色，然后红色比例提升10%，10次攻击之后，成红色。**

**4.敌人会自动攻击主角，主角也会有颜色变化。**

**5.敌人会在范围内巡逻。**

**6.UI，左边显示8个AI的被攻击次数，右边显示AI的攻击次数排序。**

public class Player : MonoBehaviour {

public Camera \_camera;

public GameObject \_prefabBullet;

private float \_angleSpeed = 120f;

//生命值

private int \_life = 10;

//玩家的状态

private bool \_state = false;

//是否被打到

public void IsState() {

if (\_state && \_life > 0) {

\_life -= 1;

\_state = false;

} else {

\_state = true;

}

}

public void RoleRotate() {

float eularY = Input.GetAxis("Mouse X") \* \_angleSpeed \* Time.deltaTime;

}

//风发射子弹

public void RoleShoot() {

if (Input.GetMouseButtonDown(0)) {

Instantiate(\_prefabBullet, transform.position, transform.rotation);

}

}

}

public class Enemy : MonoBehaviour {

//敌人被打到状态

private bool \_state = false;

//玩家与敌人距离

private float \_distance;

//角色

public GameObject \_role;

//生命值

private int \_life = 10;

//敌人被攻击次数

private int \_timeAccack = 0;

//敌人状态

public void EnemyState() {

if (\_distance >= 10f) {

if (\_life >= 1 && \_state == true) {

\_life -= 1;

\_timeAccack++;

}

\_state = false;

}

else if (\_distance >= 0 && \_state == false) {

\_state = true;

}

}

//敌人与玩家距离

public void Distance() {

\_distance = Vector3.Distance(transform.position, \_role.transform.position);

}

void OnGUI() {

GUI.TextArea(new Rect(20, 50, 80, 30), \_timeAccack.ToString());

GUI.TextArea(new Rect(20, 90, 80, 30), \_timeAccack.ToString());

GUI.TextArea(new Rect(20, 130, 80, 30), \_timeAccack.ToString());

}

}

## 3D空间有三个cube当做点，有一条鱼的模型，要求在三点之间游动，要求转向平滑一点，控制鱼的运动朝向（用四元数和欧拉角）

使用transform.localRotation = Quaternion.Slerp(Quaternion a,Quaternion b,float c)实现物体平滑转向

# Unity性能优化

## lod 是什么，优缺点是什么

LOD 技术即 Levels of Detail 的简称，意为多细节层次。LOD 技术指根据物体模型的节点在显示环境中所处的位置和重要度，决定物体渲染的资源分配，降低非重要物体的面数和细节度，从而获得高效率的渲染运算。

优点：可根据距离动态地选择渲染不同细节的模型

缺点：加重美工的负担，要准备不同细节的同一模型，同样的会稍微增加游戏的容量。

## MipMap 是什么？作用？

在三维计算机图形的贴图渲染中有一个常用的技术被称为 Mipmapping。为了加快渲染速度和减少图像锯齿，贴图被处理成由一系列被预先计算和优化过的图片组成的文件,这样的贴图被称为 MIP map 或者 mipmap。

## 当游戏中需要频繁创建一个物体对象时，我们需要怎么做来节省内存。

做一个 pool，游戏开始时预先实例化足够的数量，然后用的时候取不用的时候收回

## 如何优化内存？

有很多种方式，例如

1.压缩自带类库；

2.将暂时不用的以后还需要使用的物体隐藏起来而不是直接 Destroy 掉；

3.释放 AssetBundle 占用的资源；

4.降低模型的片面数，降低模型的骨骼数量，降低贴图的大小；

5.使用光照贴图，使用多层次细节(LOD)，使用着色器(Shader)，使用预设(Prefab)。

## 动态加载资源的方式？和区别

1.通过 Resources 模块，调用它的 load 函数：可以直接 load 并返回某个类型的 Object，前提是要把这个资源放在 Resource 命名的文件夹下，Unity不关有没有场景引用，都会将其全部打入到安装包中。Resources.Load();

2.通过 bundle 的形式： 即将资源打成 asset bundle 放在服务器或本地磁盘， 然后使用 WWW 模块 get 下来， 然后从这个 bundle 中 load 某个 object。

AssetBundle

3.通过 AssetDatabase.loadasset ：这种方式只在 editor 范围内有效，游戏运行时没有这个函数， 它通常是在开发中调试用的 【AssetDatabase 资源数据库】

区别：Resources 的方式需要把所有资源全部打入安装包，这对游戏的分包发布（微端）和版本升级（patch）是不利的，所以 unity 推荐的方式是不用它， 都用 bundle 的方式替代， 把资源达成几个小的 bundle， 用哪个就 load哪个，这样还能分包发布和 patch，但是在开发过程中，不可能没更新一个资源就打一次 bundle， 所以 editor 环境下可以使用 AssetDatabase 来模拟，这通常需要我们封装一个 dynamic resource 的 loader 模块，在不同的环境下做不同实现。

动态资源的存放

有时我需要存放一些自己的文件在磁盘上，例如我想把几个 bundle 放在初始的安装里， unity 有一个 streaming asset 的概念，用于提供存储接口的访问。我们需要在编辑器建立一个 StreamingAssets 名字的文件夹，把需要我们放在客户磁盘上的动态文件放在这个文件夹下面，这样安装后，这些文件会放在用户磁盘的指定位置，这个位置可以通过Application.streamingAssetsPath 来得到。

## 请简述一下对象池原理,什么情况下使用？

对象池背后的理念其实是非常简单的。我们将对象存储在一个池子中，当需要时在再次使用，而不是每次都实例化一个新的对象。池的最重要的特性，也就是对象池设计模式的本质是允许我们获取一个“新的”对象而不管它真的是一个新的对象还是循环使用的对象。

对象池通常用于在那些需要经常创建又销毁的对象比如我们场景中的小怪，经常需要创建和销毁，我们就可以在小怪销毁的时候，不真正的销毁而是，而是把它放入对象池中等到下一次我们需要创建小怪的时候，再从对象池中取出来。

## 使用mipmap有什么好处？什么情况下使用？

Mipmap纹理技术是目前解决纹理分辨率与视点距离关系的最有效途径,它会先将图片压缩成很多逐渐缩小的图片,例如一张64\*64的图片,会产生 64\*64,32\*32,16\*16,8\*8,4\*4,2\*2,1\*1的7张图片,当屏幕上需要绘制像素点为20\*20 时，程序只是利用 32\*32 和 16\*16 这两张图片来计算出即将显示为 20\*20 大小的一个图片，这比单独利用 32\*32 的那张原始片计算出来的图片效果要好得多，速度也更快。

## Unity内存优化？

GC垃圾回收

## 你认为unity在开发过程中哪些地方比较容易造成内存泄漏和内存泄漏问题？如何避免？

脚本做载入不删除处理时，往往脚本中还有一些不用的变量，容易导致内容泄漏

## 使用NGUI开发滚动视图，当滚动列表达到几百条的时候，如何才能保证界面的滚动流畅运行

1. 每个滚动条目都是同一个预设体的实例

2.做缓存，只要实例化视野范围内滚动条目，往上超出视野部分，自动填补到视野的下面

## 如何解决过多创建和删除对象带来的卡顿问题

对象池，把不用对象缓存到一个对象列表中

## Unity资源加载的有几种方式，简述asset bundie

1. 直接在脚本中public一个对象，然后在监视器面板进行赋值

2. 直接在程序中进行find查找

3. Resource.load

4. 把资源打成Assetbudle，然后用的时候load进来

assetbundle就是对资源的打包处理，同时这种资源格式便于从互联网上下载

## 背包系统中只有20个格子，现在有总共有100个物体，除了显示在视野中的20个外，对其他的处理方法？（注：将其他隐藏起来不可行，对象池得有具体的说明）

1. 每个滚动条目都是同一个预设体的实例

2. 做缓存，只要实例化视野范围内滚动条目，往上超出视野部分，自动填补到视野的下面