# [技巧| Unity中Avatar换装实现](https://blog.uwa4d.com/archives/avartar.html)

* 作者：admin
* /
* 时间：2016年07月08日
* /
* 浏览：3390 次
* /
* 分类：[厚积薄发](https://blog.uwa4d.com/category/technology/)

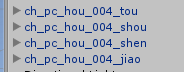
Avatar换装是MMO游戏不可缺少的一部分，一个人物模型通常可拆分为头、身体、手臂、腿、武器等部分，如何将这些部分组合到一起呢？本文将阐述如何将在Unity中实现人物模型的换装功能。

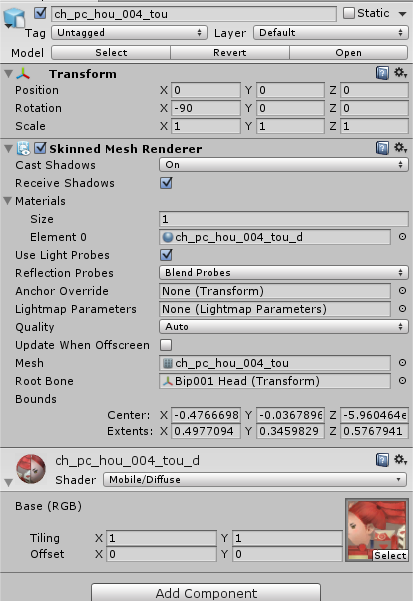
这是侑虎科技第65篇原创文章，感谢作者邹春毅（QQ：442319386）供稿。欢迎转发分享，未经作者授权请勿转载。同时如果您有任何独到的见解或者发现也欢迎联系我们，一起探讨。（QQ群465082844）

作者Github：<https://github.com/zouchunyi>

# **资源准备**

1.每一套装备模型必须使用同一套骨骼，并单独将骨骼数据保存成一个Prefab。红色部分为武器挂节点（也可以把武器做成一个SkinnedMesh，不采用挂接点的形式），骨骼数据在Unity中的展示形式就是Transform。  


2.将模型拆分成多个部分，将每一个部分单独保存成Prefab，武器也单独保存为一个Prefab。  
  


每一个Prefab都含有自身的SkinnedMeshRenderer。  


# **实现过程**

1.创建骨骼GameObject，所有装备的蒙皮数据会最终合成到这个Prefab中。

2.创建装备GameObject，用于搜集其中蒙皮数据以生成新的SkinnedMeshRenderer到骨骼Prefab中。

3.public void CombineObject(GameObject skeleton, SkinnedMeshRenderer[] meshes, bool combine = false)传入骨骼的GameObject和蒙皮数据。

4.搜集装备蒙皮数据中的有效信息。

// Collect information from meshes

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

{

SkinnedMeshRenderer smr = meshes[i];

materials.AddRange(smr.materials); // Collect materials

// Collect meshes

for (int sub = 0; sub < smr.sharedMesh.subMeshCount; sub++)

{

CombineInstance ci = new CombineInstance();

ci.mesh = smr.sharedMesh;

ci.subMeshIndex = sub;

combineInstances.Add(ci);

}

// Collect bones

for (int j = 0 ; j < smr.bones.Length; j ++)

{

int tBase = 0;

for (tBase = 0; tBase < transforms.Count; tBase ++)

{

if (smr.bones[j].name.Equals(transforms[tBase].name))

{

bones.Add(transforms[tBase]);

break;

}

}

}

}

5.为骨骼GameObject生成新的SkinnedMeshRenderer。

// Create a new SkinnedMeshRenderer

SkinnedMeshRenderer oldSKinned = skeleton.GetComponent<SkinnedMeshRenderer> ();

if (oldSKinned != null) {

GameObject.DestroyImmediate (oldSKinned);

}

SkinnedMeshRenderer r = skeleton.AddComponent<SkinnedMeshRenderer>();

r.sharedMesh = new Mesh();

r.sharedMesh.CombineMeshes(combineInstances.ToArray(), false, false);// Combine meshes

r.bones = bones.ToArray();// Use new bones

6.挂接武器。

Transform[] transforms = Instance.GetComponentsInChildren<Transform>();

foreach (Transform joint in transforms) {

if (joint.name == "weapon\_hand\_r")

{// find the joint (need the support of art designer)

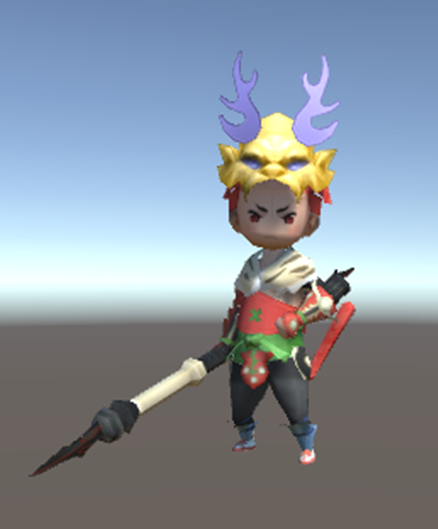
WeaponInstance.transform.parent = joint.gameObject.transform;

break;

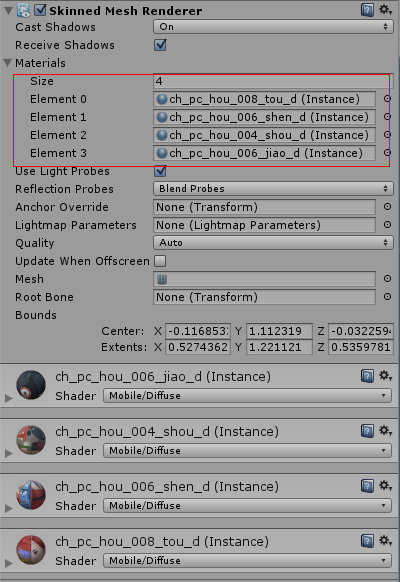
}

}

其中WeaponInstance为武器实例GameObject，Instance为骨骼实例GameObject。

**合成后的效果**  


# **如何优化**

  
合成之后的模型拥有4个独立材质，加上独立的对象武器，也就是会产生5个Draw Call；如果将在骨骼中的这4个材质合并成一个，那么就能将Draw Call减少到2个。

**其中实现过程如下：**  
优化CombineObject方法，其中Combine为bool类型，用于标识是否合并材质。

// merge materials

if (combine)

{

newMaterial = new Material (Shader.Find ("Mobile/Diffuse"));

oldUV = new List<Vector2[]>();

// merge the texture

List<Texture2D> Textures = new List<Texture2D>();

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

{

Textures.Add(materials[i].GetTexture(COMBINE\_DIFFUSE\_TEXTURE) as Texture2D);

}

newDiffuseTex = new Texture2D(COMBINE\_TEXTURE\_MAX, COMBINE\_TEXTURE\_MAX, TextureFormat.RGBA32, true);

Rect[] uvs = newDiffuseTex.PackTextures(Textures.ToArray(), 0);

newMaterial.mainTexture = newDiffuseTex;

// reset uv

Vector2[] uva, uvb;

for (int j = 0; j < combineInstances.Count; j++)

{

uva = (Vector2[])(combineInstances[j].mesh.uv);

uvb = new Vector2[uva.Length];

for (int k = 0; k < uva.Length; k++)

{

uvb[k] = new Vector2((uva[k].x \* uvs[j].width) + uvs[j].x, (uva[k].y \* uvs[j].height) + uvs[j].y);

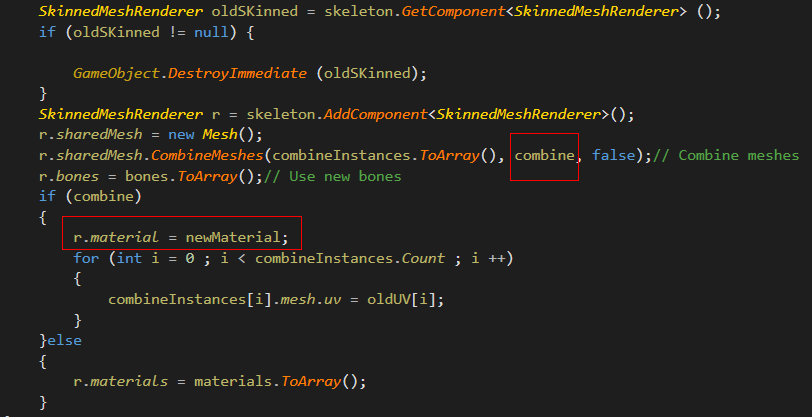
}

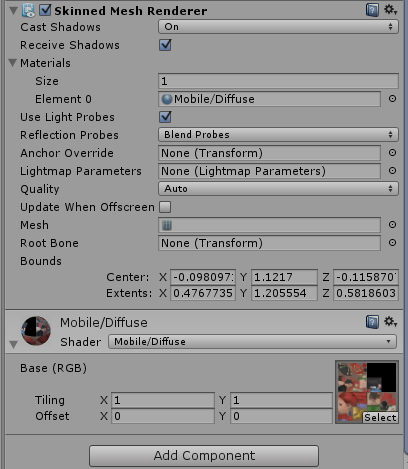
oldUV.Add(combineInstances[j].mesh.uv);

combineInstances[j].mesh.uv = uvb;

}

}

生成新的SkinnedMeshRenderer时略有区别：  


**最终效果如下：**  
  
可以看出，新的SkinnedMeshRenderer只有一个材质，Draw Call自然也就降低了。

# **示例工程**

本人已将此示例工程分享到了GitHub中：<https://github.com/zouchunyi/UnityAvater>  
感兴趣的朋友可以下载。工程中代码大家可以直接使用，但是美术资源不得用于任何商业用途。  
