**写文章**

草稿已保存

邀请预览

发布





**今天开始我们来做一个背包系统,由于内容较多,会分成多期来做,这一期主要是从配置Json数据,到读取进游戏中这样一个过程,现在就开始吧:**

**主要功能:** 信息配置

**需要工具:** LitJson插件

https://pic4.zhimg.com/v2-1381dc30593510bce9bd047fda5deb1d_b.jpg

**基本步骤:**

1.场景搭建

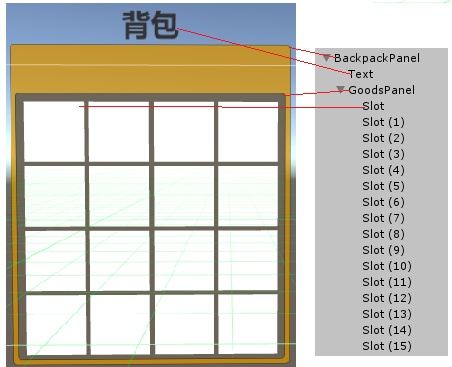
2.配置json数据

3.json数据读取

4.根据json数据创建装备物品

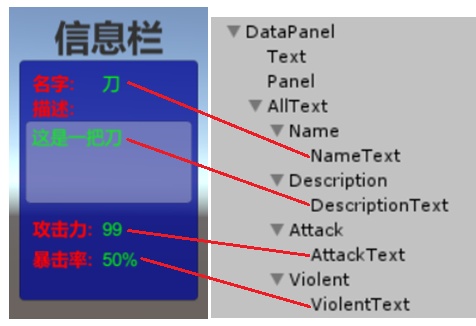
**一. 场景搭建**

1. 搭建一个背包栏:



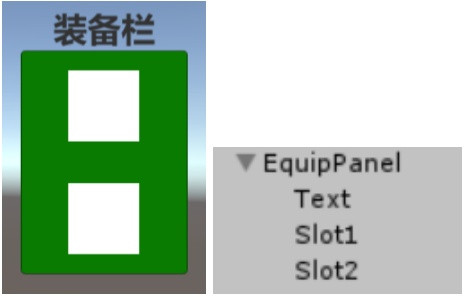
白色方格就是用来存放物品的物品槽

2. 搭建一个信息栏:



绿色字体是用来显示物品实际信息,过后将通过代码修改

3. 搭建装备栏:

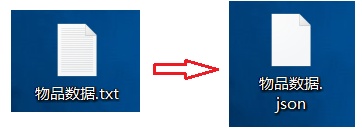


白色方格是用来装备从背包栏拖进来的物品的装备槽

**二. 配置json数据**

场景搭建好了我们现在可以配置json数据了。

1. 创建一个文本文件,将后缀名改为json:



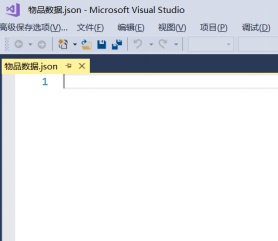
修改后会出现提示信息,确定即可,json文件就创建好了

2. 将该文件拖入Unity的Project面板中,创建一个文件夹来存放:

https://pic2.zhimg.com/v2-9d515c8bf90e5f10119de6ea6fb48f56_b.jpg

注意如果最后要发布PC平台,文件夹名要为:StreamingAssets,否则发布后会出错,发布其它平台又有不同,这里不做更多描述.

3. 双击该文件,用vs打开,就可以进行配置了:



也可以直接在本地记事本文件中配置好,再拖入Unity

4. 先试着配置一些数据,一定要按照格式来写:

[

{

"ID": 1001,

"Name": "匕首",

"Description": "杀伤力较小,但方便携带",

"Attack": 10,

"Violent": 0.05,

"ImageName": "BiShou"

},

{

"ID": 1002,

"Name": "斧头",

"Description": "破坏力惊人,无奈重量较重,一般人无法使用",

"Attack": 80,

"Violent": 0.3,

"ImageName": "FuTou"

},

{

"ID": 1003,

"Name": "弯刀",

"Description": "操作灵活,攻守兼备,且便于携带",

"Attack": 30,

"Violent": 0.2,

"ImageName": "WanDao"

}

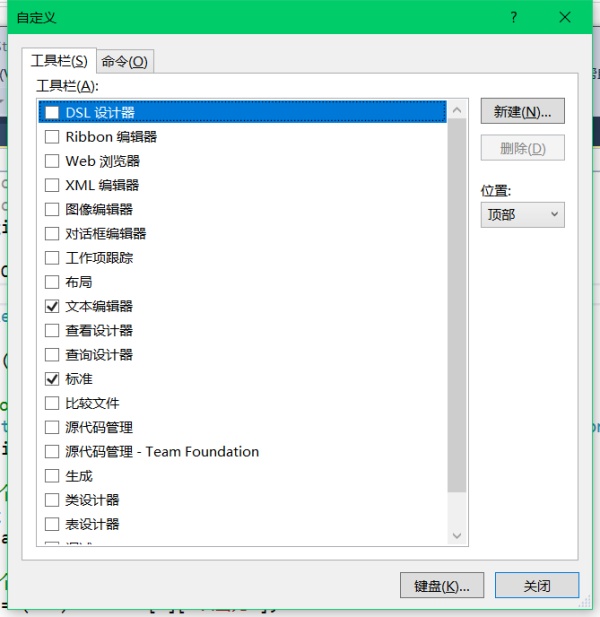
]

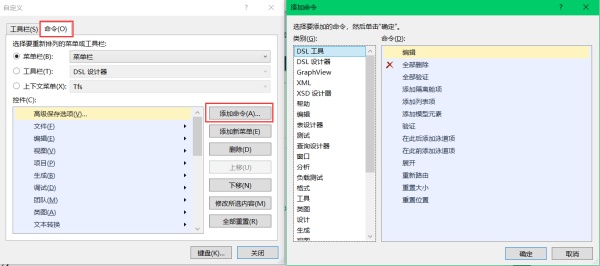
这里的每一个大括号{}代表一个数据,每一个数据里用逗号将数据进行分组,每一组数据是用键值对的形式进行存储。

当数据配置完成后,我们需要使用高级保存将数据保存成UTF8格式,这样当我们在获取中文数据时不会出现乱码,具体步骤如下:

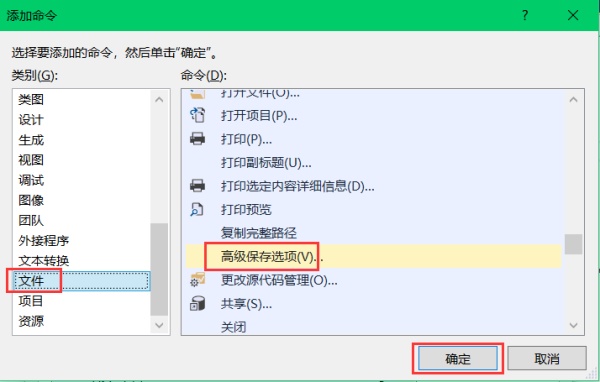


(1)视图-工具栏-自定义,点击后出现自定义面板



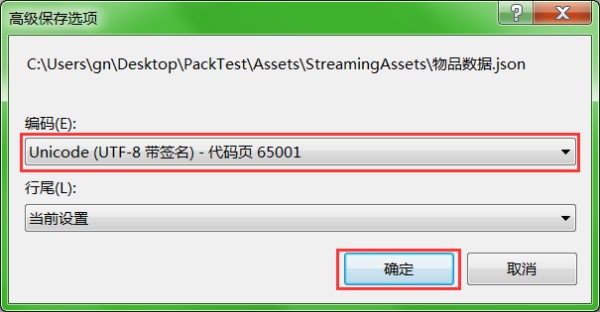


(2)命令-添加命令,点击后出现添加命令面板



(3)文件-高级保存选项-确定,点击后vs菜单面板会出现高级保存选项栏

https://pic3.zhimg.com/v2-c5fdd5b67b40c3ea35aa88ad2f83ace5_b.jpg



(4)点开高级保存选项面板,设置需要保存的编码格式,确定就可以了:

**三.json数据读取**

数据配置好后,需要把它获取到我们的游戏中,那首先需要把它们读取出来,存储在一个地方,然后要用的时候,去那个地方拿。

1. 首先把LitJson.dll插件导入unity中:

https://pic3.zhimg.com/v2-b3aa8f3d47f85026501e8db2da58bba5_b.jpg

这样我们才能对json文件进行操作

2.创建一个脚本用来读取存储json数据:

using LitJson; //导入LitJson.dll插件才可以引用

using System.IO;

public class ItemData : MonoBehaviour

{

JsonData jsonData; //引用了LitJson才能使用这个类

public void readJson()

{

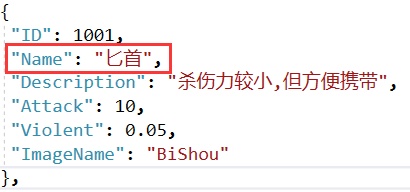
//读取本地Json文件内容,储存进jsonData中

jsonData = JsonMapper.ToObject(File.ReadAllText(Application.dataPath + "/StreamingAssets/物品数据.json"));

}

}

3.将数据存储好后该怎样获取呢,我们来随便拿取一个数据做个测试:



我们试着获取匕首

4.在脚本下添加几行测试代码:

private void Start()

{

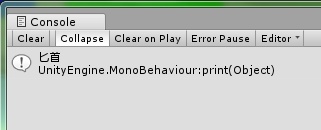
//测试

readJson();

print(jsonData[0]["Name"]); //打印该键对应的数据

}

5.挂上脚本后运行:



成功获取数据

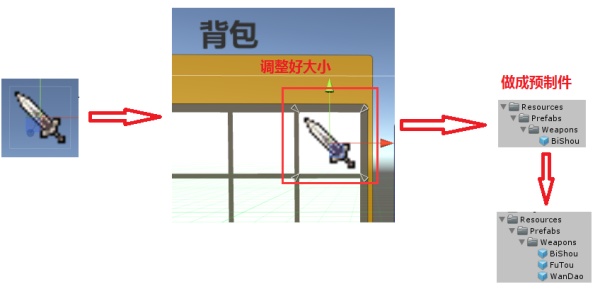
**四.根据json数据创建装备物品**

1.我们先导入武器的精灵图片:

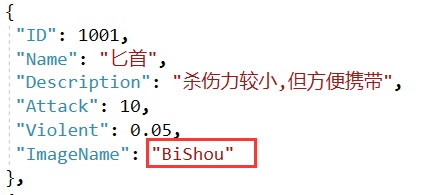
https://pic3.zhimg.com/v2-41631195c6748d6cb00c49e40071b209_b.jpg

图片为png格式,导入后转成2D精灵

2.在场景中创建Image,将武器精灵挂上去,调整好大小,做成预制件,并取好名字:



名字和配置的数据一样,这样就可以获取该数据信息加载创建相应精灵图片的实例



假如现在我们想创建一个匕首的装备,就可以找jsonData中ID为1001的数据,然后获取Imagename对应的数据,根据名字创建相应图片

3.接下来创建一个数据存储类,储存单个的数据:

public class Item : MonoBehaviour

{

public int iD { get; set; } //ID

public string name { get; set; } //名字

public string description { get; set; } //描述

public int attack { get; set; } //攻击力

public float violent { get; set; } //暴击率

public string imageName { get; set; } //图片名字

//构造函数

public Item(int \_iD, string \_name, string \_description, int \_attack, float \_violent, string \_imageName)

{

iD = \_iD;

name = \_name;

description = \_description;

attack = \_attack;

violent = \_violent;

imageName = \_imageName;

}

}

4.然后在ItemData脚本中声明一个集合,将Json数据存储到集合当中:

List<Item> itemList = new List<Item>(); //声明一个集合来保存所有json数据

void SaveItemData()

{

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

{

//使用遍历将所有数据放进数组中(代码太长了,一段分成三段)

itemList.Add(new Item((int)jsonData[i]["ID"], jsonData[i]["Name"].ToString(),

jsonData[i]["Description"].ToString(), (int)jsonData[i]["Attack"],

float.Parse(jsonData[i]["Violent"].ToString()), jsonData[i]["ImageName"].ToString()));

}

}

6.现在就是怎么拿的问题,所以创建一个根据ID拿取相应数据的方法,在其他脚本调用:

public Item AccordingToIDGetData(int \_iD) //根据ID拿取数据

{

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

{

if (\_iD == itemList[i].iD) //如果有这个ID

{

return itemList[i]; //返回该ID的数据

}

}

return null; //没有则返回空

}

7.将ItemData脚本随便挂一个物体上,使用Awake方法调用读取和存储的方法：

private void Awake()

{

readJson(); //读取

SaveItemData(); //存储

}

8.现在我们来在装备栏创建一个匕首图片的实例,首先创建一个脚本,它将在图片被创建时挂在图片上:

/// <summary>

/// 武器装备被创建时挂在上

/// </summary>

public class EquipmentPicture : MonoBehaviour

{

public Item selfItem; //用来存储属于自身的信息

}

9.创建CreateBackpack脚本,挂在GoodsPanel上,里面做一个根据ID创建图片的方法：

/// <summary>

/// 挂在GoodsPanel上

/// </summary>

public class CreateBackpack : MonoBehaviour

{

ItemData itemData; //声明ItemData

GameObject image; //用来接收创建的图片

private void Start()

{

//创建ItemData的实例,才可以调用它的方法

itemData = FindObjectOfType<ItemData>();

}

GameObject CreateImage(int iD) //创建图片

{

//根据ID获取数据

Item item = itemData.AccordingToIDGetData(iD);

//根据数据加载创建图片

GameObject image =Instantiate(Resources.Load("Prefabs/Weapons/" + item.imageName) as GameObject);

image.AddComponent<EquipmentPicture>().selfItem= item; //添加脚本并储存自身信息

return image; //返回图片

}

}

10.来创建一个方法,将创建的图片放入小方格中:

void CreateImageInSlot(GameObject item) //将图片放进空格里

{

for (int i = 0; i < transform.childCount; i++) //遍历所有小方格

{

if (transform.GetChild(i).childCount == 0) //如果小方格没有子物体,表示该小方格为空

{

item.transform.SetParent(transform.GetChild(i)); //就将创建出的图片放进该小方格中

item.transform.localPosition = Vector3.zero; //居中

break; //放进去后就可以退出了

}

}

}

11.在Start方法里调用:

private void Start()

{

//创建ItemData的实例,才可以调用它的方法

itemData = FindObjectOfType<ItemData>();

image = CreateImage(1001); //创建匕首图片

CreateImageInSlot(image); //将图片放进小方格中

image = CreateImage(1002); //创建斧头图片

CreateImageInSlot(image);

image = CreateImage(1003); //创建弯刀图片

CreateImageInSlot(image);

}

12.运行程序:



三个武器图片成功创建进小方格

好的,第一节的内容大概就是这些,如果能根据以上内容成功创建武器图片,可以对图片的数据进行打印测试,看看数据是否成功获取,下一节主要是图片和拖拽相关功能。

