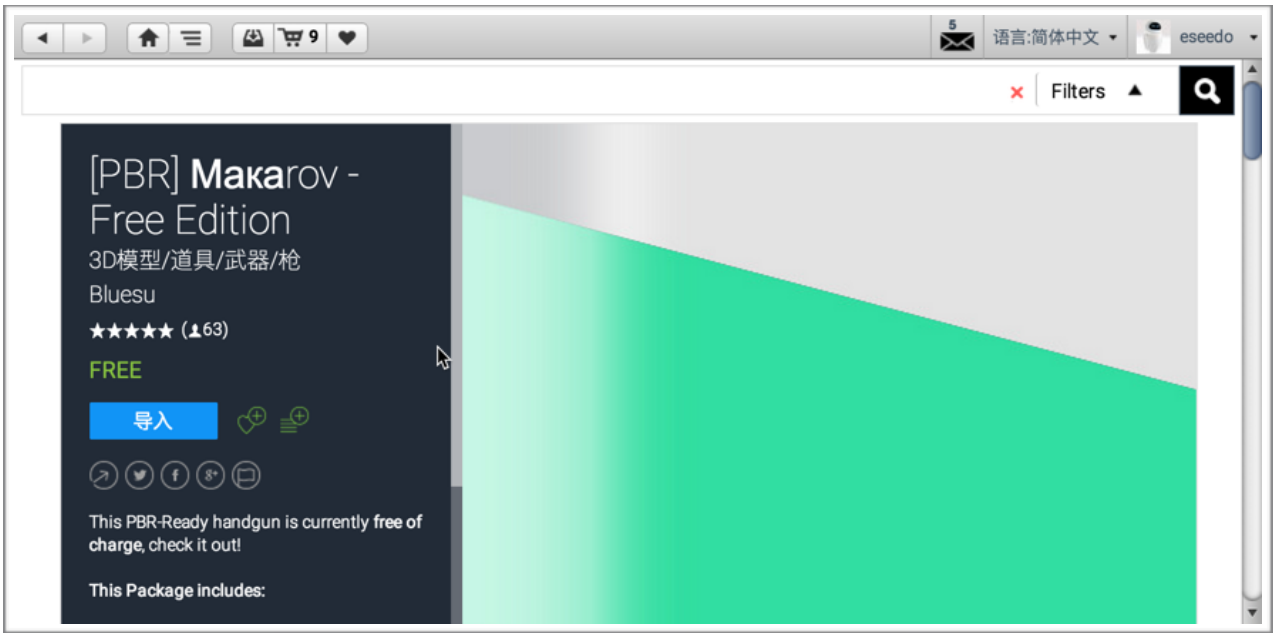
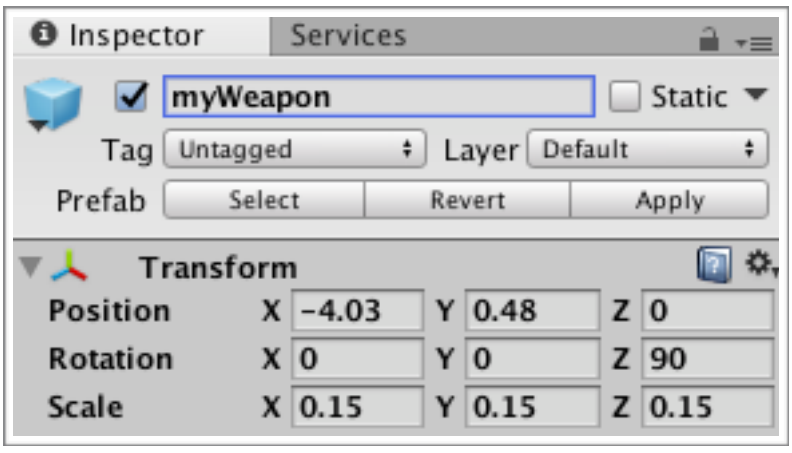
在本课的内容中，我们将让玩家可以在地面上捡起武器，然后开火。而不是一开始就已经一人一枪横扫千军了。

为此，首先我们要从Asset Store中下载一个武器资源。

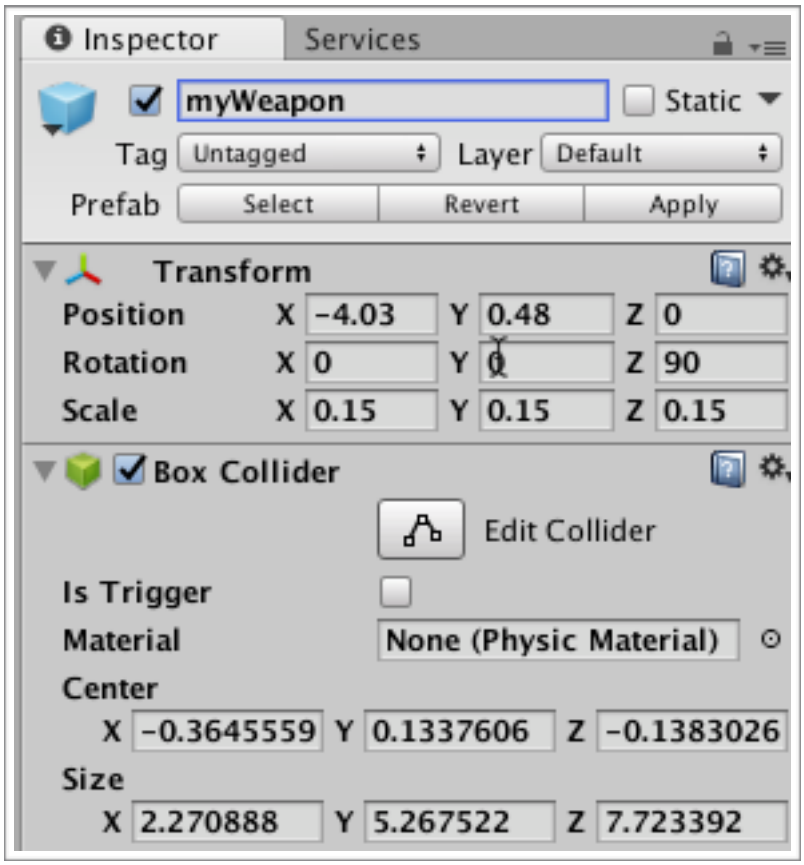
在Asset Store中搜索pistol，然后选择FREE ONLY，从搜索结果中选择下图中的这个资源。



把下载后的资源拖入到Arts文件夹。打开[PBR]Makarov文件夹，然后将其中的预设体拖动到Hierarchy视图中，使其成为ruined_house的子对象，并更名为myWeapon。接下来在Inspector视图中调整Transform中的相关属性如下（仅供参考）：



接下来点击Inspector视图中的Add Component，给pistol对象添加一个Box Collider组件，然后设置碰撞体的大小，如图所示。
再次强调，这些数字是相对比较主观的，你觉得合适就行。



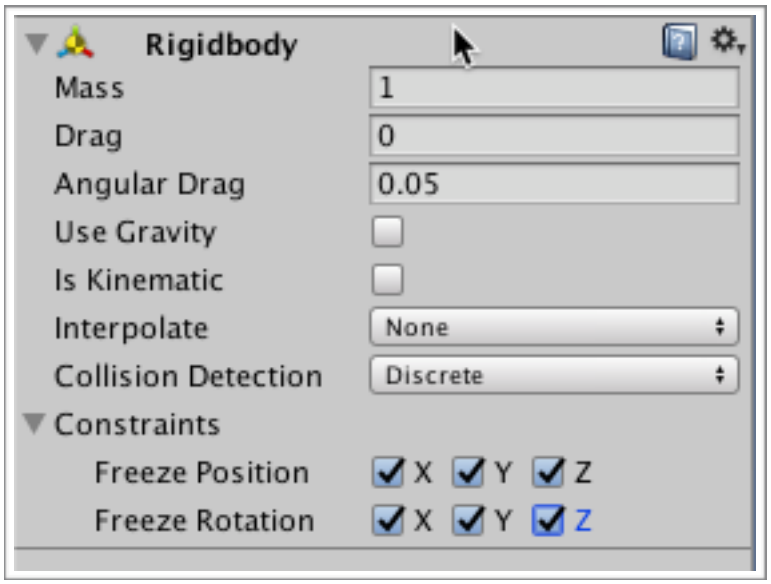
除了Box Collider，我们还需要继续给pickupWeapon对象添加一个Rigidbody组件，因为我们希望武器受到重力的影响。这样当游戏开始的时候，武器就会自动掉落到地面。

好了，接下来我们需要创建脚本，来处理拾取武器的操作。
在Hierarhcy视图中选中CameraParent—Main Camera，然后点击Inspector视图中的Add Component按钮，给它添加一个新的脚本组件，并将其命名为PickupWeapon。
在MonoDevelop中将其打开并更改其中的代码如下：

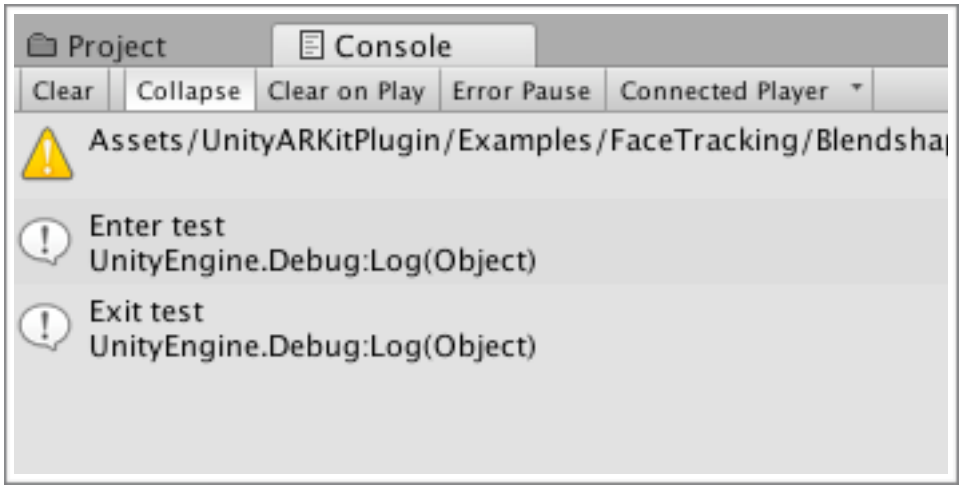
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PickupWeapon : MonoBehaviour {

	// Use this for initialization
	void Start () {

	}

	//1.开始碰撞

	void OnCollisionEnter(Collision col){

		if (col.gameObject.name == "myWeapon") {

			Debug.Log ("Enter test");

		}
	}

	//2.碰撞结束

	void OnCollisionExit(Collision col){

		if(col.gameObject.name == "myWeapon"{

			Debug.Log("Exit test");
		}

	}

}
```

这里我们只是添加了两个方法，而这两个方法在之前的课程中有提过，分别是用于处理碰撞开始和碰撞结束的事件。这里我们暂时只是在Console中输出相应的信息。

好了，接下来回到Unity编辑器，在Hierarchy视图中选中CameraParent-Main Camera，然后在Inspector视图中点击Add Component，添加一个Rigidbody组件，并取消勾选Use Gravity。此外在Rigidbody组件的Constraints属性处勾选所有的选项。

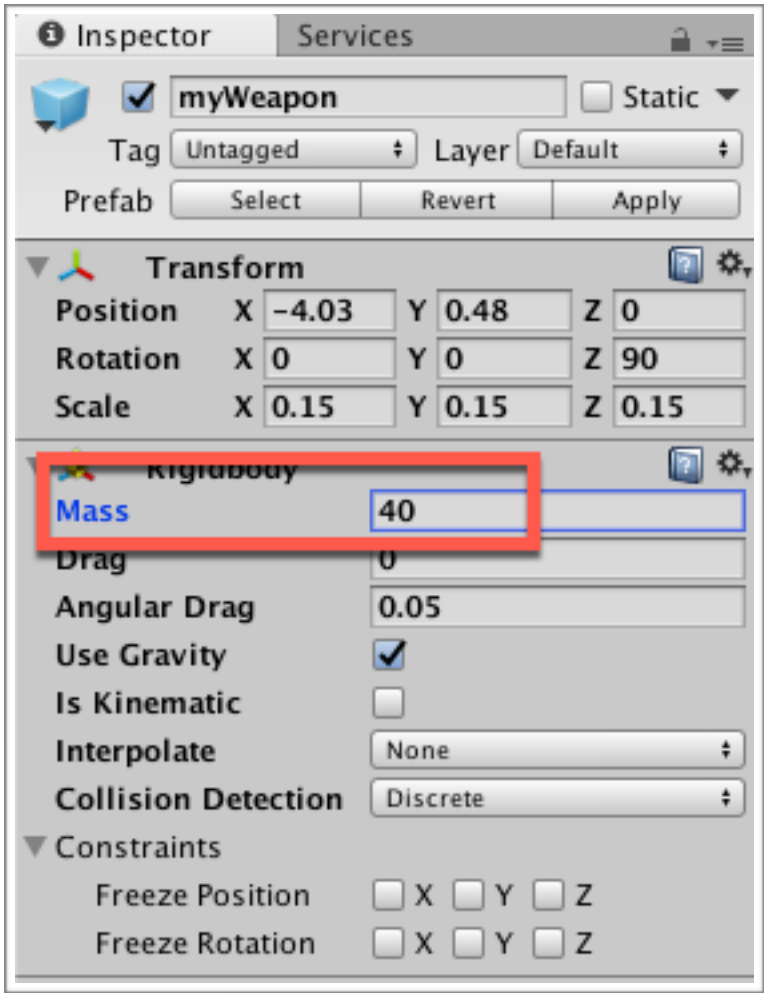


接下来可以测试一下，点击Unity编辑器上的Play按钮进行测试。
当我们在场景中拖动主摄像机到地面上的武器位置附近时，可以看到武器被撞飞了，同时在Console中输出了相应的提示。



不过看起来武器的重量太轻了点，让我们给它增加点重量。

在Hierarchy视图中选中HitCubeParent-ruined_house-myWeapon，然后在Inspector视图中将其Rigidbody组件的Mass属性设置为40.

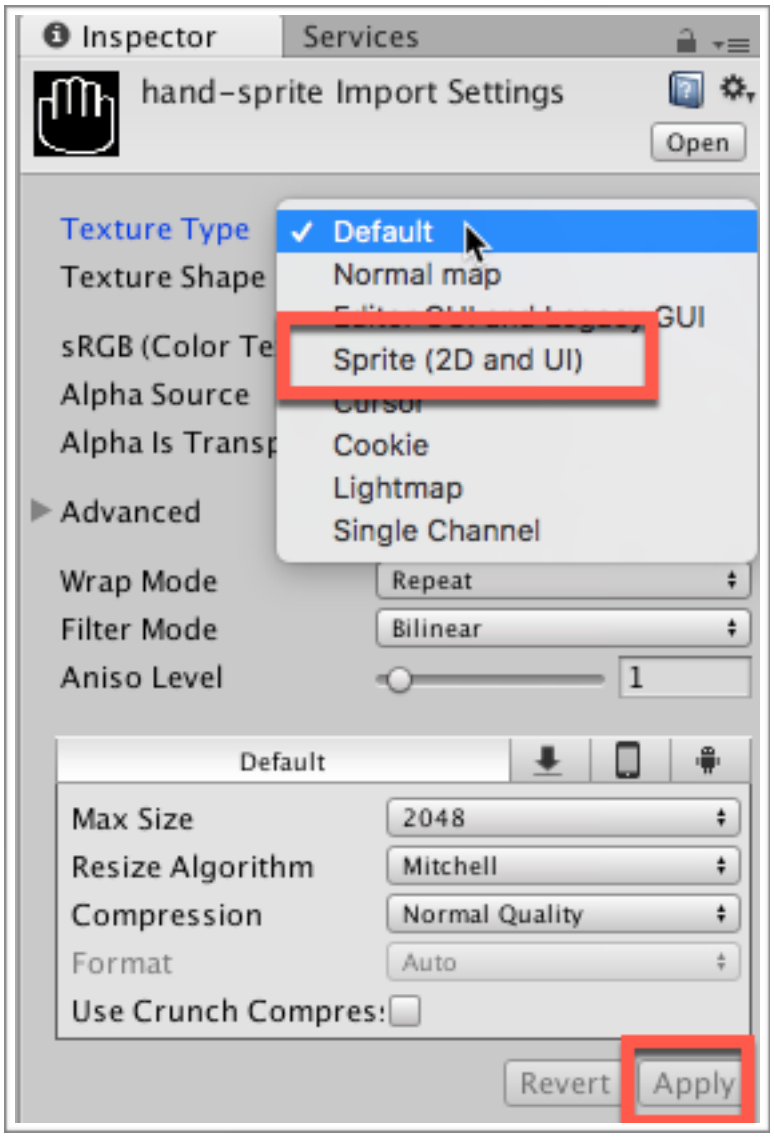


再次运行测试，看起来差不多了~

然后从这里下载本章所需的资源素材：
链接:https://pan.baidu.com/s/1pLKGvhT 密码:s6pz

将其中的文件解压缩，并将hand-sprite.png资源文件拖动到Unity编辑器的Project视图的Arts文件夹中。
选中该文件，然后在Inspector视图中点击Texture Type旁的下拉列表，然后选择Sprite(2D and UI)

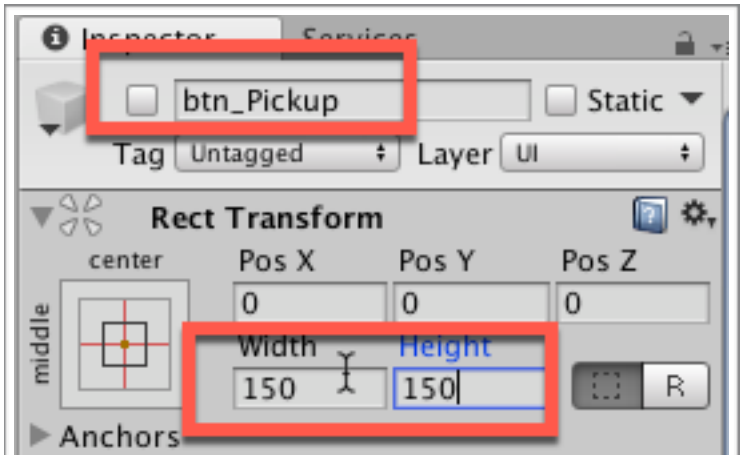
然后别忘了点击右下角的Apply按钮



接下来在Hierarchy视图中选中Canvas，添加一个新的Button UI元素，将其命名为btn_Pickup，删除该按钮的文本子对象。

然后在Inspector视图中将Image组建下的Source Image属性设置为hand-sprite。

更改Rect Transform属性中的Width 和Height为150和150.
最后在默认情况下对其禁用，如图所示。



好了，现在可以回到我们的PickupWeapon.cs脚本，修改其中的代码如下：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PickupWeapon : MonoBehaviour {

	//1.创建到拾取武器按钮的引用
	public GameObject pickupBtn;
	// Use this for initialization
	void Start () {

	}

	//1.开始碰撞

	void OnCollisionEnter(Collision col){

		if (col.gameObject.name == "myWeapon") {

//			Debug.Log ("Enter test");

			//2.启用拾取武器按钮
			pickupBtn.gameObject.SetActive (true);

		}
	}

	//2.碰撞结束

	void OnCollisionExit(Collision col){

		if(col.gameObject.name == "myWeapon"){

//			Debug.Log("Exit test");
			//3.禁用拾取武器按钮
			pickupBtn.gameObject.SetActive (false);
		}

	}

}
```

在以上代码中，按照注释行的数字编号来简单解释一下：
1.创建了到拾取武器按钮的引用。
2.当碰撞发生时，启用拾取武器按钮
3.当碰撞结束时，禁用拾取武器按钮。

回到Unity编辑器，在Hierarchy视图中选中CameraParent-Main Camera，然后在Inspector视图中将Pickup Weapon组件的Pickup Btn属性更改为btn_Pickup，如图所示。

好了，接下来我们完成一个之前别遗忘的工作，给准星创建一个引用。

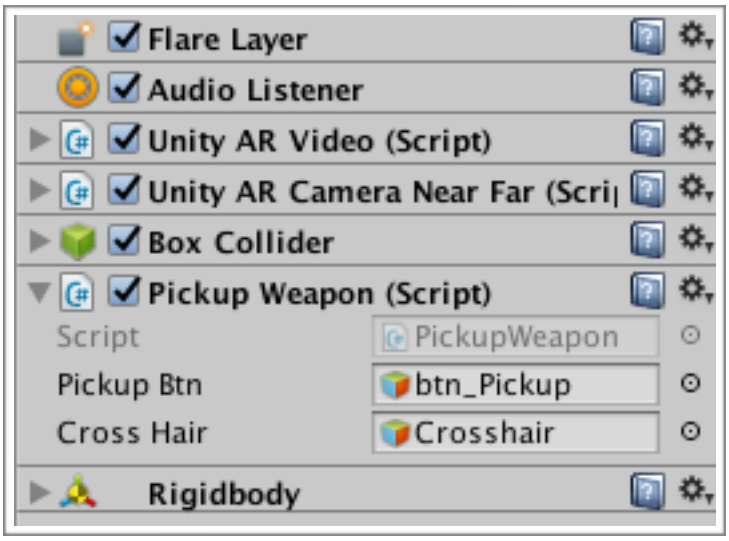
回到PickupWeapon.cs，更改代码如下：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PickupWeapon : MonoBehaviour {

    //创建到拾取武器按钮的引用
    public GameObject pickupBtn;

    //1.创建到准星的引用
    public GameObject crossHair;


    // Use this for initialization
    void Start () {
```

```
    }

    //开始碰撞

    void OnCollisionEnter(Collision col){

        if (col.gameObject.name == "myWeapon") {

//            Debug.Log ("Enter test");

            //启用拾取武器按钮
            pickupBtn.gameObject.SetActive (true);

            //2.禁用准星
            crossHair.gameObject.SetActive(false);

        }
    }

    //碰撞结束

    void OnCollisionExit(Collision col){

        if(col.gameObject.name == "myWeapon"){

//            Debug.Log("Exit test");
            //禁用拾取武器按钮
            pickupBtn.gameObject.SetActive (false);

            //3.启用准星
            crossHair.gameObject.SetActive(true);
        }

    }

}
```

按照注释行数字编号简单解释下：
1.创建了到准星UI控件对象的引用
2.当碰撞发生时，禁用准星
3.当碰撞结束时，启用准星

然后回到Unity编辑器，在Hierarchy视图中选中CameraParent-Main Camera，然后在Inspector视图中将Pickup Weapon组件的Cross Hair属性设置为Crosshiar UI元素，如图所示。



接下来点击工具栏上的Play按钮，预览下游戏效果。

到了这一步的，基本的操作已经完成了，让我们再做一些完善工作。

在Hierarchy视图中选中Canvas，然后在Inspector视图中点击Add Component，给其添加一个新的脚本组件，将其命名为WeaponPickup。在MonoDevelop中将其打开，并更改其中的代码如下：

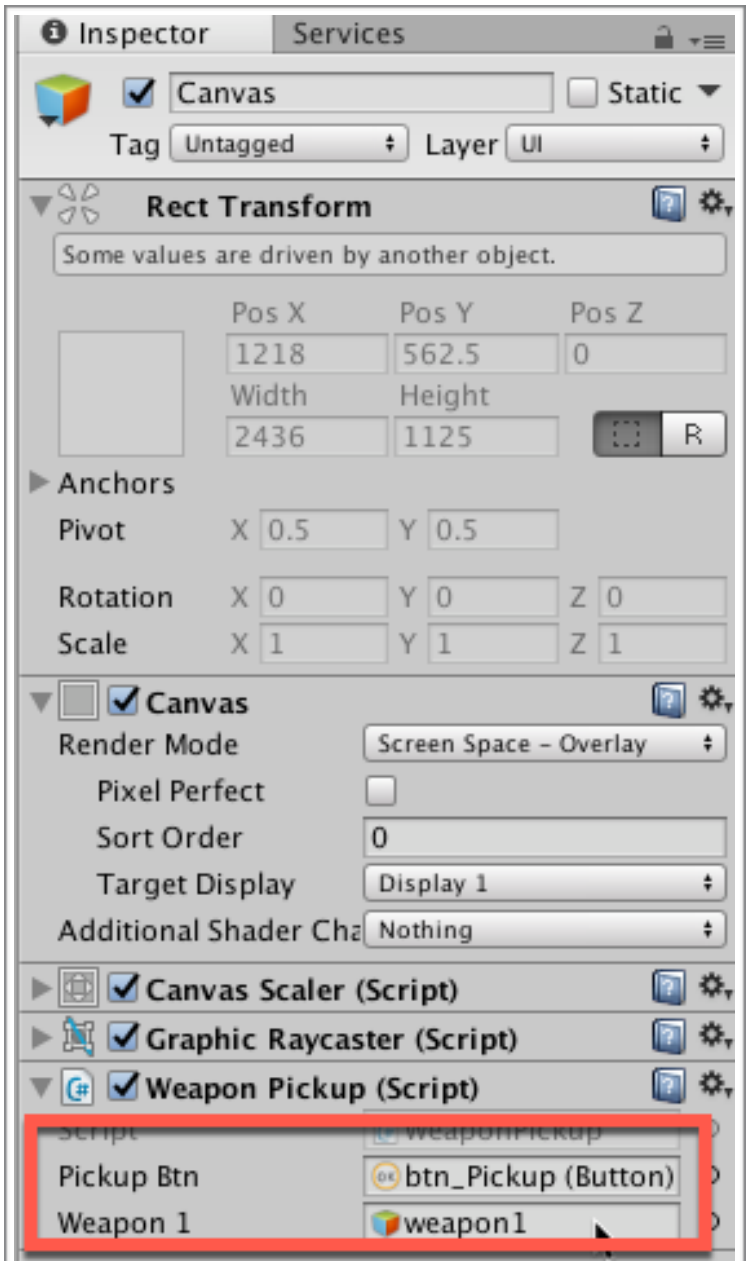
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
//1.导入UI相关的命名空间
using UnityEngine.UI;


public class WeaponPickup : MonoBehaviour {

    //2.创建到拾取武器按钮的引用
    public Button pickupBtn;

    //3.创建到武器的引用
    public GameObject weapon1;

    // Use this for initialization
```

```
        void Start () {

        }

        // Update is called once per frame
        void Update () {

        }
}
```

以上代码比较简单，这里就不再赘述了，大家直接看注释应该就可以明白。

回到Unity编辑器，首先在Hierarchy视图中找到CameraParent-Main Camera-weapon1，然后在默认状态下将其禁用。

然后在Hierarchy视图中选中Canvas，设置Weapon Pick Up组件的属性如图：



接下来回到WeaponPickup.cs脚本，更改其中的代码如下：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
//导入UI相关的命名空间
using UnityEngine.UI;


public class WeaponPickup : MonoBehaviour {

	//创建到拾取武器按钮的引用
	public Button pickupBtn;

	//创建到武器的引用
	public GameObject weapon1;

	// Use this for initialization
	void Start () {

		//1.启用武器
		pickupBtn.onClick.AddListener (EnableWeapon);

	}

	void EnableWeapon(){

		weapon1.gameObject.SetActive (true);

	}

	// Update is called once per frame
	void Update () {

	}
}
```

以上代码中，我们只是在用户按下拾取武器的按钮时启用武器。

好了，接下来可以点击工具栏上的Play按钮预览游戏效果。

本课的内容到此结束，我们下一课再见~