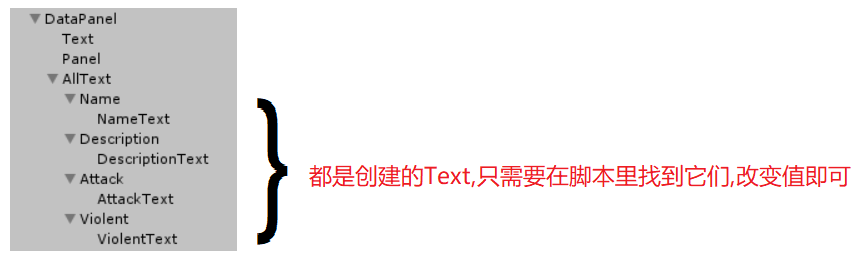
这一期我们来完成显示物品信息和装备物品的功能。

先来做物品信息，我们这里显示信息的方法是通过将鼠标光标放在物品上方，就会出现信息栏，在信息栏里显示该物品的信息，光标移出物品上方时，关闭信息栏，我们先来到物品身上的EquipmentPicture脚本里，新添加添加两个接口，并实现接口方法：

|  |
| --- |
| public class EquipmentPicture : MonoBehaviour, IBeginDragHandler, IDragHandler, IEndDragHandler,IPointerEnterHandler,IPointerExitHandler  {  GameObject dataPanel; //声明信息栏  public void OnPointerEnter(PointerEventData eventData) //光标进入时调用一次  {  //显示信息栏  if (!dataPanel) //如果dataPanel引用为空,就去获取它  //DataPanel开始时隐藏状态,一般方法找不到,所以先找到父物体,通过父物体的路径去找即可  dataPanel = GameObject.Find("Canvas").transform.Find("DataPanel").gameObject;  dataPanel.SetActive(true); //显示  }  public void OnPointerExit(PointerEventData eventData) //光标移出时调用一次  {  dataPanel.SetActive(false); //关闭信息栏  } |

然后来做一个显示信息的方法，我们先来看一下信息栏的搭建结构：



我们来做一个显示信息的方法：

|  |
| --- |
| void ShowData() //显示信息  {  //名字和描述显示当前物品的信息  GameObject.Find("NameText").GetComponent<Text>().text = selfItem.name;  GameObject.Find("DescriptionText").GetComponent<Text>().text = selfItem.description;  switch (selfItem.type) //根据物品类型显示自身独有的信息  {  case GoodsType.Weapon:  WeaponsItem weaponData = selfItem as WeaponsItem; //selfItem是Item类型,没有攻击力的字段,所以要进行转换  GameObject.Find("Attack").GetComponent<Text>().text = "攻击力:";  GameObject.Find("AttackText").GetComponent<Text>().text = weaponData.attack.ToString();  GameObject.Find("Violent").GetComponent<Text>().text = "暴击率:";  GameObject.Find("ViolentText").GetComponent<Text>().text = string.Format("{0:P0}", weaponData.violent);  break;  case GoodsType.Clothing:  ClothingItem clothingData = selfItem as ClothingItem;  GameObject.Find("Attack").GetComponent<Text>().text = "物理防御:";  GameObject.Find("AttackText").GetComponent<Text>().text = clothingData.physicalDefense.ToString();  GameObject.Find("Violent").GetComponent<Text>().text = "魔法防御:";  GameObject.Find("ViolentText").GetComponent<Text>().text = clothingData.magicDefense.ToString();  break;  case GoodsType.Prop:  PropItem propData = selfItem as PropItem;  GameObject.Find("Attack").GetComponent<Text>().text = "血量回复:";  GameObject.Find("AttackText").GetComponent<Text>().text = propData.addHP.ToString();  GameObject.Find("Violent").GetComponent<Text>().text = "法力回复:";  GameObject.Find("ViolentText").GetComponent<Text>().text = propData.addMP.ToString();  break;  }  } |

把它放到OnPointerEnter方法里调用即可：

|  |
| --- |
| public void OnPointerEnter(PointerEventData eventData) //光标进入时调用一次  {  //显示信息栏  if (!dataPanel) //如果dataPanel引用为空,就去获取它  //DataPanel开始时隐藏状态,一般方法找不到,所以先找到父物体,通过父物体的路径去找即可  dataPanel = GameObject.Find("Canvas").transform.Find("DataPanel").gameObject;  dataPanel.SetActive(true); //显示  //显示信息  ShowData();  } |

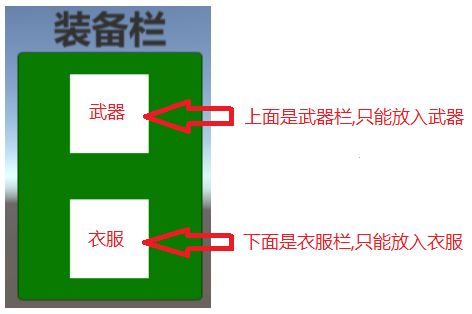
如果觉得ShowData方法内容太长，可进行优化，再创建一个方法ShowSelfData将重复代码封装起来显示各类物品特有信息：

|  |
| --- |
| void ShowData() //显示信息  {  //名字和描述显示当前物品的信息  GameObject.Find("NameText").GetComponent<Text>().text = selfItem.name;  GameObject.Find("DescriptionText").GetComponent<Text>().text = selfItem.description;  switch (selfItem.type) //根据物品类型显示自身独有的信息  {  case GoodsType.Weapon:  WeaponsItem weaponData = selfItem as WeaponsItem; //selfItem是Item类型,没有攻击力的字段,所以要进行转换  ShowSelfData("攻击力:", weaponData.attack.ToString(), "暴击率:", string.Format("{0:P0}", weaponData.violent));  break;  case GoodsType.Clothing:  ClothingItem clothingData = selfItem as ClothingItem;  ShowSelfData("物理防御:", clothingData.physicalDefense.ToString(), "魔法防御:", clothingData.magicDefense.ToString());  break;  case GoodsType.Prop:  PropItem propData = selfItem as PropItem;  ShowSelfData("血量回复:", propData.addHP.ToString(), "法力回复:", propData.addMP.ToString());  break;  }  }  void ShowSelfData(string t1,string t2,string t3,string t4) //显示特有信息  {  GameObject.Find("Attack").GetComponent<Text>().text = t1;  GameObject.Find("AttackText").GetComponent<Text>().text = t2;  GameObject.Find("Violent").GetComponent<Text>().text = t3;  GameObject.Find("ViolentText").GetComponent<Text>().text = t4;  } |

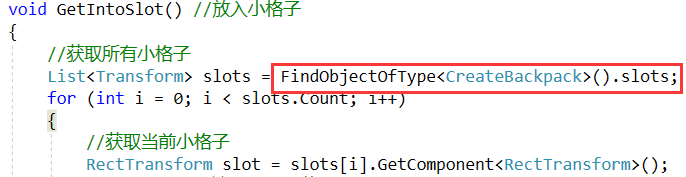
运行程序，将鼠标放在物品上，就可以弹出信息栏显示当前物品的信息了：



显示信息的功能成功了，我们来做将物品装备到装备栏的功能，我们先来分析下装备栏的特性：



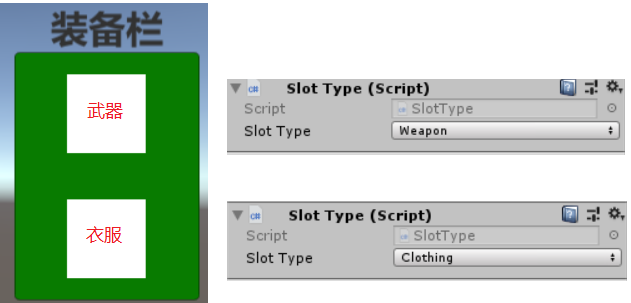
我们在结束拖拽图片的方法里，获取了背包栏下所有小格子，但是并没有获取装备栏里的两个：



所以没法将图片拖到装备栏的，所以必须要将装备栏的小格里也一起获取才行，然后对格子的类型进行判断，将物品放入应该放入的格子中，我们创建一个脚本，挂在准备栏的两个格子上，为它们设定类型：

|  |
| --- |
| public class SlotType : MonoBehaviour  {  public GoodsType slotType;  } |

挂到装备栏格子上后，在编辑器里设置好相应类型：



将所有格子(包括背包栏的和装备栏的)的标签都设为”Slot”:



然后重构EquipmentPicture里的GetIntoSlot方法:

|  |
| --- |
| void GetIntoSlot() //放入小格子  {  GameObject[] slots = GameObject.FindGameObjectsWithTag("Slot"); //根据标签获取所有小格子  RectTransform slot = null;  bool GetInto = false; //判断是否可以进入小格子  foreach (var item in slots)  {  slot = item.GetComponent<RectTransform>(); //获取该小格子的RectTransform组件  if (slot.rect.Contains(Input.mousePosition - slot.position)) //判断鼠标是否进入该小格子  {  if (slot.GetComponent<SlotType>()) //判断小格子是不是装备栏的  {  if (selfItem.type == slot.GetComponent<SlotType>().slotType)//判断物品类型和格子类型是否相同  GetInto = true;  }  else //否则就是物品栏的  GetInto = true;  break;  }  }  if (GetInto) //如果确定能进入小格子  {  if (slot.transform.childCount > 0)//判断格子里面是否已有物品  {  Transform oriP = slot.transform.GetChild(0); //获取原物品  oriP.SetParent(original); //将原物品放进拖拽图片的原格子  oriP.localPosition = Vector3.zero; //居中  }  transform.SetParent(slot); //物品进入小格子并居中  transform.localPosition = Vector3.zero;  return; //退出方法  }  transform.SetParent(original); //返回原位置  transform.localPosition = Vector3.zero; //居中  } |

运行程序:



好的,这一期的功能就全部完成,本系列作为新手入门教程,功能就做到这里,当然实际游戏中背包系统要复杂的多,如果有机会还会继续更,谢谢观看。