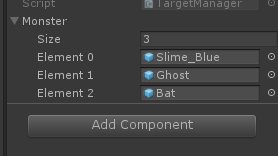
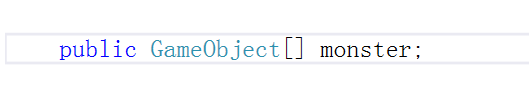
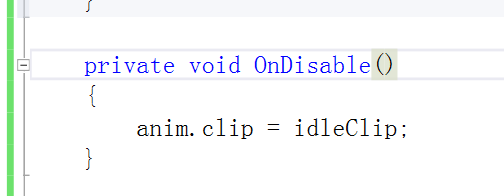
# 1,添加多个游戏物体





# 2，OnDisable

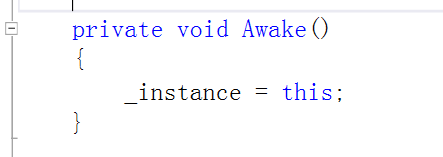


前提：物体先处于激活状态，当被射击后，处于未激活状态。

能在游戏物体取消激活时，执行函数。

# 3,单例模式注意点

public static TargetManager \_intance;



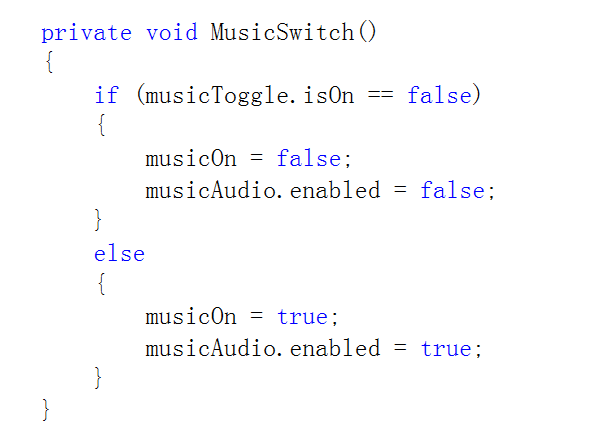
只是一个空对象，必须为他赋值，不然就是空指针。

# 4,FirePosition



例如像子弹发射，要实例化子弹，需要一个初始化位置，此时就需要一个firePosition。

# 5,例如背景音乐的开关



# 6, PlayerPrefs存储数据

playerPrefs 只能存储string float int类型

PlayerPrefs.SetInt("MusicIsOn", 0);

if (PlayerPrefs.HasKey("MusicIsOn"))

{

if (PlayerPrefs.GetInt("MusicIsOn") == 1)

{

}

}

# 7，二进制存储与读取

## （1）存储

private void SaveByBin()

{

//序列化过程

//将游戏状态保存为一个类

Save save = CreateSaveGO();

//创建一个二进制格式化程序

BinaryFormatter bf = new BinaryFormatter();

//创建一个文件流

FileStream fileStream = File.Create(Application.dataPath + "/StreamingFile" + "/byBin.txt");

//用二进制格式化方法来序列化save对象

bf.Serialize(fileStream, save);

fileStream.Close();

if (File.Exists(Application.dataPath + "/StreamingFile" + "/byBin.txt"))

{

UIManager.\_instance.ShowMessage("保存成功");

}

}



## （2）读取

//反序列化过程

//创建一个二进制格式化程序

BinaryFormatter bf = new BinaryFormatter();

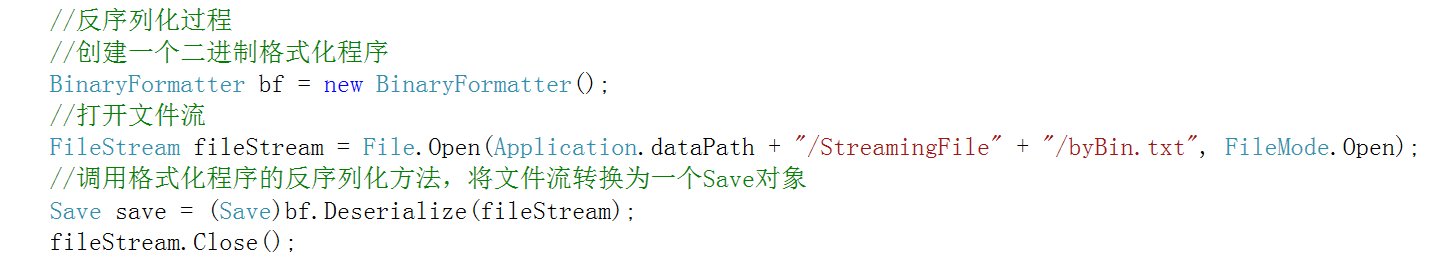
//打开文件流

FileStream fileStream = File.Open(Application.dataPath + "/StreamingFile" + "/byBin.txt", FileMode.Open);

//调用格式化程序的反序列化方法，将文件流转换为一个Save对象

Save save = (Save)bf.Deserialize(fileStream);

fileStream.Close();



# 8，XML文件的存储和读取

## 存储

private void SaveByXml()

{

Save save = CreateSaveGO();

string filePath = Application.dataPath + "/StreamingFile" + "/byXml.txt";

//创建XML文档

XmlDocument xmlDoc = new XmlDocument();

//创建根节点

XmlElement root = xmlDoc.CreateElement("save");

//设置根节点中的值

root.SetAttribute("name", "saveFile1");

//创建XmlElement

XmlElement target;

XmlElement targetPosition;

XmlElement monsterType;

//遍历save，将数据转换成XML格式

for(int i = 0; i < save.livingTargetPositions.Count; i++)

{

target = xmlDoc.CreateElement("target");

targetPosition = xmlDoc.CreateElement("targetPosition");

//设置InnerText值

targetPosition.InnerText = save.livingTargetPositions[i].ToString();

monsterType = xmlDoc.CreateElement("monsterType");

monsterType.InnerText = save.livingMonsterTypes[i].ToString();

//设置节点间的层级关系 root--target--(targetPosition,monsterType)

target.AppendChild(targetPosition);

target.AppendChild(monsterType);

root.AppendChild(target);

}

XmlElement shootNum = xmlDoc.CreateElement("shootNum");

shootNum.InnerText = save.shootNum.ToString();

root.AppendChild(shootNum);

XmlElement score = xmlDoc.CreateElement("score");

score.InnerText = save.score.ToString();

root.AppendChild(score);

xmlDoc.AppendChild(root);

xmlDoc.Save(filePath);

if (File.Exists(filePath))

{

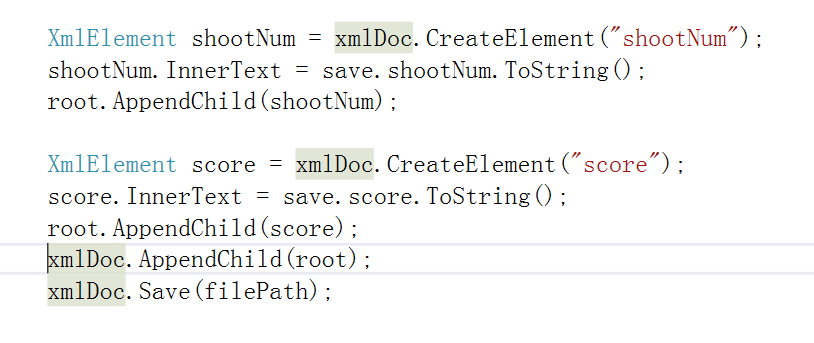
UIManager.\_instance.ShowMessage("保存成功");

}

}







## （2）读取

private void LoadByXml()

{

string filePath = Application.dataPath + "/StreamingFile" + "/byXml.txt";

if (File.Exists(filePath))

{

Save save = new Save();

//加载XML文档

XmlDocument xmlDoc = new XmlDocument();

xmlDoc.Load(filePath);

//通过节点名称来获取元素，结果为XmlNodeList类型

XmlNodeList targets = xmlDoc.GetElementsByTagName("target");

if (targets.Count != 0)

{

//遍历target节点，并获得子节点和子节点的InnerText

foreach(XmlNode target in targets)

{

XmlNode targetPosition = target.ChildNodes[0];

int targetPositionIndex = int.Parse(targetPosition.InnerText);

//把得到的值存储到save中

save.livingTargetPositions.Add(targetPositionIndex);

XmlNode monsterType = target.ChildNodes[1];

int monsterTypeIndex = int.Parse(monsterType.InnerText);

//把得到的值存储到save中

save.livingMonsterTypes.Add(monsterTypeIndex);

}

}

XmlNodeList shootNumNode = xmlDoc.GetElementsByTagName("shootNum");

int shootNum = int.Parse(shootNumNode[0].InnerText);

save.shootNum = shootNum;

XmlNodeList scoreNode = xmlDoc.GetElementsByTagName("score");

int score = int.Parse(scoreNode[0].InnerText);

save.shootNum = score;

SetGame(save);

}

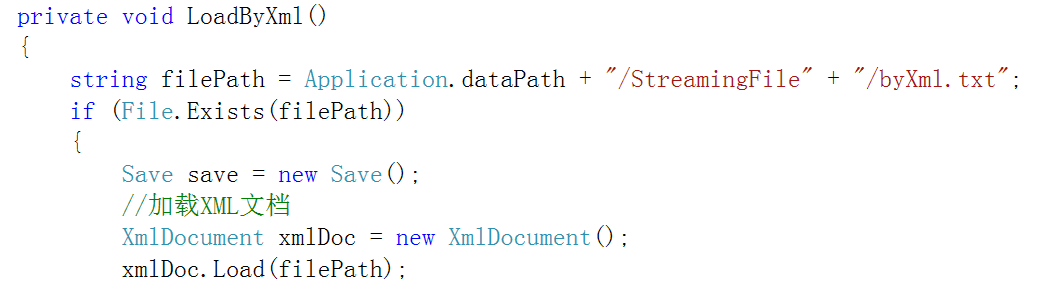
else

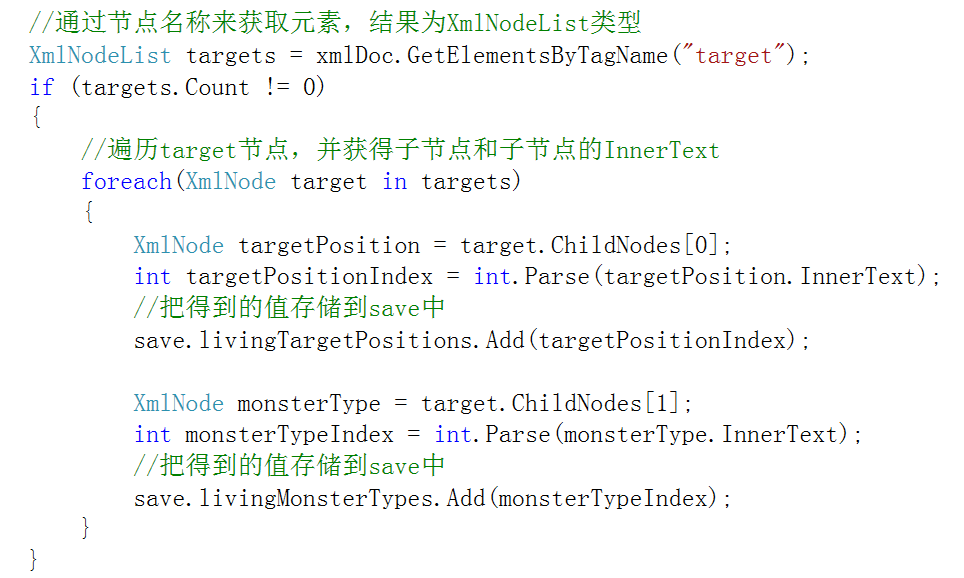
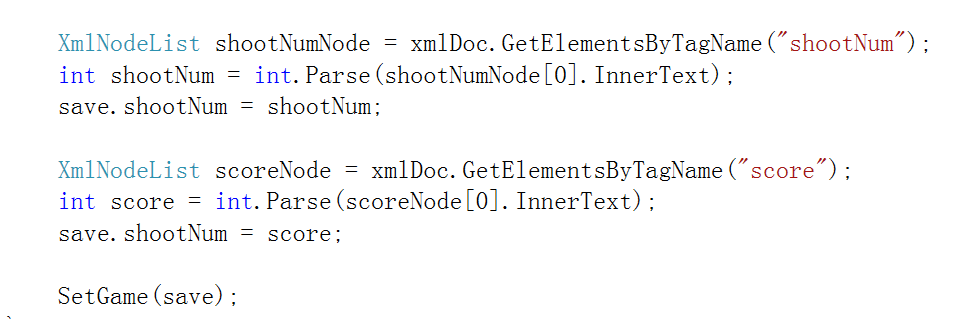
{

UIManager.\_instance.ShowMessage("加载失败");

}

}



# 9，Json文件存储和读取

## （1）存储

private void SaveByJson()

{

Save save = CreateSaveGO();

string filePath = Application.dataPath + "/StreamingFile" + "/byJson.json";

//利用JsonMapper将save对象转换为Json格式的字符串

string saveJsonStr = JsonMapper.ToJson(save);

//创建一个Streamwriter，并将字符串写入文件中

StreamWriter sw = new StreamWriter(filePath);

sw.Write(saveJsonStr);

sw.Close();

UIManager.\_instance.ShowMessage("保存成功");

}



## （2）读取

string filePath = Application.dataPath + "/StreamingFile" + "/byJson.json";

if (File.Exists(filePath))

{

//创建一个StreamReader，用来读取流

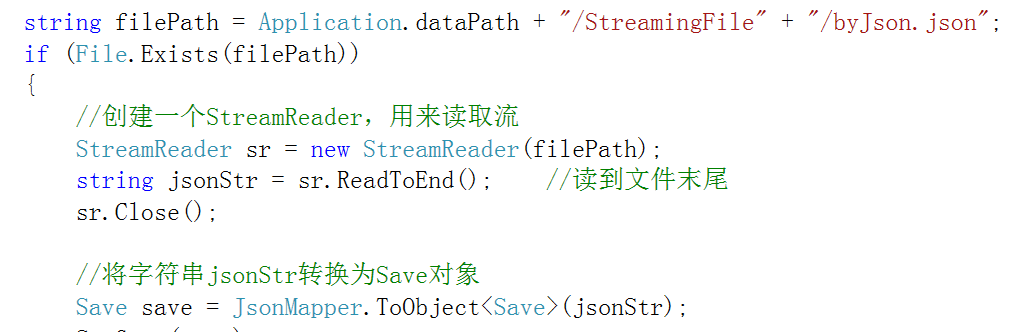
StreamReader sr = new StreamReader(filePath);

string jsonStr = sr.ReadToEnd(); //读到文件末尾

sr.Close();

//将字符串jsonStr转换为Save对象

Save save = JsonMapper.ToObject<Save>(jsonStr);



# 10，用鼠标控制手枪的旋转

float xPosPrecent = Input.mousePosition.x / Screen.width;

float yPosPrecent = Input.mousePosition.y / Screen.height;

float xAngle = -Mathf.Clamp(yPosPrecent \* maxXRotation, minXRotation, maxXRotation) + 15;

float yAngle = Mathf.Clamp(xPosPrecent \* maxYRotation, minYRotation, maxYRotation) - 60;

transform.eulerAngles = new Vector3(xAngle, yAngle, 0);