OS\Compiler\OOP\Algorithm\Database\Unity3D\OpenGL\OpenInventor



靡不有初，鲜克有终
» CSharpGL @ GitHub
» 我的博客汇总
» 捐赠列表
加入QQ群
昵称：BIT祝威
园龄：5年
荣誉：推荐博客
粉丝：624
关注：12
+加关注



## Unity3D核心类型一览

### 阅读目录(Content)



## Unity3D核心类型一览

@vczz Action appIdleAction = null; EventHandler appIdleEvent = null; public App.....

--39

2. Re:C#自定义控件：WinForm将其它应用程序窗体嵌入自己内部

@
好的，我自己再研究下下。

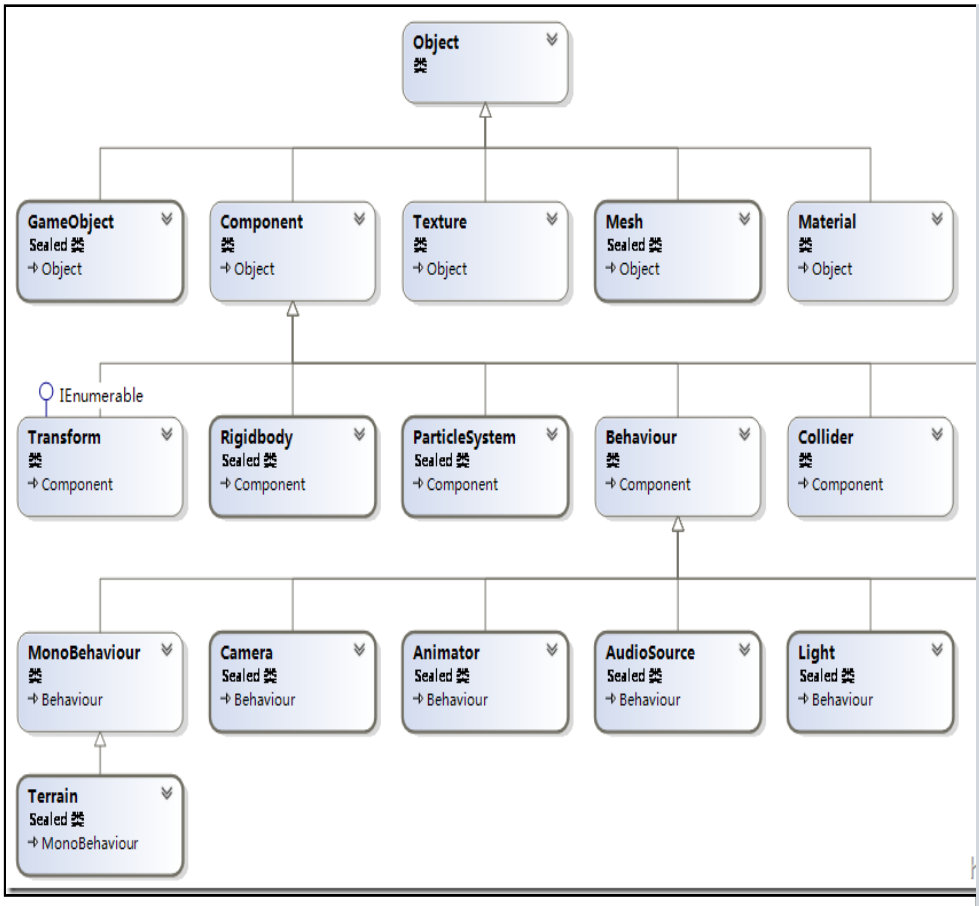
--vczz

3. Re:C#自定义控件：WinForm将其它应用程序窗体嵌入自己内部

@vczz引用 祝威你好，又是我，不好意思又来打扰你了。我在用你的软件时，发现如果嵌入的窗口比较大的话，往往显示的位置很奇怪，除非最大化，不然可能都看不到要嵌入的窗口在哪里（最大化后才能看到一部.....

--BIT祝威

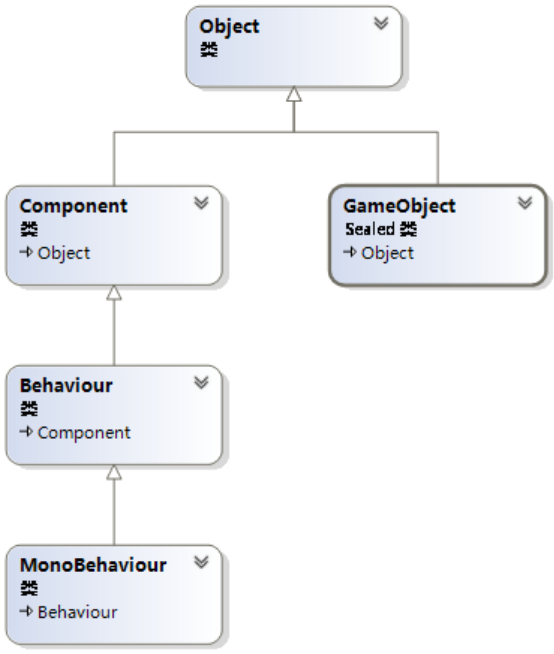
4. Re:C#自定义控件：WinForm将其它应用程序窗体嵌入自己内部




本文记录了Unity3D的最基本的核心类型。包括Object、GameObject、Component、Transform、Behaviour、Renderer、Collider、Rigidbody、Camera、Light、MonoBehaviour等。

需要展开了public类型方法的类图请点这里（http://www.cnblogs.com/bitzhuwei/gallery/image/152116.html）。

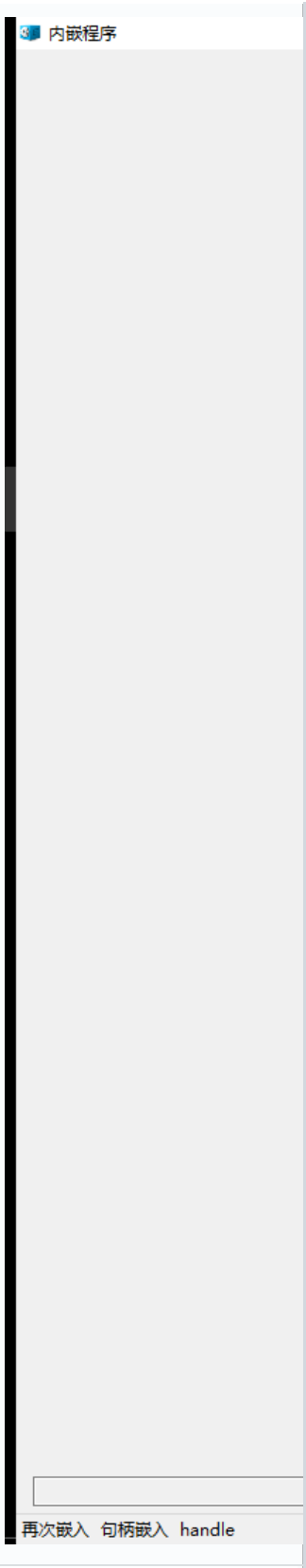
最核心的类型就这几个：Object、GameObject、Component、Behaviour、MonoBehaviour。




需要展开了这几个public类型方法的类图请点这里（http://www.cnblogs.com/bitzhuwei/gallery/ima



## UnityEngine.Object




--vczz

5. Re:C#自定义控件：WinForm将其它应用程序窗体嵌入自己内部

祝威你好，又是我，不好意思又来打扰你了。我在用你的软件时，发现如果嵌入的窗口比较大的话，往往显示的位置很奇怪，除非最大化，不然可能都看不到要嵌入的窗口在哪里（最大化后才能看到一部分），而我在看你的源码……

所有Unity3D的基类。

持有实例的ID信息。

实现了静态方法：增（Instantiate）删（Destroy）查（FindObjectsOfType）

Any public variable you make that derives from Object gets shown in the inspector as a drop target, allowing you to set the value from the GUI.

View Code

回到顶部(go to top)

## UnityEngine.GameObject

```
/// <summary>
/// game object contains components.
/// <para>Add Component</para>
/// <para>Find Component</para>
/// <para>common components</para>
/// <para>BroadcastMessage在这个游戏物体及其子物体的所有MonoBehaviour中调用名称为methodName的方法.</para>
/// </summary>
```

`GameObject.active` is obsolete. Use `GameObject.SetActive()`, `GameObject.activeSelf` (read only) or `GameObject.activeInHierarchy` (read only).

`gameObject.SetActiveRecursively()` is obsolete. Use `GameObject.SetActive()`, which is now inherited by children.

View Code

回到顶部(go to top)

## UnityEngine.Component

所有的Component，都会指向其所属的GameObject。

在脚本中
用 `this.renderer` ， `this.transform` ， `this.GetComponent(XXX)` ， `this.gameObject` 与 `this.gameObject.renderer` ， `this.gameObject.transform` ， `this.gameObject.GetComponent(XXX)` ， `this.gameObject.gameObject` 的结果是完全一样的。这意味着，你用 `this.renderer.transform.renderer.collider` 这种写法，仍然可以得到 `this.collider` 。（在这些组件不是 `null` 的前提下）

the `active` property is deprecated on components. Please use `gameObject.active` instead. If you meant to enable / disable a single component use `enabled` instead.

`GameObject.active` is obsolete. Use `GameObject.SetActive()`, `GameObject.activeSelf` (read only) or `GameObject.activeInHierarchy` (read only).

View Code

```
1         //this.gameObject.active = false;//GameObject.active is obselete
2         this.gameObject.SetActive(false);// ! use
3         this.gameObject.activeSelf = false;//readc
4         this.gameObject.activeInHierarchy = false;
5
```

```
6         //this.active = false;//Component.active is obsolete
7         this.transform.active = false;//cannot disable singly
8         this.particleSystem.active = false;//cannot disable singly
9         this.rigidbody.active = false;//cannot disable singly
10
11        this.GetComponent<TestEqual>().enabled = false;//work on single behaviour
12        this.renderer.enabled = false;//work on single renderer
13        this.collider.enabled = false;//work on single collider
```

回到顶部(go to top)

## UnityEngine.Texture

```
1  namespace UnityEngine
2  {
3      using System;
4      using System.Runtime.CompilerServices;
5      using System.Runtime.InteropServices;
6
7      public class Texture : UnityEngine.Object
8      {
9          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
10         public extern int GetNativeTextureID();
11         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
12         public extern IntPtr GetNativeTexturePtr();
13         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
14         private static extern int Internal_GetHeight(Texture mono);
15         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
16         private static extern void Internal_GetTexelSize(Texture tex, out Vector2
output);
17         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
18         private static extern int Internal_GetWidth(Texture mono);
19         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
20         public static extern void SetGlobalAnisotropicFilteringLimits(int forcedMin,
int globalMax);
21
22         public int anisoLevel { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
23
24         public static AnisotropicFiltering anisotropicFiltering {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
25
26         public FilterMode filterMode { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessI
set; }
27
28         public virtual int height
29         {
30             get
31             {
```

```
32                 return Internal_GetHeight(this);
33             }
34             set
35             {
36                 throw new Exception("not implemented");
37             }
38         }
39 
40         public static int masterTextureLimit {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
41 
42         public float mipMapBias { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
43 
44         public Vector2 texelSize
45         {
46             get
47             {
48                 Vector2 vector;
49                 Internal_GetTexelSize(this, out vector);
50                 return vector;
51             }
52         }
53 
54         public virtual int width
55         {
56             get
57             {
58                 return Internal_GetWidth(this);
59             }
60             set
61             {
62                 throw new Exception("not implemented");
63             }
64         }
65 
66         public TextureWrapMode wrapMode {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
67     }
68 }
```

回到顶部(go to top)

## UnityEngine.Mesh

```
1 namespace UnityEngine
2 {
3     using System;
4     using System.Runtime.CompilerServices;
```

```
 5     using System.Runtime.InteropServices;
 6     using UnityEngine.Internal;
 7
 8     public sealed class Mesh : UnityEngine.Object
 9     {
10         public Mesh()
11         {
12             Internal_Create(this);
13         }
14
15         [ExcludeFromDocs]
16         public void Clear()
17         {
18             bool keepVertexLayout = true;
19             this.Clear(keepVertexLayout);
20         }
21
22         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
23         public extern void Clear([DefaultValue("true")] bool keepVertexLayout);
24         [ExcludeFromDocs]
25         public void CombineMeshes(CombineInstance[] combine)
26         {
27             bool useMatrices = true;
28             bool mergeSubMeshes = true;
29             this.CombineMeshes(combine, mergeSubMeshes, useMatrices);
30         }
31
32         [ExcludeFromDocs]
33         public void CombineMeshes(CombineInstance[] combine, bool mergeSubMeshes)
34         {
35             bool useMatrices = true;
36             this.CombineMeshes(combine, mergeSubMeshes, useMatrices);
37         }
38
39         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
40         public extern void CombineMeshes(CombineInstance[] combine,
[DefaultValue("true")] bool mergeSubMeshes, [DefaultValue("true")] bool useMatrices);
41         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
42         public extern int GetBlendShapeIndex(string blendShapeName);
43         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
44         public extern string GetBlendShapeName(int index);
45         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
46         public extern int[] GetIndices(int submesh);
47         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
48         public extern MeshTopology GetTopology(int submesh);
49         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
50         public extern int[] GetTriangles(int submesh);
51         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall, Obsolete("Use
GetTriangles instead. Internally this function converts a list of triangles to a strip,
so it might be slow, it might be a mess.")]
52         public extern int[] GetTriangleStrip(int submesh);
53         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
54         private static extern void Internal_Create([Writable] Mesh mono);
55         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
56         private extern void INTERNAL_get_bounds(o
57         [MethodImpl(MethodImplOptions.InternalCal
58         private extern void INTERNAL_set_bounds(r
59         [MethodImpl(MethodImplOptions.InternalCal
```

```
60          public extern void MarkDynamic();
61          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
62          public extern void Optimize();
63          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
64          public extern void RecalculateBounds();
65          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
66          public extern void RecalculateNormals();
67          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
68          public extern void SetIndices(int[] indices, MeshTopology topology, int
submesh);
69          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
70          public extern void SetTriangles(int[] triangles, int submesh);
71          [MethodImpl(MethodImplOptions.InternalCall), Obsolete("Use SetTriangles
instead. Internally this function will convert the triangle strip to a list of triangles
anyway."), WrapperlessIcall]
72          public extern void SetTriangleStrip(int[] triangles, int submesh);
73          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
74          public extern void UploadMeshData(bool markNoLogerReadable);
75
76          public Matrix4x4[] bindposes { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
77
78          public int blendShapeCount { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
79
80          public BoneWeight[] boneWeights {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
81
82          public Bounds bounds
83          {
84              get
85              {
86                  Bounds bounds;
87                  this.INTERNAL_get_bounds(out bounds);
88                  return bounds;
89              }
90              set
91              {
92                  this.INTERNAL_set_bounds(ref value);
93              }
94          }
95
96          internal bool canAccess { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
97
98          public Color[] colors { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
99
100         public Color32[] colors32 { [MethodImpl(MethodImplOptions.InternalCa
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessI
set; }
101
102         public bool isReadable { [MethodImpl(Meth                    all)
WrapperlessIcall] get; }
103
```

```
104         public Vector3[] normals { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
105
106         public int subMeshCount { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
107
108         public Vector4[] tangents { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
109
110         public int[] triangles { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
111
112         public Vector2[] uv { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
113
114         public Vector2[] uv1
115         {
116             get
117             {
118                 return this.uv2;
119             }
120             set
121             {
122                 this.uv2 = value;
123             }
124         }
125
126         public Vector2[] uv2 { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
127
128         public int vertexCount { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
129
130         public Vector3[] vertices { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
131     }
132 }
```

回到顶部(go to top)

## UnityEngine.Material

```
1 namespace UnityEngine
2 {
3     using System;
```

```
4   using System.Runtime.CompilerServices;
5   using System.Runtime.InteropServices;
6   using UnityEngine.Internal;
7
8   public class Material : UnityEngine.Object
9   {
10      public Material(string contents)
11      {
12          Internal_CreateWithString(this, contents);
13      }
14
15      public Material(Material source)
16      {
17          Internal_CreateWithMaterial(this, source);
18      }
19
20      public Material(Shader shader)
21      {
22          Internal_CreateWithShader(this, shader);
23      }
24
25      [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
26      public extern void CopyPropertiesFromMaterial(Material mat);
27      [Obsolete("Use the Material constructor instead.")]
28      public static Material Create(string scriptContents)
29      {
30          return new Material(scriptContents);
31      }
32
33      [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
34      public extern void DisableKeyword(string keyword);
35      [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
36      public extern void EnableKeyword(string keyword);
37      [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
38      public extern Color GetColor(int nameID);
39      public Color GetColor(string propertyName)
40      {
41          return this.GetColor(Shader.PropertyToID(propertyName));
42      }
43
44      [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
45      public extern float GetFloat(int nameID);
46      public float GetFloat(string propertyName)
47      {
48          return this.GetFloat(Shader.PropertyToID(propertyName));
49      }
50
51      public int GetInt(int nameID)
52      {
53          return (int) this.GetFloat(nameID);
54      }
55
56      public int GetInt(string propertyName)
57      {
58          return (int) this.GetFloat(propertyNa
59      }
60
61      [MethodImpl(MethodImplOptions.InternalCall)
```

```
62          public extern Matrix4x4 GetMatrix(int nameID);
63          public Matrix4x4 GetMatrix(string propertyName)
64          {
65              return this.GetMatrix(Shader.PropertyToID(propertyName));
66          }
67
68          [ExcludeFromDocs]
69          public string GetTag(string tag, bool searchFallbacks)
70          {
71              string defaultValue = string.Empty;
72              return this.GetTag(tag, searchFallbacks, defaultValue);
73          }
74
75          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
76          public extern string GetTag(string tag, bool searchFallbacks,
[DefaultValue("\"\"")] string defaultValue);
77          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
78          public extern Texture GetTexture(int nameID);
79          public Texture GetTexture(string propertyName)
80          {
81              return this.GetTexture(Shader.PropertyToID(propertyName));
82          }
83
84          public Vector2 GetTextureOffset(string propertyName)
85          {
86              Vector2 vector;
87              Internal_GetTextureOffset(this, propertyName, out vector);
88              return vector;
89          }
90
91          public Vector2 GetTextureScale(string propertyName)
92          {
93              Vector2 vector;
94              Internal_GetTextureScale(this, propertyName, out vector);
95              return vector;
96          }
97
98          public Vector4 GetVector(int nameID)
99          {
100             Color color = this.GetColor(nameID);
101             return new Vector4(color.r, color.g, color.b, color.a);
102         }
103
104         public Vector4 GetVector(string propertyName)
105         {
106             Color color = this.GetColor(propertyName);
107             return new Vector4(color.r, color.g, color.b, color.a);
108         }
109
110         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
111         public extern bool HasProperty(int nameID);
112         public bool HasProperty(string propertyName)
113         {
114             return this.HasProperty(Shader.PropertyToID(propertyName));
115         }
116
117         [MethodImpl(MethodImplOptions.InternalCal
118         private static extern void INTERNAL_CALL_
```

```
ref Color color);
119         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
120         private static extern void INTERNAL_CALL_SetMatrix(Material self, int
nameID, ref Matrix4x4 matrix);
121         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
122         private static extern void INTERNAL_CALL_SetTextureOffset(Material self,
string propertyName, ref Vector2 offset);
123         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
124         private static extern void INTERNAL_CALL_SetTextureScale(Material self,
string propertyName, ref Vector2 scale);
125         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
126         private static extern void Internal_CreateWithMaterial([Writable] Material
mono, Material source);
127         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
128         private static extern void Internal_CreateWithShader([Writable] Material
mono, Shader shader);
129         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
130         private static extern void Internal_CreateWithString([Writable] Material
mono, string contents);
131         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
132         private static extern void Internal_GetTextureOffset(Material mat, string
name, out Vector2 output);
133         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
134         private static extern void Internal_GetTextureScale(Material mat, string
name, out Vector2 output);
135         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
136         public extern void Lerp(Material start, Material end, float t);
137         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
138         public extern void SetBuffer(string propertyName, ComputeBuffer buffer);
139         public void SetColor(int nameID, Color color)
140         {
141             INTERNAL_CALL_SetColor(this, nameID, ref color);
142         }
143
144         public void SetColor(string propertyName, Color color)
145         {
146             this.SetColor(Shader.PropertyToID(propertyName), color);
147         }
148
149         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
150         public extern void SetFloat(int nameID, float value);
151         public void SetFloat(string propertyName, float value)
152         {
153             this.SetFloat(Shader.PropertyToID(propertyName), value);
154         }
155
156         public void SetInt(int nameID, int value)
157         {
158             this.SetFloat(nameID, (float) value);
159         }
160
161         public void SetInt(string propertyName, int value)
162         {
163             this.SetFloat(propertyName, (float) value);
164         }
165
166         public void SetMatrix(int nameID, Matrix4
167         {
```

```
168             INTERNAL_CALL_SetMatrix(this, nameID, ref matrix);
169         }
170 
171         public void SetMatrix(string propertyName, Matrix4x4 matrix)
172         {
173             this.SetMatrix(Shader.PropertyToID(propertyName), matrix);
174         }
175 
176         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
177         public extern bool SetPass(int pass);
178         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
179         public extern void SetTexture(int nameID, Texture texture);
180         public void SetTexture(string propertyName, Texture texture)
181         {
182             this.SetTexture(Shader.PropertyToID(propertyName), texture);
183         }
184 
185         public void SetTextureOffset(string propertyName, Vector2 offset)
186         {
187             INTERNAL_CALL_SetTextureOffset(this, propertyName, ref offset);
188         }
189 
190         public void SetTextureScale(string propertyName, Vector2 scale)
191         {
192             INTERNAL_CALL_SetTextureScale(this, propertyName, ref scale);
193         }
194 
195         public void SetVector(int nameID, Vector4 vector)
196         {
197             this.SetColor(nameID, new Color(vector.x, vector.y, vector.z,
vector.w));
198         }
199 
200         public void SetVector(string propertyName, Vector4 vector)
201         {
202             this.SetColor(propertyName, new Color(vector.x, vector.y, vector.z,
vector.w));
203         }
204 
205         public Color color
206         {
207             get
208             {
209                 return this.GetColor("_Color");
210             }
211             set
212             {
213                 this.SetColor("_Color", value);
214             }
215         }
216 
217         public Texture mainTexture
218         {
219             get
220             {
221                 return this.GetTexture("_MainTex"
222             }
223             set
```

```
224             {
225                 this.SetTexture("_MainTex", value);
226             }
227         }
228 
229         public Vector2 mainTextureOffset
230         {
231             get
232             {
233                 return this.GetTextureOffset("_MainTex");
234             }
235             set
236             {
237                 this.SetTextureOffset("_MainTex", value);
238             }
239         }
240 
241         public Vector2 mainTextureScale
242         {
243             get
244             {
245                 return this.GetTextureScale("_MainTex");
246             }
247             set
248             {
249                 this.SetTextureScale("_MainTex", value);
250             }
251         }
252 
253         public int passCount { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
254 
255         public int renderQueue { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
256 
257         public Shader shader { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
258 
259         public string[] shaderKeywords {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
260     }
261 }
```

回到顶部(go to top)

## UnityEngine.Transform

```
 1 namespace UnityEngine
 2 {
```

```
 3     using System;
 4     using System.Collections;
 5     using System.Runtime.CompilerServices;
 6     using System.Runtime.InteropServices;
 7     using UnityEngine.Internal;
 8
 9     public class Transform : Component, IEnumerable
10     {
11         protected Transform()
12         {
13         }
14
15         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
16         public extern void DetachChildren();
17         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
18         public extern Transform Find(string name);
19         public Transform FindChild(string name)
20         {
21             return this.Find(name);
22         }
23
24         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
25         public extern Transform GetChild(int index);
26         [MethodImpl(MethodImplOptions.InternalCall), Obsolete("use
Transform.childCount instead."), WrapperlessIcall]
27         public extern int GetChildCount();
28         public IEnumerator GetEnumerator()
29         {
30             return new Enumerator(this);
31         }
32
33         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
34         public extern int GetSiblingIndex();
35         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
36         private static extern Vector3
INTERNAL_CALL_InverseTransformDirection(Transform self, ref Vector3 direction);
37         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
38         private static extern Vector3 INTERNAL_CALL_InverseTransformPoint(Transform
self, ref Vector3 position);
39         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
40         private static extern Vector3 INTERNAL_CALL_InverseTransformVector(Transform
self, ref Vector3 vector);
41         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
42         private static extern void INTERNAL_CALL_LookAt(Transform self, ref Vector3
worldPosition, ref Vector3 worldUp);
43         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
44         private static extern void INTERNAL_CALL_RotateAround(Transform self, ref
Vector3 axis, float angle);
45         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
46         private static extern void INTERNAL_CALL_RotateAroundInternal(Transform
self, ref Vector3 axis, float angle);
47         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
48         private static extern void INTERNAL_CALL_RotateAroundLocal(Transform
ref Vector3 axis, float angle);
49         [MethodImpl(MethodImplOptions.InternalCal
50         private static extern Vector3 INTERNAL_CA                    rans
self, ref Vector3 direction);
51         [MethodImpl(MethodImplOptions.InternalCal
```

```
52          private static extern Vector3 INTERNAL_CALL_TransformPoint(Transform self,
ref Vector3 position);
53          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
54          private static extern Vector3 INTERNAL_CALL_TransformVector(Transform self,
ref Vector3 vector);
55          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
56          private extern void INTERNAL_get_localEulerAngles(out Vector3 value);
57          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
58          private extern void INTERNAL_get_localPosition(out Vector3 value);
59          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
60          private extern void INTERNAL_get_localRotation(out Quaternion value);
61          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
62          private extern void INTERNAL_get_localScale(out Vector3 value);
63          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
64          private extern void INTERNAL_get_localToWorldMatrix(out Matrix4x4 value);
65          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
66          private extern void INTERNAL_get_lossyScale(out Vector3 value);
67          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
68          private extern void INTERNAL_get_position(out Vector3 value);
69          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
70          private extern void INTERNAL_get_rotation(out Quaternion value);
71          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
72          private extern void INTERNAL_get_worldToLocalMatrix(out Matrix4x4 value);
73          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
74          private extern void INTERNAL_set_localEulerAngles(ref Vector3 value);
75          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
76          private extern void INTERNAL_set_localPosition(ref Vector3 value);
77          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
78          private extern void INTERNAL_set_localRotation(ref Quaternion value);
79          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
80          private extern void INTERNAL_set_localScale(ref Vector3 value);
81          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
82          private extern void INTERNAL_set_position(ref Vector3 value);
83          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
84          private extern void INTERNAL_set_rotation(ref Quaternion value);
85          public Vector3 InverseTransformDirection(Vector3 direction)
86          {
87              return INTERNAL_CALL_InverseTransformDirection(this, ref direction);
88          }
89
90          public Vector3 InverseTransformDirection(float x, float y, float z)
91          {
92              return this.InverseTransformDirection(new Vector3(x, y, z));
93          }
94
95          public Vector3 InverseTransformPoint(Vector3 position)
96          {
97              return INTERNAL_CALL_InverseTransformPoint(this, ref position);
98          }
99
100         public Vector3 InverseTransformPoint(float x, float y, float z)
101         {
102             return this.InverseTransformPoint(new Vector3(x, y, z));
103         }
104
105         public Vector3 InverseTransformVector(Vec
106         {
107             return INTERNAL_CALL_InverseTransform
```

```
108         }
109
110         public Vector3 InverseTransformVector(float x, float y, float z)
111         {
112             return this.InverseTransformVector(new Vector3(x, y, z));
113         }
114
115         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
116         public extern bool IsChildOf(Transform parent);
117         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
118         internal extern bool IsNonUniformScaleTransform();
119         [ExcludeFromDocs]
120         public void LookAt(Transform target)
121         {
122             Vector3 up = Vector3.up;
123             this.LookAt(target, up);
124         }
125
126         [ExcludeFromDocs]
127         public void LookAt(Vector3 worldPosition)
128         {
129             Vector3 up = Vector3.up;
130             INTERNAL_CALL_LookAt(this, ref worldPosition, ref up);
131         }
132
133         public void LookAt(Transform target, [DefaultValue("Vector3.up")] Vector3
worldUp)
134         {
135             if (target != null)
136             {
137                 this.LookAt(target.position, worldUp);
138             }
139         }
140
141         public void LookAt(Vector3 worldPosition, [DefaultValue("Vector3.up")]
Vector3 worldUp)
142         {
143             INTERNAL_CALL_LookAt(this, ref worldPosition, ref worldUp);
144         }
145
146         [ExcludeFromDocs]
147         public void Rotate(Vector3 eulerAngles)
148         {
149             Space self = Space.Self;
150             this.Rotate(eulerAngles, self);
151         }
152
153         [ExcludeFromDocs]
154         public void Rotate(Vector3 axis, float angle)
155         {
156             Space self = Space.Self;
157             this.Rotate(axis, angle, self);
158         }
159
160         public void Rotate(Vector3 eulerAngles, [
relativeTo)
161         {
162             Quaternion quaternion = Quaternion.Euler(
```

```
eulerAngles.z);
163             if (relativeTo == Space.Self)
164             {
165                 this.localRotation *= quaternion;
166             }
167             else
168             {
169                 this.rotation *= (Quaternion.Inverse(this.rotation) * quaternion) *
this.rotation;
170             }
171         }
172 
173         [ExcludeFromDocs]
174         public void Rotate(float xAngle, float yAngle, float zAngle)
175         {
176             Space self = Space.Self;
177             this.Rotate(xAngle, yAngle, zAngle, self);
178         }
179 
180         public void Rotate(Vector3 axis, float angle, [DefaultValue("Space.Self")]
Space relativeTo)
181         {
182             if (relativeTo == Space.Self)
183             {
184                 this.RotateAroundInternal(base.transform.TransformDirection(axis),
angle * 0.01745329f);
185             }
186             else
187             {
188                 this.RotateAroundInternal(axis, angle * 0.01745329f);
189             }
190         }
191 
192         public void Rotate(float xAngle, float yAngle, float zAngle,
[DefaultValue("Space.Self")] Space relativeTo)
193         {
194             this.Rotate(new Vector3(xAngle, yAngle, zAngle), relativeTo);
195         }
196 
197         [Obsolete("use Transform.Rotate instead.")]
198         public void RotateAround(Vector3 axis, float angle)
199         {
200             INTERNAL_CALL_RotateAround(this, ref axis, angle);
201         }
202 
203         public void RotateAround(Vector3 point, Vector3 axis, float angle)
204         {
205             Vector3 position = this.position;
206             Quaternion quaternion = Quaternion.AngleAxis(angle, axis);
207             Vector3 vector2 = position - point;
208             vector2 = (Vector3) (quaternion * vector2);
209             position = point + vector2;
210             this.position = position;
211             this.RotateAroundInternal(axis, angle * 0.01745329f);
212         }
213 
214         internal void RotateAroundInternal(Vector
215         {
```

```
216             INTERNAL_CALL_RotateAroundInternal(this, ref axis, angle);
217         }
218 
219         [Obsolete("use Transform.Rotate instead.")]
220         public void RotateAroundLocal(Vector3 axis, float angle)
221         {
222             INTERNAL_CALL_RotateAroundLocal(this, ref axis, angle);
223         }
224 
225         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
226         internal extern void SendTransformChangedScale();
227         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
228         public extern void SetAsFirstSibling();
229         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
230         public extern void SetAsLastSibling();
231         public void SetParent(Transform parent)
232         {
233             this.SetParent(parent, true);
234         }
235 
236         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
237         public extern void SetParent(Transform parent, bool worldPositionStays);
238         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
239         public extern void SetSiblingIndex(int index);
240         public Vector3 TransformDirection(Vector3 direction)
241         {
242             return INTERNAL_CALL_TransformDirection(this, ref direction);
243         }
244 
245         public Vector3 TransformDirection(float x, float y, float z)
246         {
247             return this.TransformDirection(new Vector3(x, y, z));
248         }
249 
250         public Vector3 TransformPoint(Vector3 position)
251         {
252             return INTERNAL_CALL_TransformPoint(this, ref position);
253         }
254 
255         public Vector3 TransformPoint(float x, float y, float z)
256         {
257             return this.TransformPoint(new Vector3(x, y, z));
258         }
259 
260         public Vector3 TransformVector(Vector3 vector)
261         {
262             return INTERNAL_CALL_TransformVector(this, ref vector);
263         }
264 
265         public Vector3 TransformVector(float x, float y, float z)
266         {
267             return this.TransformVector(new Vector3(x, y, z));
268         }
269 
270         [ExcludeFromDocs]
271         public void Translate(Vector3 translation
272         {
273             Space self = Space.Self;
```

```
274             this.Translate(translation, self);
275         }
276 
277         public void Translate(Vector3 translation, [DefaultValue("Space.Self")]
Space relativeTo)
278         {
279             if (relativeTo == Space.World)
280             {
281                 this.position += translation;
282             }
283             else
284             {
285                 this.position += this.TransformDirection(translation);
286             }
287         }
288 
289         public void Translate(Vector3 translation, Transform relativeTo)
290         {
291             if (relativeTo != null)
292             {
293                 this.position += relativeTo.TransformDirection(translation);
294             }
295             else
296             {
297                 this.position += translation;
298             }
299         }
300 
301         [ExcludeFromDocs]
302         public void Translate(float x, float y, float z)
303         {
304             Space self = Space.Self;
305             this.Translate(x, y, z, self);
306         }
307 
308         public void Translate(float x, float y, float z,
[DefaultValue("Space.Self")] Space relativeTo)
309         {
310             this.Translate(new Vector3(x, y, z), relativeTo);
311         }
312 
313         public void Translate(float x, float y, float z, Transform relativeTo)
314         {
315             this.Translate(new Vector3(x, y, z), relativeTo);
316         }
317 
318         public int childCount { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
319 
320         public Vector3 eulerAngles
321         {
322             get
323             {
324                 return this.rotation.eulerAngles;
325             }
326             set
327             {
328                 this.rotation = Quaternion.Euler(value);
```

```
329             }
330         }
331 
332         public Vector3 forward
333         {
334             get
335             {
336                 return (Vector3) (this.rotation * Vector3.forward);
337             }
338             set
339             {
340                 this.rotation = Quaternion.LookRotation(value);
341             }
342         }
343 
344         public bool hasChanged { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
345 
346         public Vector3 localEulerAngles
347         {
348             get
349             {
350                 Vector3 vector;
351                 this.INTERNAL_get_localEulerAngles(out vector);
352                 return vector;
353             }
354             set
355             {
356                 this.INTERNAL_set_localEulerAngles(ref value);
357             }
358         }
359 
360         public Vector3 localPosition
361         {
362             get
363             {
364                 Vector3 vector;
365                 this.INTERNAL_get_localPosition(out vector);
366                 return vector;
367             }
368             set
369             {
370                 this.INTERNAL_set_localPosition(ref value);
371             }
372         }
373 
374         public Quaternion localRotation
375         {
376             get
377             {
378                 Quaternion quaternion;
379                 this.INTERNAL_get_localRotation(out quaternion);
380                 return quaternion;
381             }
382             set
383             {
384                 this.INTERNAL_set_localRotation(ref value);
```

```
385             }
386         }
387 
388         public Vector3 localScale
389         {
390             get
391             {
392                 Vector3 vector;
393                 this.INTERNAL_get_localScale(out vector);
394                 return vector;
395             }
396             set
397             {
398                 this.INTERNAL_set_localScale(ref value);
399             }
400         }
401 
402         public Matrix4x4 localToWorldMatrix
403         {
404             get
405             {
406                 Matrix4x4 matrixx;
407                 this.INTERNAL_get_localToWorldMatrix(out matrixx);
408                 return matrixx;
409             }
410         }
411 
412         public Vector3 lossyScale
413         {
414             get
415             {
416                 Vector3 vector;
417                 this.INTERNAL_get_lossyScale(out vector);
418                 return vector;
419             }
420         }
421 
422         public Transform parent
423         {
424             get
425             {
426                 return this.parentInternal;
427             }
428             set
429             {
430                 if (this is RectTransform)
431                 {
432                     Debug.LogWarning("Parent of RectTransform is being set with parent property. Consider using the SetParent method instead, with the worldPositionStays argument set to false. This will retain local orientation and scale rather than world orientation and scale, which can prevent common UI scaling issues.", this);
433                 }
434                 this.parentInternal = value;
435             }
436         }
437 
438         internal Transform parentInternal {
```

```
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
439
440         public Vector3 position
441         {
442             get
443             {
444                 Vector3 vector;
445                 this.INTERNAL_get_position(out vector);
446                 return vector;
447             }
448             set
449             {
450                 this.INTERNAL_set_position(ref value);
451             }
452         }
453
454         public Vector3 right
455         {
456             get
457             {
458                 return (Vector3) (this.rotation * Vector3.right);
459             }
460             set
461             {
462                 this.rotation = Quaternion.FromToRotation(Vector3.right, value);
463             }
464         }
465
466         public Transform root { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
467
468         public Quaternion rotation
469         {
470             get
471             {
472                 Quaternion quaternion;
473                 this.INTERNAL_get_rotation(out quaternion);
474                 return quaternion;
475             }
476             set
477             {
478                 this.INTERNAL_set_rotation(ref value);
479             }
480         }
481
482         public Vector3 up
483         {
484             get
485             {
486                 return (Vector3) (this.rotation * Vector3.up);
487             }
488             set
489             {
490                 this.rotation = Quaternion.FromTo                    ue)
491             }
492         }
493
```

```
494         public Matrix4x4 worldToLocalMatrix
495         {
496             get
497             {
498                 Matrix4x4 matrixx;
499                 this.INTERNAL_get_worldToLocalMatrix(out matrixx);
500                 return matrixx;
501             }
502         }
503 
504         private sealed class Enumerator : IEnumerator
505         {
506             private int currentIndex = -1;
507             private Transform outer;
508 
509             internal Enumerator(Transform outer)
510             {
511                 this.outer = outer;
512             }
513 
514             public bool MoveNext()
515             {
516                 int childCount = this.outer.childCount;
517                 return (++this.currentIndex < childCount);
518             }
519 
520             public void Reset()
521             {
522                 this.currentIndex = -1;
523             }
524 
525             public object Current
526             {
527                 get
528                 {
529                     return this.outer.GetChild(this.currentIndex);
530                 }
531             }
532         }
533     }
534 }
```

回到顶部(go to top)

## UnityEngine.Renderer

```
1 namespace UnityEngine
2 {
3     using System;
4     using System.Runtime.CompilerServices;
5     using System.Runtime.InteropServices;
6 
```

```
7     public class Renderer : Component
8     {
9         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
10        public extern void GetPropertyBlock(MaterialPropertyBlock dest);
11        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
12        private extern void INTERNAL_get_lightmapTilingOffset(out Vector4 value);
13        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
14        private extern void INTERNAL_get_localToWorldMatrix(out Matrix4x4 value);
15        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
16        private extern void INTERNAL_get_worldToLocalMatrix(out Matrix4x4 value);
17        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
18        private extern void INTERNAL_set_lightmapTilingOffset(ref Vector4 value);
19        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
20        public extern void Render(int material);
21        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
22        public extern void SetPropertyBlock(MaterialPropertyBlock properties);
23        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
24        internal extern void SetSubsetIndex(int index, int subSetIndexForMaterial);
25
26        public Bounds bounds { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
27
28        public bool castShadows { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
29
30        public bool enabled { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
31
32        public bool isPartOfStaticBatch {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
33
34        public bool isVisible { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
35
36        public int lightmapIndex { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
37
38        public Vector4 lightmapTilingOffset
39        {
40            get
41            {
42                Vector4 vector;
43                this.INTERNAL_get_lightmapTilingOffset(out vector);
44                return vector;
45            }
46            set
47            {
48                this.INTERNAL_set_lightmapTilingOffset(ref value);
49            }
50        }
51
52        public Transform lightProbeAnchor {
[MethodImpl(MethodImplOptions.InternalCall), Wrapperl
[MethodImpl(MethodImplOptions.InternalCall), Wrapperl
53
```

```
54          public Matrix4x4 localToWorldMatrix
55          {
56              get
57              {
58                  Matrix4x4 matrixx;
59                  this.INTERNAL_get_localToWorldMatrix(out matrixx);
60                  return matrixx;
61              }
62          }
63
64          public Material material { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
65
66          public Material[] materials { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
67
68          public bool receiveShadows { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
69
70          public Material sharedMaterial { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
71
72          public Material[] sharedMaterials {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
73
74          public int sortingLayerID { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
75
76          public string sortingLayerName { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
77
78          public int sortingOrder { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
79
80          internal int staticBatchIndex { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
81
82          internal Transform staticBatchRootTransform {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
83
84          public bool useLightProbes { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
85
86          public Matrix4x4 worldToLocalMatrix
87          {
88              get
89              {
90                  Matrix4x4 matrixx;
```

```
91                 this.INTERNAL_get_worldToLocalMatrix(out matrixx);
92                 return matrixx;
93             }
94         }
95     }
96 }
```

回到顶部(go to top)

## UnityEngine.ParticalSystem

```
1  namespace UnityEngine
2  {
3      using System;
4      using System.Collections;
5      using System.Collections.Generic;
6      using System.Runtime.CompilerServices;
7      using System.Runtime.InteropServices;
8      using UnityEngine.Internal;
9
10     public sealed class ParticleSystem : Component
11     {
12         [ExcludeFromDocs]
13         public void Clear()
14         {
15             bool withChildren = true;
16             this.Clear(withChildren);
17         }
18
19         public void Clear([DefaultValue("true")] bool withChildren)
20         {
21             if (withChildren)
22             {
23                 foreach (ParticleSystem system in GetParticleSystems(this))
24                 {
25                     system.Internal_Clear();
26                 }
27             }
28             else
29             {
30                 this.Internal_Clear();
31             }
32         }
33
34         public void Emit(int count)
35         {
36             INTERNAL_CALL_Emit(this, count);
37         }
38
39         public void Emit(Particle particle)
40         {
41             this.Internal_Emit(ref particle);
```

```
42         }
43
44         public void Emit(Vector3 position, Vector3 velocity, float size, float
lifetime, Color32 color)
45         {
46             Particle particle = new Particle {
47                 position = position,
48                 velocity = velocity,
49                 lifetime = lifetime,
50                 startLifetime = lifetime,
51                 size = size,
52                 rotation = 0f,
53                 angularVelocity = 0f,
54                 color = color,
55                 randomSeed = 5
56             };
57             this.Internal_Emit(ref particle);
58         }
59
60         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
61         public extern int GetCollisionEvents(GameObject go, CollisionEvent[]
collisionEvents);
62         private static void GetDirectParticleSystemChildrenRecursive(Transform
transform, List<ParticleSystem> particleSystems)
63         {
64             IEnumerator enumerator = transform.GetEnumerator();
65             try
66             {
67                 while (enumerator.MoveNext())
68                 {
69                     Transform current = (Transform) enumerator.Current;
70                     ParticleSystem component =
current.gameObject.GetComponent<ParticleSystem>();
71                     if (component != null)
72                     {
73                         particleSystems.Add(component);
74                         GetDirectParticleSystemChildrenRecursive(current,
particleSystems);
75                     }
76                 }
77             }
78             finally
79             {
80                 IDisposable disposable = enumerator as IDisposable;
81                 if (disposable == null)
82                 {
83                 }
84                 disposable.Dispose();
85             }
86         }
87
88         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
89         public extern int GetParticles(Particle[] particles);
90         internal static ParticleSystem[] GetParticleSystems(ParticleSystem r
91         {
92             if (root == null)
93             {
94                 return null;
```

```
 95             }
 96             List<ParticleSystem> particleSystems = new List<ParticleSystem> {
 97                 root
 98             };
 99             GetDirectParticleSystemChildrenRecursive(root.transform,
particleSystems);
100             return particleSystems.ToArray();
101         }
102 
103         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
104         internal static extern Collider InstanceIDToCollider(int instanceID);
105         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
106         private static extern void INTERNAL_CALL_Emit(ParticleSystem self, int
count);
107         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
108         private extern void Internal_Clear();
109         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
110         private extern void Internal_Emit(ref Particle particle);
111         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
112         private extern void INTERNAL_get_startColor(out Color value);
113         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
114         private extern bool Internal_IsAlive();
115         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
116         private extern void Internal_Pause();
117         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
118         private extern void Internal_Play();
119         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
120         private extern void INTERNAL_set_startColor(ref Color value);
121         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
122         private extern void Internal_Simulate(float t, bool restart);
123         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
124         private extern void Internal_Stop();
125         [ExcludeFromDocs]
126         public bool IsAlive()
127         {
128             bool withChildren = true;
129             return this.IsAlive(withChildren);
130         }
131 
132         public bool IsAlive([DefaultValue("true")] bool withChildren)
133         {
134             if (!withChildren)
135             {
136                 return this.Internal_IsAlive();
137             }
138             foreach (ParticleSystem system in GetParticleSystems(this))
139             {
140                 if (system.Internal_IsAlive())
141                 {
142                     return true;
143                 }
144             }
145             return false;
146         }
147 
148         [ExcludeFromDocs]
149         public void Pause()
150         {
```

```
151             bool withChildren = true;
152             this.Pause(withChildren);
153         }
154 
155         public void Pause([DefaultValue("true")] bool withChildren)
156         {
157             if (withChildren)
158             {
159                 foreach (ParticleSystem system in GetParticleSystems(this))
160                 {
161                     system.Internal_Pause();
162                 }
163             }
164             else
165             {
166                 this.Internal_Pause();
167             }
168         }
169 
170         [ExcludeFromDocs]
171         public void Play()
172         {
173             bool withChildren = true;
174             this.Play(withChildren);
175         }
176 
177         public void Play([DefaultValue("true")] bool withChildren)
178         {
179             if (withChildren)
180             {
181                 foreach (ParticleSystem system in GetParticleSystems(this))
182                 {
183                     system.Internal_Play();
184                 }
185             }
186             else
187             {
188                 this.Internal_Play();
189             }
190         }
191 
192         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
193         public extern void SetParticles(Particle[] particles, int size);
194         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
195         internal extern void SetupDefaultType(int type);
196         [ExcludeFromDocs]
197         public void Simulate(float t)
198         {
199             bool restart = true;
200             bool withChildren = true;
201             this.Simulate(t, withChildren, restart);
202         }
203 
204         [ExcludeFromDocs]
205         public void Simulate(float t, bool withCh
206         {
207             bool restart = true;
208             this.Simulate(t, withChildren, restar
```

```
209         }
210
211         public void Simulate(float t, [DefaultValue("true")] bool withChildren,
[DefaultValue("true")] bool restart)
212         {
213             if (withChildren)
214             {
215                 foreach (ParticleSystem system in GetParticleSystems(this))
216                 {
217                     system.Internal_Simulate(t, restart);
218                 }
219             }
220             else
221             {
222                 this.Internal_Simulate(t, restart);
223             }
224         }
225
226         [ExcludeFromDocs]
227         public void Stop()
228         {
229             bool withChildren = true;
230             this.Stop(withChildren);
231         }
232
233         public void Stop([DefaultValue("true")] bool withChildren)
234         {
235             if (withChildren)
236             {
237                 foreach (ParticleSystem system in GetParticleSystems(this))
238                 {
239                     system.Internal_Stop();
240                 }
241             }
242             else
243             {
244                 this.Internal_Stop();
245             }
246         }
247
248         public float duration { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
249
250         public float emissionRate { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
251
252         public bool enableEmission { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
253
254         public float gravityModifier { [MethodImpl(MethodImplOptions.Interna
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessI
set; }
255
256         public bool isPaused { [MethodImpl(Metho                    ),
WrapperlessIcall] get; }
257
```

```
258         public bool isPlaying { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
259
260         public bool isStopped { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
261
262         public bool loop { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
263
264         public int maxParticles { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
265
266         public int particleCount { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
267
268         public float playbackSpeed { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
269
270         public bool playOnAwake { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
271
272         public uint randomSeed { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
273
274         public int safeCollisionEventSize {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
275
276         public ParticleSystemSimulationSpace simulationSpace {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
277
278         public Color startColor
279         {
280             get
281             {
282                 Color color;
283                 this.INTERNAL_get_startColor(out color);
284                 return color;
285             }
286             set
287             {
288                 this.INTERNAL_set_startColor(ref value);
289             }
290         }
291
292         public float startDelay { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessI
set; }
293
294         public float startLifetime { [MethodImpl(
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.
set; }
295
```

```
296         public float startRotation { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
297
298         public float startSize { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
299
300         public float startSpeed { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
301
302         public float time { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
303
304         [StructLayout(LayoutKind.Sequential)]
305         public struct CollisionEvent
306         {
307             private Vector3 m_Intersection;
308             private Vector3 m_Normal;
309             private Vector3 m_Velocity;
310             private int m_ColliderInstanceID;
311             public Vector3 intersection
312             {
313                 get
314                 {
315                     return this.m_Intersection;
316                 }
317             }
318             public Vector3 normal
319             {
320                 get
321                 {
322                     return this.m_Normal;
323                 }
324             }
325             public Vector3 velocity
326             {
327                 get
328                 {
329                     return this.m_Velocity;
330                 }
331             }
332             public Collider collider
333             {
334                 get
335                 {
336                     return
ParticleSystem.InstanceIDToCollider(this.m_ColliderInstanceID);
337                 }
338             }
339         }
340
341         [StructLayout(LayoutKind.Sequential)]
342         public struct Particle
343         {
344             private Vector3 m_Position;
```

```
345             private Vector3 m_Velocity;
346             private Vector3 m_AnimatedVelocity;
347             private Vector3 m_AxisOfRotation;
348             private float m_Rotation;
349             private float m_AngularVelocity;
350             private float m_Size;
351             private Color32 m_Color;
352             private uint m_RandomSeed;
353             private float m_Lifetime;
354             private float m_StartLifetime;
355             private float m_EmitAccumulator0;
356             private float m_EmitAccumulator1;
357             public Vector3 position
358             {
359                 get
360                 {
361                     return this.m_Position;
362                 }
363                 set
364                 {
365                     this.m_Position = value;
366                 }
367             }
368             public Vector3 velocity
369             {
370                 get
371                 {
372                     return this.m_Velocity;
373                 }
374                 set
375                 {
376                     this.m_Velocity = value;
377                 }
378             }
379             public float lifetime
380             {
381                 get
382                 {
383                     return this.m_Lifetime;
384                 }
385                 set
386                 {
387                     this.m_Lifetime = value;
388                 }
389             }
390             public float startLifetime
391             {
392                 get
393                 {
394                     return this.m_StartLifetime;
395                 }
396                 set
397                 {
398                     this.m_StartLifetime = value;
399                 }
400             }
401             public float size
402             {
```

```
403                 get
404                 {
405                     return this.m_Size;
406                 }
407                 set
408                 {
409                     this.m_Size = value;
410                 }
411             }
412             public Vector3 axisOfRotation
413             {
414                 get
415                 {
416                     return this.m_AxisOfRotation;
417                 }
418                 set
419                 {
420                     this.m_AxisOfRotation = value;
421                 }
422             }
423             public float rotation
424             {
425                 get
426                 {
427                     return (this.m_Rotation * 57.29578f);
428                 }
429                 set
430                 {
431                     this.m_Rotation = value * 0.01745329f;
432                 }
433             }
434             public float angularVelocity
435             {
436                 get
437                 {
438                     return (this.m_AngularVelocity * 57.29578f);
439                 }
440                 set
441                 {
442                     this.m_AngularVelocity = value * 0.01745329f;
443                 }
444             }
445             public Color32 color
446             {
447                 get
448                 {
449                     return this.m_Color;
450                 }
451                 set
452                 {
453                     this.m_Color = value;
454                 }
455             }
456             [Obsolete("randomValue property is deprecated. Use randomSeed in
control random behavior of particles.")]
457             public float randomValue
458             {
459                 get
```

```
460                 {
461                     return
BitConverter.ToSingle(BitConverter.GetBytes(this.m_RandomSeed), 0);
462                 }
463                 set
464                 {
465                     this.m_RandomSeed =
BitConverter.ToUInt32(BitConverter.GetBytes(value), 0);
466                 }
467             }
468             public uint randomSeed
469             {
470                 get
471                 {
472                     return this.m_RandomSeed;
473                 }
474                 set
475                 {
476                     this.m_RandomSeed = value;
477                 }
478             }
479         }
480     }
481 }
```

回到顶部(go to top)

## UnityEngine.Behaviour

```
1 namespace UnityEngine
2 {
3     using System;
4     using System.Runtime.CompilerServices;
5
6     public class Behaviour : Component
7     {
8         public bool enabled { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
9     }
10 }
```

回到顶部(g o)

## UnityEngine.Collider

```
 1 namespace UnityEngine
 2 {
 3     using System;
 4     using System.Runtime.CompilerServices;
 5     using System.Runtime.InteropServices;
 6 
 7     public class Collider : Component
 8     {
 9         public Vector3 ClosestPointOnBounds(Vector3 position)
10         {
11             return INTERNAL_CALL_ClosestPointOnBounds(this, ref position);
12         }
13 
14         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
15         private static extern Vector3 INTERNAL_CALL_ClosestPointOnBounds(Collider
self, ref Vector3 position);
16         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
17         private static extern bool INTERNAL_CALL_Internal_Raycast(Collider col, ref
Ray ray, out RaycastHit hitInfo, float distance);
18         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
19         private extern void INTERNAL_get_bounds(out Bounds value);
20         private static bool Internal_Raycast(Collider col, Ray ray, out RaycastHit
hitInfo, float distance)
21         {
22             return INTERNAL_CALL_Internal_Raycast(col, ref ray, out hitInfo,
distance);
23         }
24 
25         public bool Raycast(Ray ray, out RaycastHit hitInfo, float distance)
26         {
27             return Internal_Raycast(this, ray, out hitInfo, distance);
28         }
29 
30         public Rigidbody attachedRigidbody {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
31 
32         public Bounds bounds
33         {
34             get
35             {
36                 Bounds bounds;
37                 this.INTERNAL_get_bounds(out bounds);
38                 return bounds;
39             }
40         }
41 
42         public bool enabled { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
43 
44         public bool isTrigger { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessI
set; }
45 
46         public PhysicMaterial material { [MethodIm                          tern
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.                       essI
set; }
```

```
47
48         public PhysicMaterial sharedMaterial {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
49     }
50 }
```

回到顶部(go to top)

## UnityEngine.Rigidbody

```
1 namespace UnityEngine
2 {
3     using System;
4     using System.Runtime.CompilerServices;
5     using System.Runtime.InteropServices;
6     using UnityEngine.Internal;
7
8     public sealed class Rigidbody : Component
9     {
10         [ExcludeFromDocs]
11         public void AddExplosionForce(float explosionForce, Vector3
explosionPosition, float explosionRadius)
12         {
13             ForceMode force = ForceMode.Force;
14             float upwardsModifier = 0f;
15             INTERNAL_CALL_AddExplosionForce(this, explosionForce, ref
explosionPosition, explosionRadius, upwardsModifier, force);
16         }
17
18         [ExcludeFromDocs]
19         public void AddExplosionForce(float explosionForce, Vector3
explosionPosition, float explosionRadius, float upwardsModifier)
20         {
21             ForceMode force = ForceMode.Force;
22             INTERNAL_CALL_AddExplosionForce(this, explosionForce, ref
explosionPosition, explosionRadius, upwardsModifier, force);
23         }
24
25         public void AddExplosionForce(float explosionForce, Vector3
explosionPosition, float explosionRadius, [DefaultValue("0.0F")] float upwardsModifier,
[DefaultValue("ForceMode.Force")] ForceMode mode)
26         {
27             INTERNAL_CALL_AddExplosionForce(this, explosionForce, ref
explosionPosition, explosionRadius, upwardsModifier, mode);
28         }
29
30         [ExcludeFromDocs]
31         public void AddForce(Vector3 force)
32         {
33             ForceMode mode = ForceMode.Force;
34             INTERNAL_CALL_AddForce(this, ref force, mode);
```

```
35          }
36
37          public void AddForce(Vector3 force, [DefaultValue("ForceMode.Force")]
ForceMode mode)
38          {
39              INTERNAL_CALL_AddForce(this, ref force, mode);
40          }
41
42          [ExcludeFromDocs]
43          public void AddForce(float x, float y, float z)
44          {
45              ForceMode force = ForceMode.Force;
46              this.AddForce(x, y, z, force);
47          }
48
49          public void AddForce(float x, float y, float z,
[DefaultValue("ForceMode.Force")] ForceMode mode)
50          {
51              this.AddForce(new Vector3(x, y, z), mode);
52          }
53
54          [ExcludeFromDocs]
55          public void AddForceAtPosition(Vector3 force, Vector3 position)
56          {
57              ForceMode mode = ForceMode.Force;
58              INTERNAL_CALL_AddForceAtPosition(this, ref force, ref position, mode);
59          }
60
61          public void AddForceAtPosition(Vector3 force, Vector3 position,
[DefaultValue("ForceMode.Force")] ForceMode mode)
62          {
63              INTERNAL_CALL_AddForceAtPosition(this, ref force, ref position, mode);
64          }
65
66          [ExcludeFromDocs]
67          public void AddRelativeForce(Vector3 force)
68          {
69              ForceMode mode = ForceMode.Force;
70              INTERNAL_CALL_AddRelativeForce(this, ref force, mode);
71          }
72
73          public void AddRelativeForce(Vector3 force,
[DefaultValue("ForceMode.Force")] ForceMode mode)
74          {
75              INTERNAL_CALL_AddRelativeForce(this, ref force, mode);
76          }
77
78          [ExcludeFromDocs]
79          public void AddRelativeForce(float x, float y, float z)
80          {
81              ForceMode force = ForceMode.Force;
82              this.AddRelativeForce(x, y, z, force);
83          }
84
85          public void AddRelativeForce(float x, flo
[DefaultValue("ForceMode.Force")] ForceMode mode)
86          {
87              this.AddRelativeForce(new Vector3(x,
```

```
88          }
89
90          [ExcludeFromDocs]
91          public void AddRelativeTorque(Vector3 torque)
92          {
93              ForceMode force = ForceMode.Force;
94              INTERNAL_CALL_AddRelativeTorque(this, ref torque, force);
95          }
96
97          public void AddRelativeTorque(Vector3 torque,
[DefaultValue("ForceMode.Force")] ForceMode mode)
98          {
99              INTERNAL_CALL_AddRelativeTorque(this, ref torque, mode);
100         }
101
102         [ExcludeFromDocs]
103         public void AddRelativeTorque(float x, float y, float z)
104         {
105             ForceMode force = ForceMode.Force;
106             this.AddRelativeTorque(x, y, z, force);
107         }
108
109         public void AddRelativeTorque(float x, float y, float z,
[DefaultValue("ForceMode.Force")] ForceMode mode)
110         {
111             this.AddRelativeTorque(new Vector3(x, y, z), mode);
112         }
113
114         [ExcludeFromDocs]
115         public void AddTorque(Vector3 torque)
116         {
117             ForceMode force = ForceMode.Force;
118             INTERNAL_CALL_AddTorque(this, ref torque, force);
119         }
120
121         public void AddTorque(Vector3 torque, [DefaultValue("ForceMode.Force")]
ForceMode mode)
122         {
123             INTERNAL_CALL_AddTorque(this, ref torque, mode);
124         }
125
126         [ExcludeFromDocs]
127         public void AddTorque(float x, float y, float z)
128         {
129             ForceMode force = ForceMode.Force;
130             this.AddTorque(x, y, z, force);
131         }
132
133         public void AddTorque(float x, float y, float z,
[DefaultValue("ForceMode.Force")] ForceMode mode)
134         {
135             this.AddTorque(new Vector3(x, y, z), mode);
136         }
137
138         public Vector3 ClosestPointOnBounds(Vecto
139         {
140             return INTERNAL_CALL_ClosestPointOnBo                    ;
141         }
```

```
142
143         public Vector3 GetPointVelocity(Vector3 worldPoint)
144         {
145             return INTERNAL_CALL_GetPointVelocity(this, ref worldPoint);
146         }
147
148         public Vector3 GetRelativePointVelocity(Vector3 relativePoint)
149         {
150             return INTERNAL_CALL_GetRelativePointVelocity(this, ref relativePoint);
151         }
152
153         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
154         private static extern void INTERNAL_CALL_AddExplosionForce(Rigidbody self,
float explosionForce, ref Vector3 explosionPosition, float explosionRadius, float
upwardsModifier, ForceMode mode);
155         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
156         private static extern void INTERNAL_CALL_AddForce(Rigidbody self, ref
Vector3 force, ForceMode mode);
157         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
158         private static extern void INTERNAL_CALL_AddForceAtPosition(Rigidbody self,
ref Vector3 force, ref Vector3 position, ForceMode mode);
159         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
160         private static extern void INTERNAL_CALL_AddRelativeForce(Rigidbody self,
ref Vector3 force, ForceMode mode);
161         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
162         private static extern void INTERNAL_CALL_AddRelativeTorque(Rigidbody self,
ref Vector3 torque, ForceMode mode);
163         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
164         private static extern void INTERNAL_CALL_AddTorque(Rigidbody self, ref
Vector3 torque, ForceMode mode);
165         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
166         private static extern Vector3 INTERNAL_CALL_ClosestPointOnBounds(Rigidbody
self, ref Vector3 position);
167         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
168         private static extern Vector3 INTERNAL_CALL_GetPointVelocity(Rigidbody self,
ref Vector3 worldPoint);
169         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
170         private static extern Vector3
INTERNAL_CALL_GetRelativePointVelocity(Rigidbody self, ref Vector3 relativePoint);
171         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
172         private static extern bool INTERNAL_CALL_IsSleeping(Rigidbody self);
173         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
174         private static extern void INTERNAL_CALL_MovePosition(Rigidbody self, ref
Vector3 position);
175         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
176         private static extern void INTERNAL_CALL_MoveRotation(Rigidbody self, ref
Quaternion rot);
177         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
178         private static extern void INTERNAL_CALL_SetDensity(Rigidbody self, float
density);
179         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
180         private static extern void INTERNAL_CALL_Sleep(Rigidbody self);
181         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
182         private static extern bool INTERNAL_CALL_SweepTest(Rigidbody self, r
Vector3 direction, out RaycastHit hitInfo, float dist
183         [MethodImpl(MethodImplOptions.InternalCal
184         private static extern RaycastHit[] INTERN                    gidb
self, ref Vector3 direction, float distance);
```

```
185         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
186         private static extern void INTERNAL_CALL_WakeUp(Rigidbody self);
187         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
188         private extern void INTERNAL_get_angularVelocity(out Vector3 value);
189         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
190         private extern void INTERNAL_get_centerOfMass(out Vector3 value);
191         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
192         private extern void INTERNAL_get_inertiaTensor(out Vector3 value);
193         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
194         private extern void INTERNAL_get_inertiaTensorRotation(out Quaternion
value);
195         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
196         private extern void INTERNAL_get_position(out Vector3 value);
197         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
198         private extern void INTERNAL_get_rotation(out Quaternion value);
199         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
200         private extern void INTERNAL_get_velocity(out Vector3 value);
201         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
202         private extern void INTERNAL_get_worldCenterOfMass(out Vector3 value);
203         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
204         private extern void INTERNAL_set_angularVelocity(ref Vector3 value);
205         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
206         private extern void INTERNAL_set_centerOfMass(ref Vector3 value);
207         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
208         private extern void INTERNAL_set_inertiaTensor(ref Vector3 value);
209         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
210         private extern void INTERNAL_set_inertiaTensorRotation(ref Quaternion
value);
211         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
212         private extern void INTERNAL_set_position(ref Vector3 value);
213         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
214         private extern void INTERNAL_set_rotation(ref Quaternion value);
215         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
216         private extern void INTERNAL_set_velocity(ref Vector3 value);
217         public bool IsSleeping()
218         {
219             return INTERNAL_CALL_IsSleeping(this);
220         }
221
222         public void MovePosition(Vector3 position)
223         {
224             INTERNAL_CALL_MovePosition(this, ref position);
225         }
226
227         public void MoveRotation(Quaternion rot)
228         {
229             INTERNAL_CALL_MoveRotation(this, ref rot);
230         }
231
232         public void SetDensity(float density)
233         {
234             INTERNAL_CALL_SetDensity(this, density);
235         }
236
237         [Obsolete("use Rigidbody.maxAngularVeloci
238         public void SetMaxAngularVelocity(float a
239         {
240             this.maxAngularVelocity = a;
```

```
241         }
242
243         public void Sleep()
244         {
245             INTERNAL_CALL_Sleep(this);
246         }
247
248         [ExcludeFromDocs]
249         public bool SweepTest(Vector3 direction, out RaycastHit hitInfo)
250         {
251             float positiveInfinity = float.PositiveInfinity;
252             return INTERNAL_CALL_SweepTest(this, ref direction, out hitInfo,
positiveInfinity);
253         }
254
255         public bool SweepTest(Vector3 direction, out RaycastHit hitInfo,
[DefaultValue("Mathf.Infinity")] float distance)
256         {
257             return INTERNAL_CALL_SweepTest(this, ref direction, out hitInfo,
distance);
258         }
259
260         [ExcludeFromDocs]
261         public RaycastHit[] SweepTestAll(Vector3 direction)
262         {
263             float positiveInfinity = float.PositiveInfinity;
264             return INTERNAL_CALL_SweepTestAll(this, ref direction,
positiveInfinity);
265         }
266
267         public RaycastHit[] SweepTestAll(Vector3 direction,
[DefaultValue("Mathf.Infinity")] float distance)
268         {
269             return INTERNAL_CALL_SweepTestAll(this, ref direction, distance);
270         }
271
272         public void WakeUp()
273         {
274             INTERNAL_CALL_WakeUp(this);
275         }
276
277         public float angularDrag { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
278
279         public Vector3 angularVelocity
280         {
281             get
282             {
283                 Vector3 vector;
284                 this.INTERNAL_get_angularVelocity(out vector);
285                 return vector;
286             }
287             set
288             {
289                 this.INTERNAL_set_angularVelocity
290             }
291         }
```

```
292
293         public Vector3 centerOfMass
294         {
295             get
296             {
297                 Vector3 vector;
298                 this.INTERNAL_get_centerOfMass(out vector);
299                 return vector;
300             }
301             set
302             {
303                 this.INTERNAL_set_centerOfMass(ref value);
304             }
305         }
306
307         public CollisionDetectionMode collisionDetectionMode {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
308
309         public RigidbodyConstraints constraints {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
310
311         public bool detectCollisions { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
312
313         public float drag { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
314
315         public bool freezeRotation { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
316
317         public Vector3 inertiaTensor
318         {
319             get
320             {
321                 Vector3 vector;
322                 this.INTERNAL_get_inertiaTensor(out vector);
323                 return vector;
324             }
325             set
326             {
327                 this.INTERNAL_set_inertiaTensor(ref value);
328             }
329         }
330
331         public Quaternion inertiaTensorRotation
332         {
333             get
334             {
335                 Quaternion quaternion;
336                 this.INTERNAL_get_inertiaTensorRo
337                 return quaternion;
338             }
339             set
```

```
340             {
341                 this.INTERNAL_set_inertiaTensorRotation(ref value);
342             }
343         }
344 
345         public RigidbodyInterpolation interpolation {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
346 
347         public bool isKinematic { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
348 
349         public float mass { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
350 
351         public float maxAngularVelocity {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
352 
353         public Vector3 position
354         {
355             get
356             {
357                 Vector3 vector;
358                 this.INTERNAL_get_position(out vector);
359                 return vector;
360             }
361             set
362             {
363                 this.INTERNAL_set_position(ref value);
364             }
365         }
366 
367         public Quaternion rotation
368         {
369             get
370             {
371                 Quaternion quaternion;
372                 this.INTERNAL_get_rotation(out quaternion);
373                 return quaternion;
374             }
375             set
376             {
377                 this.INTERNAL_set_rotation(ref value);
378             }
379         }
380 
381         public float sleepAngularVelocity {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
382 
383         public float sleepVelocity { [MethodImpl(MethodImplOptions.InternalC
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.          essI
set; }
384 
385         public int solverIterationCount {
```

```
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
386
387         public bool useConeFriction { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
388
389         public bool useGravity { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
390
391         public Vector3 velocity
392         {
393             get
394             {
395                 Vector3 vector;
396                 this.INTERNAL_get_velocity(out vector);
397                 return vector;
398             }
399             set
400             {
401                 this.INTERNAL_set_velocity(ref value);
402             }
403         }
404
405         public Vector3 worldCenterOfMass
406         {
407             get
408             {
409                 Vector3 vector;
410                 this.INTERNAL_get_worldCenterOfMass(out vector);
411                 return vector;
412             }
413         }
414     }
415 }
```

回到顶部(go to top)

## UnityEngine.AudioListener

```
 1 namespace UnityEngine
 2 {
 3     using System;
 4     using System.Runtime.CompilerServices;
 5
 6     public sealed class AudioListener : Behaviour
 7     {
 8         [Obsolete("GetOutputData returning a float
and pass a pre allocated array instead.")]
 9         public static float[] GetOutputData(int nu
10         {
```

```
11              float[] samples = new float[numSamples];
12              GetOutputDataHelper(samples, channel);
13              return samples;
14          }
15
16          public static void GetOutputData(float[] samples, int channel)
17          {
18              GetOutputDataHelper(samples, channel);
19          }
20
21          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
22          private static extern void GetOutputDataHelper(float[] samples, int channel);
23          [Obsolete("GetSpectrumData returning a float[] is deprecated, use
GetOutputData and pass a pre allocated array instead.")]
24          public static float[] GetSpectrumData(int numSamples, int channel, FFTWindow
window)
25          {
26              float[] samples = new float[numSamples];
27              GetSpectrumDataHelper(samples, channel, window);
28              return samples;
29          }
30
31          public static void GetSpectrumData(float[] samples, int channel, FFTWindow
window)
32          {
33              GetSpectrumDataHelper(samples, channel, window);
34          }
35
36          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
37          private static extern void GetSpectrumDataHelper(float[] samples, int
channel, FFTWindow window);
38
39          public static bool pause { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
40
41          public AudioVelocityUpdateMode velocityUpdateMode {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
42
43          public static float volume { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
44      }
45  }
```

回到顶部(go to top)

## UnityEngine.Camera

```
1  namespace UnityEngine
2  {
```

```
 3     using System;
 4     using System.Runtime.CompilerServices;
 5     using System.Runtime.InteropServices;
 6     using UnityEngine.Internal;
 7
 8     public sealed class Camera : Behaviour
 9     {
10         public Matrix4x4 CalculateObliqueMatrix(Vector4 clipPlane)
11         {
12             return INTERNAL_CALL_CalculateObliqueMatrix(this, ref clipPlane);
13         }
14
15         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
16         public extern void CopyFrom(Camera other);
17         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall,
Obsolete("Camera.DoClear is deprecated and may be removed in the future.")]
18         public extern void DoClear();
19         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
20         public static extern int GetAllCameras(Camera[] cameras);
21         [MethodImpl(MethodImplOptions.InternalCall), Obsolete("use Screen.height
instead."), WrapperlessIcall]
22         public extern float GetScreenHeight();
23         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall, Obsolete("use
Screen.width instead.")]
24         public extern float GetScreenWidth();
25         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
26         private static extern Matrix4x4 INTERNAL_CALL_CalculateObliqueMatrix(Camera
self, ref Vector4 clipPlane);
27         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
28         private static extern void INTERNAL_CALL_ResetAspect(Camera self);
29         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
30         private static extern void INTERNAL_CALL_ResetProjectionMatrix(Camera self);
31         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
32         private static extern void INTERNAL_CALL_ResetReplacementShader(Camera
self);
33         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
34         private static extern void INTERNAL_CALL_ResetWorldToCameraMatrix(Camera
self);
35         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
36         private static extern Ray INTERNAL_CALL_ScreenPointToRay(Camera self, ref
Vector3 position);
37         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
38         private static extern Vector3 INTERNAL_CALL_ScreenToViewportPoint(Camera
self, ref Vector3 position);
39         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
40         private static extern Vector3 INTERNAL_CALL_ScreenToWorldPoint(Camera self,
ref Vector3 position);
41         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
42         private static extern Ray INTERNAL_CALL_ViewportPointToRay(Camera self, ref
Vector3 position);
43         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
44         private static extern Vector3 INTERNAL_CALL_ViewportToScreenPoint(Ca
self, ref Vector3 position);
45         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
46         private static extern Vector3 INTERNAL_CA                    (Cam
self, ref Vector3 position);
47         [MethodImpl(MethodImplOptions.InternalCal
48         private static extern Vector3 INTERNAL_CA
```

```
ref Vector3 position);
 49         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 50         private static extern Vector3 INTERNAL_CALL_WorldToViewportPoint(Camera
self, ref Vector3 position);
 51         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 52         private extern void INTERNAL_get_backgroundColor(out Color value);
 53         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 54         private extern void INTERNAL_get_cameraToWorldMatrix(out Matrix4x4 value);
 55         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 56         private extern void INTERNAL_get_pixelRect(out Rect value);
 57         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 58         private extern void INTERNAL_get_projectionMatrix(out Matrix4x4 value);
 59         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 60         private extern void INTERNAL_get_rect(out Rect value);
 61         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 62         private extern void INTERNAL_get_velocity(out Vector3 value);
 63         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 64         private extern void INTERNAL_get_worldToCameraMatrix(out Matrix4x4 value);
 65         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 66         private extern bool Internal_RenderToCubemapRT(RenderTexture cubemap, int
faceMask);
 67         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 68         private extern bool Internal_RenderToCubemapTexture(Cubemap cubemap, int
faceMask);
 69         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 70         private extern void INTERNAL_set_backgroundColor(ref Color value);
 71         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 72         private extern void INTERNAL_set_pixelRect(ref Rect value);
 73         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 74         private extern void INTERNAL_set_projectionMatrix(ref Matrix4x4 value);
 75         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 76         private extern void INTERNAL_set_rect(ref Rect value);
 77         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 78         private extern void INTERNAL_set_worldToCameraMatrix(ref Matrix4x4 value);
 79         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 80         internal extern bool IsFiltered(GameObject go);
 81         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 82         public extern void Render();
 83         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
 84         public extern void RenderDontRestore();
 85         [ExcludeFromDocs]
 86         public bool RenderToCubemap(Cubemap cubemap)
 87         {
 88             int faceMask = 0x3f;
 89             return this.RenderToCubemap(cubemap, faceMask);
 90         }
 91
 92         [ExcludeFromDocs]
 93         public bool RenderToCubemap(RenderTexture cubemap)
 94         {
 95             int faceMask = 0x3f;
 96             return this.RenderToCubemap(cubemap, faceMask);
 97         }
 98
 99         public bool RenderToCubemap(Cubemap cubem                                  in
faceMask)
100         {
101             return this.Internal_RenderToCubemapTexture(cubemap, faceMask);
```

```
102         }
103
104         public bool RenderToCubemap(RenderTexture cubemap, [DefaultValue("63")] int
faceMask)
105         {
106             return this.Internal_RenderToCubemapRT(cubemap, faceMask);
107         }
108
109         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
110         public extern void RenderWithShader(Shader shader, string replacementTag);
111         public void ResetAspect()
112         {
113             INTERNAL_CALL_ResetAspect(this);
114         }
115
116         public void ResetProjectionMatrix()
117         {
118             INTERNAL_CALL_ResetProjectionMatrix(this);
119         }
120
121         public void ResetReplacementShader()
122         {
123             INTERNAL_CALL_ResetReplacementShader(this);
124         }
125
126         public void ResetWorldToCameraMatrix()
127         {
128             INTERNAL_CALL_ResetWorldToCameraMatrix(this);
129         }
130
131         public Ray ScreenPointToRay(Vector3 position)
132         {
133             return INTERNAL_CALL_ScreenPointToRay(this, ref position);
134         }
135
136         public Vector3 ScreenToViewportPoint(Vector3 position)
137         {
138             return INTERNAL_CALL_ScreenToViewportPoint(this, ref position);
139         }
140
141         public Vector3 ScreenToWorldPoint(Vector3 position)
142         {
143             return INTERNAL_CALL_ScreenToWorldPoint(this, ref position);
144         }
145
146         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
147         public extern void SetReplacementShader(Shader shader, string
replacementTag);
148         public void SetTargetBuffers(RenderBuffer colorBuffer, RenderBuffer
depthBuffer)
149         {
150             this.SetTargetBuffersImpl(out colorBuffer, out depthBuffer);
151         }
152
153         public void SetTargetBuffers(RenderBuffer
depthBuffer)
154         {
155             this.SetTargetBuffersMRTImpl(colorBuffer, out depthBuffer);
```

```
156         }
157
158         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
159         private extern void SetTargetBuffersImpl(out RenderBuffer color, out
RenderBuffer depth);
160         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
161         private extern void SetTargetBuffersMRTImpl(RenderBuffer[] color, out
RenderBuffer depth);
162         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
163         public static extern void SetupCurrent(Camera cur);
164         public Ray ViewportPointToRay(Vector3 position)
165         {
166             return INTERNAL_CALL_ViewportPointToRay(this, ref position);
167         }
168
169         public Vector3 ViewportToScreenPoint(Vector3 position)
170         {
171             return INTERNAL_CALL_ViewportToScreenPoint(this, ref position);
172         }
173
174         public Vector3 ViewportToWorldPoint(Vector3 position)
175         {
176             return INTERNAL_CALL_ViewportToWorldPoint(this, ref position);
177         }
178
179         public Vector3 WorldToScreenPoint(Vector3 position)
180         {
181             return INTERNAL_CALL_WorldToScreenPoint(this, ref position);
182         }
183
184         public Vector3 WorldToViewportPoint(Vector3 position)
185         {
186             return INTERNAL_CALL_WorldToViewportPoint(this, ref position);
187         }
188
189         public RenderingPath actualRenderingPath {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
190
191         public static Camera[] allCameras {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
192
193         public static int allCamerasCount {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
194
195         public float aspect { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
196
197         public Color backgroundColor
198         {
199             get
200             {
201                 Color color;
202                 this.INTERNAL_get_backgroundColor(out color);
203                 return color;
204             }
205             set
206             {
```

```
207                 this.INTERNAL_set_backgroundColor(ref value);
208             }
209         }
210 
211         public Matrix4x4 cameraToWorldMatrix
212         {
213             get
214             {
215                 Matrix4x4 matrixx;
216                 this.INTERNAL_get_cameraToWorldMatrix(out matrixx);
217                 return matrixx;
218             }
219         }
220 
221         public CameraClearFlags clearFlags {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
222 
223         public bool clearStencilAfterLightingPass {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
224 
225         public int cullingMask { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
226 
227         public static Camera current { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
228 
229         public float depth { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
230 
231         public DepthTextureMode depthTextureMode {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
232 
233         public int eventMask { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
234 
235         [Obsolete("use Camera.farClipPlane instead.")]
236         public float far
237         {
238             get
239             {
240                 return this.farClipPlane;
241             }
242             set
243             {
244                 this.farClipPlane = value;
245             }
246         }
247 
248         public float farClipPlane { [MethodImpl(M                    alCa
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.                essI
set; }
249 
```

```
250         public float fieldOfView { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
251
252         [Obsolete("use Camera.fieldOfView instead.")]
253         public float fov
254         {
255             get
256             {
257                 return this.fieldOfView;
258             }
259             set
260             {
261                 this.fieldOfView = value;
262             }
263         }
264
265         public bool hdr { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
266
267         public bool isOrthoGraphic
268         {
269             get
270             {
271                 return this.orthographic;
272             }
273             set
274             {
275                 this.orthographic = value;
276             }
277         }
278
279         public float[] layerCullDistances {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
280
281         public bool layerCullSpherical {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
282
283         public static Camera main { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
284
285         [Obsolete("use Camera.main instead.")]
286         public static Camera mainCamera
287         {
288             get
289             {
290                 return main;
291             }
292         }
293
294         [Obsolete("use Camera.nearClipPlane instead.")]
295         public float near
296         {
297             get
298             {
```

```
299                 return this.nearClipPlane;
300             }
301             set
302             {
303                 this.nearClipPlane = value;
304             }
305         }
306 
307         public float nearClipPlane { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
308 
309         public bool orthographic { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
310 
311         public float orthographicSize { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
312 
313         public float pixelHeight { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
314 
315         public Rect pixelRect
316         {
317             get
318             {
319                 Rect rect;
320                 this.INTERNAL_get_pixelRect(out rect);
321                 return rect;
322             }
323             set
324             {
325                 this.INTERNAL_set_pixelRect(ref value);
326             }
327         }
328 
329         public float pixelWidth { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
330 
331         public Matrix4x4 projectionMatrix
332         {
333             get
334             {
335                 Matrix4x4 matrixx;
336                 this.INTERNAL_get_projectionMatrix(out matrixx);
337                 return matrixx;
338             }
339             set
340             {
341                 this.INTERNAL_set_projectionMatrix(ref value);
342             }
343         }
344 
345         public Rect rect
346         {
347             get
348             {
```

```
349                 Rect rect;
350                 this.INTERNAL_get_rect(out rect);
351                 return rect;
352             }
353             set
354             {
355                 this.INTERNAL_set_rect(ref value);
356             }
357         }
358 
359         public RenderingPath renderingPath {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
360 
361         public float stereoConvergence {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
362 
363         public bool stereoEnabled { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
364 
365         public float stereoSeparation { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
366 
367         public RenderTexture targetTexture {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
368 
369         public TransparencySortMode transparencySortMode {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
370 
371         public bool useOcclusionCulling {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
372 
373         public Vector3 velocity
374         {
375             get
376             {
377                 Vector3 vector;
378                 this.INTERNAL_get_velocity(out vector);
379                 return vector;
380             }
381         }
382 
383         public Matrix4x4 worldToCameraMatrix
384         {
385             get
386             {
387                 Matrix4x4 matrixx;
388                 this.INTERNAL_get_worldToCameraMatrix(out matrixx);
389                 return matrixx;
390             }
391             set
392             {
393                 this.INTERNAL_set_worldToCameraMatrix(ref value);
```

```
394             }
395         }
396     }
397 }
```

回到顶部(go to top)

## UnityEngine.Animator

```
 1 namespace UnityEngine
 2 {
 3     using System;
 4     using System.Runtime.CompilerServices;
 5     using System.Runtime.InteropServices;
 6     using UnityEngine.Internal;
 7 
 8     public sealed class Animator : Behaviour
 9     {
10         [ExcludeFromDocs]
11         public void CrossFade(int stateNameHash, float transitionDuration)
12         {
13             float negativeInfinity = float.NegativeInfinity;
14             int layer = -1;
15             this.CrossFade(stateNameHash, transitionDuration, layer,
negativeInfinity);
16         }
17 
18         [ExcludeFromDocs]
19         public void CrossFade(string stateName, float transitionDuration)
20         {
21             float negativeInfinity = float.NegativeInfinity;
22             int layer = -1;
23             this.CrossFade(stateName, transitionDuration, layer, negativeInfinity);
24         }
25 
26         [ExcludeFromDocs]
27         public void CrossFade(int stateNameHash, float transitionDuration, int
layer)
28         {
29             float negativeInfinity = float.NegativeInfinity;
30             this.CrossFade(stateNameHash, transitionDuration, layer,
negativeInfinity);
31         }
32 
33         [ExcludeFromDocs]
34         public void CrossFade(string stateName, float transitionDuration, in
35         {
36             float negativeInfinity = float.NegativeInfinity;
37             this.CrossFade(stateName, transitionD                    eInf
38         }
39 
40         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
```

```
41          public extern void CrossFade(int stateNameHash, float transitionDuration,
[DefaultValue("-1")] int layer, [DefaultValue("float.NegativeInfinity")] float
normalizedTime);
42          public void CrossFade(string stateName, float transitionDuration,
[DefaultValue("-1")] int layer, [DefaultValue("float.NegativeInfinity")] float
normalizedTime)
43          {
44              this.CrossFade(StringToHash(stateName), transitionDuration, layer,
normalizedTime);
45          }
46
47          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
48          internal extern void EvaluateSM();
49          [Obsolete("ForceStateNormalizedTime is deprecated. Please use Play or
CrossFade instead.")]
50          public void ForceStateNormalizedTime(float normalizedTime)
51          {
52              this.Play(0, 0, normalizedTime);
53          }
54
55          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
56          public extern AnimatorTransitionInfo GetAnimatorTransitionInfo(int
layerIndex);
57          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
58          public extern Transform GetBoneTransform(HumanBodyBones humanBoneId);
59          public bool GetBool(int id)
60          {
61              return this.GetBoolID(id);
62          }
63
64          public bool GetBool(string name)
65          {
66              return this.GetBoolString(name);
67          }
68
69          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
70          private extern bool GetBoolID(int id);
71          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
72          private extern bool GetBoolString(string name);
73          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
74          public extern AnimationInfo[] GetCurrentAnimationClipState(int layerIndex);
75          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
76          public extern AnimatorStateInfo GetCurrentAnimatorStateInfo(int layerIndex);
77          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
78          internal extern string GetCurrentStateName(int layerIndex);
79          public float GetFloat(int id)
80          {
81              return this.GetFloatID(id);
82          }
83
84          public float GetFloat(string name)
85          {
86              return this.GetFloatString(name);
87          }
88
89          [MethodImpl(MethodImplOptions.InternalCal
90          private extern float GetFloatID(int id);
91          [MethodImpl(MethodImplOptions.InternalCal
```

```
92          private extern float GetFloatString(string name);
93          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
94          public extern Vector3 GetIKPosition(AvatarIKGoal goal);
95          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
96          public extern float GetIKPositionWeight(AvatarIKGoal goal);
97          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
98          public extern Quaternion GetIKRotation(AvatarIKGoal goal);
99          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
100         public extern float GetIKRotationWeight(AvatarIKGoal goal);
101         public int GetInteger(int id)
102         {
103             return this.GetIntegerID(id);
104         }
105
106         public int GetInteger(string name)
107         {
108             return this.GetIntegerString(name);
109         }
110
111         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
112         private extern int GetIntegerID(int id);
113         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
114         private extern int GetIntegerString(string name);
115         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
116         public extern string GetLayerName(int layerIndex);
117         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
118         public extern float GetLayerWeight(int layerIndex);
119         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
120         public extern AnimationInfo[] GetNextAnimationClipState(int layerIndex);
121         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
122         public extern AnimatorStateInfo GetNextAnimatorStateInfo(int layerIndex);
123         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
124         internal extern string GetNextStateName(int layerIndex);
125         [Obsolete("GetQuaternion is deprecated.")]
126         public Quaternion GetQuaternion(int id)
127         {
128             return Quaternion.identity;
129         }
130
131         [Obsolete("GetQuaternion is deprecated.")]
132         public Quaternion GetQuaternion(string name)
133         {
134             return Quaternion.identity;
135         }
136
137         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
138         internal extern string GetStats();
139         [Obsolete("GetVector is deprecated.")]
140         public Vector3 GetVector(int id)
141         {
142             return Vector3.zero;
143         }
144
145         [Obsolete("GetVector is deprecated.")]
146         public Vector3 GetVector(string name)
147         {
148             return Vector3.zero;
149         }
```

```
150
151         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
152         private static extern void INTERNAL_CALL_MatchTarget(Animator self, ref
Vector3 matchPosition, ref Quaternion matchRotation, AvatarTarget targetBodyPart, ref
MatchTargetWeightMask weightMask, float startNormalizedTime, float
targetNormalizedTime);
153         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
154         private static extern void INTERNAL_CALL_SetIKPosition(Animator self,
AvatarIKGoal goal, ref Vector3 goalPosition);
155         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
156         private static extern void INTERNAL_CALL_SetIKRotation(Animator self,
AvatarIKGoal goal, ref Quaternion goalRotation);
157         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
158         private static extern void INTERNAL_CALL_SetLookAtPosition(Animator self,
ref Vector3 lookAtPosition);
159         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
160         private extern void INTERNAL_get_bodyPosition(out Vector3 value);
161         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
162         private extern void INTERNAL_get_bodyRotation(out Quaternion value);
163         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
164         private extern void INTERNAL_get_rootPosition(out Vector3 value);
165         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
166         private extern void INTERNAL_get_rootRotation(out Quaternion value);
167         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
168         private extern void INTERNAL_set_bodyPosition(ref Vector3 value);
169         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
170         private extern void INTERNAL_set_bodyRotation(ref Quaternion value);
171         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
172         private extern void INTERNAL_set_rootPosition(ref Vector3 value);
173         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
174         private extern void INTERNAL_set_rootRotation(ref Quaternion value);
175         [ExcludeFromDocs]
176         public void InterruptMatchTarget()
177         {
178             bool completeMatch = true;
179             this.InterruptMatchTarget(completeMatch);
180         }
181
182         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
183         public extern void InterruptMatchTarget([DefaultValue("true")] bool
completeMatch);
184         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
185         internal extern bool IsBoneTransform(Transform transform);
186         [MethodImpl(MethodImplOptions.InternalCall), Obsolete("use mask and layers
to control subset of transfroms in a skeleton", true), WrapperlessIcall]
187         public extern bool IsControlled(Transform transform);
188         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
189         public extern bool IsInTransition(int layerIndex);
190         public bool IsParameterControlledByCurve(int id)
191         {
192             return this.IsParameterControlledByCurveID(id);
193         }
194
195         public bool IsParameterControlledByCurve(string name)
196         {
197             return this.IsParameterControlledByCu
198         }
199
```

```
200         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
201         private extern bool IsParameterControlledByCurveID(int id);
202         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
203         private extern bool IsParameterControlledByCurveString(string name);
204         [ExcludeFromDocs]
205         public void MatchTarget(Vector3 matchPosition, Quaternion matchRotation,
AvatarTarget targetBodyPart, MatchTargetWeightMask weightMask, float
startNormalizedTime)
206         {
207             float targetNormalizedTime = 1f;
208             INTERNAL_CALL_MatchTarget(this, ref matchPosition, ref matchRotation,
targetBodyPart, ref weightMask, startNormalizedTime, targetNormalizedTime);
209         }
210
211         public void MatchTarget(Vector3 matchPosition, Quaternion matchRotation,
AvatarTarget targetBodyPart, MatchTargetWeightMask weightMask, float
startNormalizedTime, [DefaultValue("1")] float targetNormalizedTime)
212         {
213             INTERNAL_CALL_MatchTarget(this, ref matchPosition, ref matchRotation,
targetBodyPart, ref weightMask, startNormalizedTime, targetNormalizedTime);
214         }
215
216         [ExcludeFromDocs]
217         public void Play(int stateNameHash)
218         {
219             float negativeInfinity = float.NegativeInfinity;
220             int layer = -1;
221             this.Play(stateNameHash, layer, negativeInfinity);
222         }
223
224         [ExcludeFromDocs]
225         public void Play(string stateName)
226         {
227             float negativeInfinity = float.NegativeInfinity;
228             int layer = -1;
229             this.Play(stateName, layer, negativeInfinity);
230         }
231
232         [ExcludeFromDocs]
233         public void Play(int stateNameHash, int layer)
234         {
235             float negativeInfinity = float.NegativeInfinity;
236             this.Play(stateNameHash, layer, negativeInfinity);
237         }
238
239         [ExcludeFromDocs]
240         public void Play(string stateName, int layer)
241         {
242             float negativeInfinity = float.NegativeInfinity;
243             this.Play(stateName, layer, negativeInfinity);
244         }
245
246         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
247         public extern void Play(int stateNameHash, [DefaultValue("-1")] int
[DefaultValue("float.NegativeInfinity")] float normal
248         public void Play(string stateName, [Defau                    t,
[DefaultValue("float.NegativeInfinity")] float normal
249         {
```

```
250             this.Play(StringToHash(stateName), layer, normalizedTime);
251         }
252 
253         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
254         public extern void Rebind();
255         public void ResetTrigger(int id)
256         {
257             this.ResetTriggerID(id);
258         }
259 
260         public void ResetTrigger(string name)
261         {
262             this.ResetTriggerString(name);
263         }
264 
265         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
266         private extern void ResetTriggerID(int id);
267         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
268         private extern void ResetTriggerString(string name);
269         public void SetBool(int id, bool value)
270         {
271             this.SetBoolID(id, value);
272         }
273 
274         public void SetBool(string name, bool value)
275         {
276             this.SetBoolString(name, value);
277         }
278 
279         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
280         private extern void SetBoolID(int id, bool value);
281         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
282         private extern void SetBoolString(string name, bool value);
283         public void SetFloat(int id, float value)
284         {
285             this.SetFloatID(id, value);
286         }
287 
288         public void SetFloat(string name, float value)
289         {
290             this.SetFloatString(name, value);
291         }
292 
293         public void SetFloat(int id, float value, float dampTime, float deltaTime)
294         {
295             this.SetFloatIDDamp(id, value, dampTime, deltaTime);
296         }
297 
298         public void SetFloat(string name, float value, float dampTime, float
deltaTime)
299         {
300             this.SetFloatStringDamp(name, value, dampTime, deltaTime);
301         }
302 
303         [MethodImpl(MethodImplOptions.InternalCal
304         private extern void SetFloatID(int id, fl
305         [MethodImpl(MethodImplOptions.InternalCal
306         private extern void SetFloatIDDamp(int id
```

```
float deltaTime);
307         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
308         private extern void SetFloatString(string name, float value);
309         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
310         private extern void SetFloatStringDamp(string name, float value, float
dampTime, float deltaTime);
311         public void SetIKPosition(AvatarIKGoal goal, Vector3 goalPosition)
312         {
313             INTERNAL_CALL_SetIKPosition(this, goal, ref goalPosition);
314         }
315
316         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
317         public extern void SetIKPositionWeight(AvatarIKGoal goal, float value);
318         public void SetIKRotation(AvatarIKGoal goal, Quaternion goalRotation)
319         {
320             INTERNAL_CALL_SetIKRotation(this, goal, ref goalRotation);
321         }
322
323         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
324         public extern void SetIKRotationWeight(AvatarIKGoal goal, float value);
325         public void SetInteger(int id, int value)
326         {
327             this.SetIntegerID(id, value);
328         }
329
330         public void SetInteger(string name, int value)
331         {
332             this.SetIntegerString(name, value);
333         }
334
335         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
336         private extern void SetIntegerID(int id, int value);
337         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
338         private extern void SetIntegerString(string name, int value);
339         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
340         public extern void SetLayerWeight(int layerIndex, float weight);
341         public void SetLookAtPosition(Vector3 lookAtPosition)
342         {
343             INTERNAL_CALL_SetLookAtPosition(this, ref lookAtPosition);
344         }
345
346         [ExcludeFromDocs]
347         public void SetLookAtWeight(float weight)
348         {
349             float clampWeight = 0.5f;
350             float eyesWeight = 0f;
351             float headWeight = 1f;
352             float bodyWeight = 0f;
353             this.SetLookAtWeight(weight, bodyWeight, headWeight, eyesWeight,
clampWeight);
354         }
355
356         [ExcludeFromDocs]
357         public void SetLookAtWeight(float weight, float bodyWeight)
358         {
359             float clampWeight = 0.5f;
360             float eyesWeight = 0f;
361             float headWeight = 1f;
```

```
362             this.SetLookAtWeight(weight, bodyWeight, headWeight, eyesWeight,
clampWeight);
363         }
364
365         [ExcludeFromDocs]
366         public void SetLookAtWeight(float weight, float bodyWeight, float
headWeight)
367         {
368             float clampWeight = 0.5f;
369             float eyesWeight = 0f;
370             this.SetLookAtWeight(weight, bodyWeight, headWeight, eyesWeight,
clampWeight);
371         }
372
373         [ExcludeFromDocs]
374         public void SetLookAtWeight(float weight, float bodyWeight, float
headWeight, float eyesWeight)
375         {
376             float clampWeight = 0.5f;
377             this.SetLookAtWeight(weight, bodyWeight, headWeight, eyesWeight,
clampWeight);
378         }
379
380         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
381         public extern void SetLookAtWeight(float weight, [DefaultValue("0.00f")]
float bodyWeight, [DefaultValue("1.00f")] float headWeight, [DefaultValue("0.00f")]
float eyesWeight, [DefaultValue("0.50f")] float clampWeight);
382         [Obsolete("SetQuaternion is deprecated.")]
383         public void SetQuaternion(int id, Quaternion value)
384         {
385         }
386
387         [Obsolete("SetQuaternion is deprecated.")]
388         public void SetQuaternion(string name, Quaternion value)
389         {
390         }
391
392         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
393         public extern void SetTarget(AvatarTarget targetIndex, float
targetNormalizedTime);
394         public void SetTrigger(int id)
395         {
396             this.SetTriggerID(id);
397         }
398
399         public void SetTrigger(string name)
400         {
401             this.SetTriggerString(name);
402         }
403
404         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
405         private extern void SetTriggerID(int id);
406         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
407         private extern void SetTriggerString(string name);
408         [Obsolete("SetVector is deprecated.")]
409         public void SetVector(int id, Vector3 val
410         {
411         }
```

```
412
413         [Obsolete("SetVector is deprecated.")]
414         public void SetVector(string name, Vector3 value)
415         {
416         }
417
418         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
419         public extern void StartPlayback();
420         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
421         public extern void StartRecording(int frameCount);
422         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
423         public extern void StopPlayback();
424         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
425         public extern void StopRecording();
426         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
427         public static extern int StringToHash(string name);
428         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
429         public extern void Update(float deltaTime);
430         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
431         internal extern void WriteDefaultPose();
432
433         internal bool allowConstantClipSamplingOptimization {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
434
435         [Obsolete("Use AnimationMode.updateMode instead")]
436         public bool animatePhysics
437         {
438             get
439             {
440                 return (this.updateMode == AnimatorUpdateMode.AnimatePhysics);
441             }
442             set
443             {
444                 this.updateMode = !value ? AnimatorUpdateMode.Normal :
AnimatorUpdateMode.AnimatePhysics;
445             }
446         }
447
448         public bool applyRootMotion { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
449
450         public Avatar avatar { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
451
452         internal Transform avatarRoot { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
453
454         public Vector3 bodyPosition
455         {
456             get
457             {
458                 Vector3 vector;
459                 this.INTERNAL_get_bodyPosition(ou
460                 return vector;
461             }
```

```
462             set
463             {
464                 this.INTERNAL_set_bodyPosition(ref value);
465             }
466         }
467 
468         public Quaternion bodyRotation
469         {
470             get
471             {
472                 Quaternion quaternion;
473                 this.INTERNAL_get_bodyRotation(out quaternion);
474                 return quaternion;
475             }
476             set
477             {
478                 this.INTERNAL_set_bodyRotation(ref value);
479             }
480         }
481 
482         public AnimatorCullingMode cullingMode {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
483 
484         public Vector3 deltaPosition { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
485 
486         public Quaternion deltaRotation {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
487 
488         public float feetPivotActive { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
489 
490         public bool fireEvents { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
491 
492         public float gravityWeight { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
493 
494         public bool hasRootMotion { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
495 
496         public bool hasTransformHierarchy {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
497 
498         public float humanScale { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
499 
500         public bool isHuman { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
501 
502         private bool isInManagerList { [MethodImpl(MethodImplOptions.Interna
WrapperlessIcall] get; }
503 
504         public bool isMatchingTarget { [MethodImp                    erna
WrapperlessIcall] get; }
```

```
505
506         public bool isOptimizable { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
507
508         public int layerCount { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
509
510         public bool layersAffectMassCenter {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
511
512         public float leftFeetBottomHeight {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
513
514         public bool logWarnings { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
515
516         public Vector3 pivotPosition { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
517
518         public float pivotWeight { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
519
520         public float playbackTime { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
521
522         public float recorderStartTime {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
523
524         public float recorderStopTime { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
525
526         public float rightFeetBottomHeight {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
527
528         public Vector3 rootPosition
529         {
530             get
531             {
532                 Vector3 vector;
533                 this.INTERNAL_get_rootPosition(out vector);
534                 return vector;
535             }
536             set
537             {
538                 this.INTERNAL_set_rootPosition(ref value);
539             }
540         }
541
542         public Quaternion rootRotation
543         {
544             get
545             {
546                 Quaternion quaternion;
```

```
547                 this.INTERNAL_get_rootRotation(out quaternion);
548                 return quaternion;
549             }
550             set
551             {
552                 this.INTERNAL_set_rootRotation(ref value);
553             }
554         }
555 
556         public RuntimeAnimatorController runtimeAnimatorController {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
557 
558         public float speed { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
559 
560         public bool stabilizeFeet { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
561 
562         internal bool supportsOnAnimatorMove {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
563 
564         public Vector3 targetPosition { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
565 
566         public Quaternion targetRotation {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; }
567 
568         public AnimatorUpdateMode updateMode {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
569     }
570 }
```

回到顶部(go to top)

## UnityEngine.AudioSource

```
 1 namespace UnityEngine
 2 {
 3     using System;
 4     using System.Runtime.CompilerServices;
 5     using UnityEngine.Internal;
 6 
 7     public sealed class AudioSource : Behaviour
 8     {
 9         [Obsolete("GetOutputData return a float[] )utp
passing a pre allocated array instead.")]
10         public float[] GetOutputData(int numSampl
11         {
```

```
12              float[] samples = new float[numSamples];
13              this.GetOutputDataHelper(samples, channel);
14              return samples;
15          }
16
17          public void GetOutputData(float[] samples, int channel)
18          {
19              this.GetOutputDataHelper(samples, channel);
20          }
21
22          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
23          private extern void GetOutputDataHelper(float[] samples, int channel);
24          [Obsolete("GetSpectrumData returning a float[] is deprecated, use
GetSpectrumData passing a pre allocated array instead.")]
25          public float[] GetSpectrumData(int numSamples, int channel, FFTWindow
window)
26          {
27              float[] samples = new float[numSamples];
28              this.GetSpectrumDataHelper(samples, channel, window);
29              return samples;
30          }
31
32          public void GetSpectrumData(float[] samples, int channel, FFTWindow window)
33          {
34              this.GetSpectrumDataHelper(samples, channel, window);
35          }
36
37          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
38          private extern void GetSpectrumDataHelper(float[] samples, int channel,
FFTWindow window);
39          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
40          private static extern void INTERNAL_CALL_Pause(AudioSource self);
41          public void Pause()
42          {
43              INTERNAL_CALL_Pause(this);
44          }
45
46          [ExcludeFromDocs]
47          public void Play()
48          {
49              ulong delay = 0L;
50              this.Play(delay);
51          }
52
53          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
54          public extern void Play([DefaultValue("0")] ulong delay);
55          [ExcludeFromDocs]
56          public static void PlayClipAtPoint(AudioClip clip, Vector3 position)
57          {
58              float volume = 1f;
59              PlayClipAtPoint(clip, position, volume);
60          }
61
62          public static void PlayClipAtPoint(AudioClip clip, Vector3 position,
[DefaultValue("1.0F")] float volume)
63          {
64              GameObject obj2 = new GameObject("One
65                  transform = { position = position }
```

```
66              };
67              AudioSource source = (AudioSource)
obj2.AddComponent(typeof(AudioSource));
68              source.clip = clip;
69              source.volume = volume;
70              source.Play();
71              UnityEngine.Object.Destroy(obj2, clip.length * Time.timeScale);
72          }
73
74          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
75          public extern void PlayDelayed(float delay);
76          [ExcludeFromDocs]
77          public void PlayOneShot(AudioClip clip)
78          {
79              float volumeScale = 1f;
80              this.PlayOneShot(clip, volumeScale);
81          }
82
83          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
84          public extern void PlayOneShot(AudioClip clip, [DefaultValue("1.0F")] float
volumeScale);
85          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
86          public extern void PlayScheduled(double time);
87          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
88          public extern void SetScheduledEndTime(double time);
89          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
90          public extern void SetScheduledStartTime(double time);
91          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
92          public extern void Stop();
93
94          public bool bypassEffects { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
95
96          public bool bypassListenerEffects {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
97
98          public bool bypassReverbZones { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
99
100         public AudioClip clip { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
101
102         public float dopplerLevel { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
103
104         public bool ignoreListenerPause {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
105
106         public bool ignoreListenerVolume {
[MethodImpl(MethodImplOptions.InternalCall), Wrapperl
[MethodImpl(MethodImplOptions.InternalCall), Wrapperl
107
```

```
108         public bool isPlaying { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
109
110         public bool loop { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
111
112         public float maxDistance { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
113
114         [Obsolete("maxVolume is not supported anymore. Use min-, maxDistance and
rolloffMode instead.", true)]
115         public float maxVolume { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
116
117         public float minDistance { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
118
119         [Obsolete("minVolume is not supported anymore. Use min-, maxDistance and
rolloffMode instead.", true)]
120         public float minVolume { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
121
122         public bool mute { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
123
124         public float pan { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
125
126         public float panLevel { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
127
128         public float pitch { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
129
130         public bool playOnAwake { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
131
132         public int priority { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
133
134         [Obsolete("rolloffFactor is not supported anymore. Use min-, maxDist
rolloffMode instead.", true)]
135         public float rolloffFactor { [MethodImpl(MethodImplOptions.InternalC
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.
set; }
136
137         public AudioRolloffMode rolloffMode {
```

```
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
138
139         public float spread { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
140
141         public float time { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
142
143         public int timeSamples { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
144
145         public AudioVelocityUpdateMode velocityUpdateMode {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
146
147         public float volume { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
148     }
149 }
```

回到顶部(go to top)

## UnityEngine.Light

```
 1 namespace UnityEngine
 2 {
 3     using System;
 4     using System.Runtime.CompilerServices;
 5     using System.Runtime.InteropServices;
 6
 7     public sealed class Light : Behaviour
 8     {
 9         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
10         public static extern Light[] GetLights(LightType type, int layer);
11         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
12         private extern void INTERNAL_get_areaSize(out Vector2 value);
13         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
14         private extern void INTERNAL_get_color(out Color value);
15         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
16         private extern void INTERNAL_set_areaSize(ref Vector2 value);
17         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
18         private extern void INTERNAL_set_color(ref Color value);
19
20         public bool alreadyLightmapped {
[MethodImpl(MethodImplOptions.InternalCall), Wrapperl
[MethodImpl(MethodImplOptions.InternalCall), Wrapperl
21
```

```
22          public Vector2 areaSize
23          {
24              get
25              {
26                  Vector2 vector;
27                  this.INTERNAL_get_areaSize(out vector);
28                  return vector;
29              }
30              set
31              {
32                  this.INTERNAL_set_areaSize(ref value);
33              }
34          }
35
36          [Obsolete("light.attenuate was removed; all lights always attenuate now",
true)]
37          public bool attenuate
38          {
39              get
40              {
41                  return true;
42              }
43              set
44              {
45              }
46          }
47
48          public Color color
49          {
50              get
51              {
52                  Color color;
53                  this.INTERNAL_get_color(out color);
54                  return color;
55              }
56              set
57              {
58                  this.INTERNAL_set_color(ref value);
59              }
60          }
61
62          public Texture cookie { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
63
64          public float cookieSize { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
65
66          public int cullingMask { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
67
68          public Flare flare { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.
set; }
69
70          public float intensity { [MethodImpl(Meth
```

```
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
 71
 72         [Obsolete("Use QualitySettings.pixelLightCount instead.")]
 73         public static int pixelLightCount {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
 74
 75         public float range { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
 76
 77         public LightRenderMode renderMode {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
 78
 79         public float shadowBias { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
 80
 81         [Obsolete("light.shadowConstantBias was removed, use light.shadowBias",
true)]
 82         public float shadowConstantBias
 83         {
 84             get
 85             {
 86                 return 0f;
 87             }
 88             set
 89             {
 90             }
 91         }
 92
 93         [Obsolete("light.shadowObjectSizeBias was removed, use light.shadowBias",
true)]
 94         public float shadowObjectSizeBias
 95         {
 96             get
 97             {
 98                 return 0f;
 99             }
100             set
101             {
102             }
103         }
104
105         public LightShadows shadows { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
106
107         public float shadowSoftness { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessI
set; }
108
109         public float shadowSoftnessFade {
[MethodImpl(MethodImplOptions.InternalCall), Wrapperl
[MethodImpl(MethodImplOptions.InternalCall), Wrapperl
110
```

```
111         public float shadowStrength { [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
112
113         public float spotAngle { [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
114
115         public LightType type { [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
116     }
117 }
```

回到顶部(go to top)

## UnityEngine.Animation

```
1  namespace UnityEngine
2  {
3      using System;
4      using System.Collections;
5      using System.Reflection;
6      using System.Runtime.CompilerServices;
7      using System.Runtime.InteropServices;
8      using UnityEngine.Internal;
9
10     public sealed class Animation : Behaviour, IEnumerable
11     {
12         public void AddClip(AnimationClip clip, string newName)
13         {
14             this.AddClip(clip, newName, -2147483648, 0x7fffffff);
15         }
16
17         [ExcludeFromDocs]
18         public void AddClip(AnimationClip clip, string newName, int firstFrame, int lastFrame)
19         {
20             bool addLoopFrame = false;
21             this.AddClip(clip, newName, firstFrame, lastFrame, addLoopFrame);
22         }
23
24         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
25         public extern void AddClip(AnimationClip clip, string newName, int firstFrame, int lastFrame, [DefaultValue("false")] bool addLoopFrame);
26         [ExcludeFromDocs]
27         public void Blend(string animation)
28         {
29             float fadeLength = 0.3f;
30             float targetWeight = 1f;
31             this.Blend(animation, targetWeight, f
32         }
```

```
33
34          [ExcludeFromDocs]
35          public void Blend(string animation, float targetWeight)
36          {
37              float fadeLength = 0.3f;
38              this.Blend(animation, targetWeight, fadeLength);
39          }
40
41          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
42          public extern void Blend(string animation, [DefaultValue("1.0F")] float
targetWeight, [DefaultValue("0.3F")] float fadeLength);
43          [ExcludeFromDocs]
44          public void CrossFade(string animation)
45          {
46              PlayMode stopSameLayer = PlayMode.StopSameLayer;
47              float fadeLength = 0.3f;
48              this.CrossFade(animation, fadeLength, stopSameLayer);
49          }
50
51          [ExcludeFromDocs]
52          public void CrossFade(string animation, float fadeLength)
53          {
54              PlayMode stopSameLayer = PlayMode.StopSameLayer;
55              this.CrossFade(animation, fadeLength, stopSameLayer);
56          }
57
58          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
59          public extern void CrossFade(string animation, [DefaultValue("0.3F")] float
fadeLength, [DefaultValue("PlayMode.StopSameLayer")] PlayMode mode);
60          [ExcludeFromDocs]
61          public AnimationState CrossFadeQueued(string animation)
62          {
63              PlayMode stopSameLayer = PlayMode.StopSameLayer;
64              QueueMode completeOthers = QueueMode.CompleteOthers;
65              float fadeLength = 0.3f;
66              return this.CrossFadeQueued(animation, fadeLength, completeOthers,
stopSameLayer);
67          }
68
69          [ExcludeFromDocs]
70          public AnimationState CrossFadeQueued(string animation, float fadeLength)
71          {
72              PlayMode stopSameLayer = PlayMode.StopSameLayer;
73              QueueMode completeOthers = QueueMode.CompleteOthers;
74              return this.CrossFadeQueued(animation, fadeLength, completeOthers,
stopSameLayer);
75          }
76
77          [ExcludeFromDocs]
78          public AnimationState CrossFadeQueued(string animation, float fadeLength,
QueueMode queue)
79          {
80              PlayMode stopSameLayer = PlayMode.StopSameLayer;
81              return this.CrossFadeQueued(animation, fadeLength, queue,
stopSameLayer);
82          }
83
84          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
```

```
85         public extern AnimationState CrossFadeQueued(string animation,
[DefaultValue("0.3F")] float fadeLength, [DefaultValue("QueueMode.CompleteOthers")]
QueueMode queue, [DefaultValue("PlayMode.StopSameLayer")] PlayMode mode);
86         public AnimationClip GetClip(string name)
87         {
88             AnimationState state = this.GetState(name);
89             if (state != null)
90             {
91                 return state.clip;
92             }
93             return null;
94         }
95
96         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
97         public extern int GetClipCount();
98         public IEnumerator GetEnumerator()
99         {
100            return new Enumerator(this);
101        }
102
103        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
104        internal extern AnimationState GetState(string name);
105        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
106        internal extern AnimationState GetStateAtIndex(int index);
107        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
108        internal extern int GetStateCount();
109        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
110        private static extern void INTERNAL_CALL_Rewind(Animation self);
111        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
112        private static extern void INTERNAL_CALL_Sample(Animation self);
113        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
114        private static extern void INTERNAL_CALL_Stop(Animation self);
115        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
116        private static extern void INTERNAL_CALL_SyncLayer(Animation self, int
layer);
117        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
118        private extern void INTERNAL_get_localBounds(out Bounds value);
119        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
120        private extern void Internal_RewindByName(string name);
121        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
122        private extern void INTERNAL_set_localBounds(ref Bounds value);
123        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
124        private extern void Internal_StopByName(string name);
125        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
126        public extern bool IsPlaying(string name);
127        [ExcludeFromDocs]
128        public bool Play()
129        {
130            PlayMode stopSameLayer = PlayMode.StopSameLayer;
131            return this.Play(stopSameLayer);
132        }
133
134        [ExcludeFromDocs]
135        public bool Play(string animation)
136        {
137            PlayMode stopSameLayer = PlayMode.Sto
138            return this.Play(animation, stopSameL
139        }
```

```
140
141         [Obsolete("use PlayMode instead of AnimationPlayMode.")]
142         public bool Play(AnimationPlayMode mode)
143         {
144             return this.PlayDefaultAnimation((PlayMode) mode);
145         }
146
147         public bool Play([DefaultValue("PlayMode.StopSameLayer")] PlayMode mode)
148         {
149             return this.PlayDefaultAnimation(mode);
150         }
151
152         [Obsolete("use PlayMode instead of AnimationPlayMode.")]
153         public bool Play(string animation, AnimationPlayMode mode)
154         {
155             return this.Play(animation, (PlayMode) mode);
156         }
157
158         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
159         public extern bool Play(string animation,
[DefaultValue("PlayMode.StopSameLayer")] PlayMode mode);
160         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
161         private extern bool PlayDefaultAnimation(PlayMode mode);
162         [ExcludeFromDocs]
163         public AnimationState PlayQueued(string animation)
164         {
165             PlayMode stopSameLayer = PlayMode.StopSameLayer;
166             QueueMode completeOthers = QueueMode.CompleteOthers;
167             return this.PlayQueued(animation, completeOthers, stopSameLayer);
168         }
169
170         [ExcludeFromDocs]
171         public AnimationState PlayQueued(string animation, QueueMode queue)
172         {
173             PlayMode stopSameLayer = PlayMode.StopSameLayer;
174             return this.PlayQueued(animation, queue, stopSameLayer);
175         }
176
177         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
178         public extern AnimationState PlayQueued(string animation,
[DefaultValue("QueueMode.CompleteOthers")] QueueMode queue,
[DefaultValue("PlayMode.StopSameLayer")] PlayMode mode);
179         public void RemoveClip(string clipName)
180         {
181             this.RemoveClip2(clipName);
182         }
183
184         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
185         public extern void RemoveClip(AnimationClip clip);
186         [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
187         private extern void RemoveClip2(string clipName);
188         public void Rewind()
189         {
190             INTERNAL_CALL_Rewind(this);
191         }
192
193         public void Rewind(string name)
194         {
```

```
195             this.Internal_RewindByName(name);
196         }
197 
198         public void Sample()
199         {
200             INTERNAL_CALL_Sample(this);
201         }
202 
203         public void Stop()
204         {
205             INTERNAL_CALL_Stop(this);
206         }
207 
208         public void Stop(string name)
209         {
210             this.Internal_StopByName(name);
211         }
212 
213         public void SyncLayer(int layer)
214         {
215             INTERNAL_CALL_SyncLayer(this, layer);
216         }
217 
218         [Obsolete("Use cullingType instead")]
219         public bool animateOnlyIfVisible {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
220 
221         public bool animatePhysics { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
222 
223         public AnimationClip clip { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
224 
225         public AnimationCullingType cullingType {
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] get;
[MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall] set; }
226 
227         public bool isPlaying { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; }
228 
229         public AnimationState this[string name]
230         {
231             get
232             {
233                 return this.GetState(name);
234             }
235         }
236 
237         public Bounds localBounds
238         {
239             get
240             {
241                 Bounds bounds;
242                 this.INTERNAL_get_localBounds(out
243                 return bounds;
```

```
244             }
245             set
246             {
247                 this.INTERNAL_set_localBounds(ref value);
248             }
249         }
250 
251         public bool playAutomatically { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
252 
253         public WrapMode wrapMode { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
254 
255         private sealed class Enumerator : IEnumerator
256         {
257             private int m_CurrentIndex = -1;
258             private Animation m_Outer;
259 
260             internal Enumerator(Animation outer)
261             {
262                 this.m_Outer = outer;
263             }
264 
265             public bool MoveNext()
266             {
267                 int stateCount = this.m_Outer.GetStateCount();
268                 this.m_CurrentIndex++;
269                 return (this.m_CurrentIndex < stateCount);
270             }
271 
272             public void Reset()
273             {
274                 this.m_CurrentIndex = -1;
275             }
276 
277             public object Current
278             {
279                 get
280                 {
281                     return this.m_Outer.GetStateAtIndex(this.m_CurrentIndex);
282                 }
283             }
284         }
285     }
286 }
```

回到顶部(g o)

## UnityEngine.MonoBehaviour

```
namespace UnityEngine
{
    using System;
    using System.Collections;
    using System.Runtime.CompilerServices;
    using UnityEngine.Internal;

    public class MonoBehaviour : Behaviour
    {
        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
        public extern MonoBehaviour();
        public void CancelInvoke()
        {
            this.Internal_CancelInvokeAll();
        }

        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
        public extern void CancelInvoke(string methodName);
        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
        private extern void Internal_CancelInvokeAll();
        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
        private extern bool Internal_IsInvokingAll();
        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
        public extern void Invoke(string methodName, float time);
        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
        public extern void InvokeRepeating(string methodName, float time, float
repeatRate);
        public bool IsInvoking()
        {
            return this.Internal_IsInvokingAll();
        }

        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
        public extern bool IsInvoking(string methodName);
        public static void print(object message)
        {
            Debug.Log(message);
        }

        public Coroutine StartCoroutine(IEnumerator routine)
        {
            return this.StartCoroutine_Auto(routine);
        }

        [ExcludeFromDocs]
        public Coroutine StartCoroutine(string methodName)
        {
            object obj2 = null;
            return this.StartCoroutine(methodName, obj2);
        }

        [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
        public extern Coroutine StartCoroutine(string methodName,
[DefaultValue("null")] object value);
        [MethodImpl(MethodImplOptions.InternalCall
        public extern Coroutine StartCoroutine_Aut
        [MethodImpl(MethodImplOptions.InternalCall
        public extern void StopAllCoroutines();
```

```
57          public void StopCoroutine(IEnumerator routine)
58          {
59              this.StopCoroutineViaEnumerator_Auto(routine);
60          }
61
62          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
63          public extern void StopCoroutine(string methodName);
64          public void StopCoroutine(Coroutine routine)
65          {
66              this.StopCoroutine_Auto(routine);
67          }
68
69          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
70          internal extern void StopCoroutine_Auto(Coroutine routine);
71          [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
72          internal extern void StopCoroutineViaEnumerator_Auto(IEnumerator routine);
73
74          public bool useGUILayout { [MethodImpl(MethodImplOptions.InternalCall),
WrapperlessIcall] get; [MethodImpl(MethodImplOptions.InternalCall), WrapperlessIcall]
set; }
75      }
76 }
```

回到顶部(go to top)

## 总结

如果您愿意花几块钱请我喝杯茶的话，可以用手机扫描下方的二维码，通过微信捐赠。（微信捐助不显示您的信息，如需要，请注明您的联系方式）我会努力写出更好的文章。
Thank you for your kindly donation!

微信捐赠二维码：
Donate by microMsg:

标签：Unity3D



BIT祝威
关注 - 12
粉丝 - 624
荣誉：推荐博客
+加关注

« 上一篇：自制Unity小游戏TankHero-2D(3)开始玩起来
» 下一篇：自制Unity小游戏TankHero-2D(4)关卡+小地图图标+碰撞条件分析

**评论列表**

#1楼 2015-04-23 10:36 Good_good

这些源码怎么在Unity3D查看到呢？

支持(0) 反对(0)

#2楼[楼主] 2015-04-23 17:47 BIT祝威

@ Good_good
引用
这些源码怎么在Unity3D查看到呢？

reflector

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

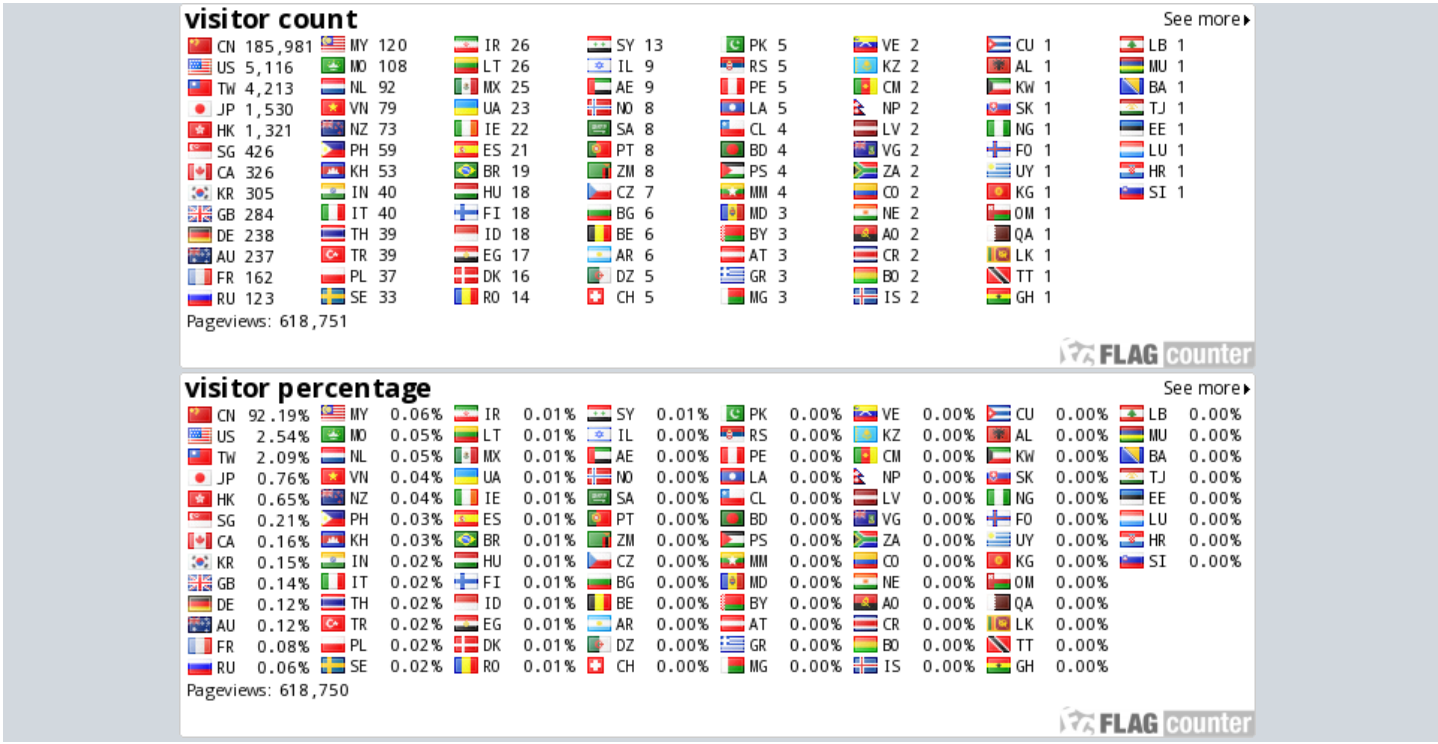
**注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。**



**历史上的今天:**
2014-02-07 【OpenGL(SharpGL)】支持任意相机可平移缩放的轨迹球实现
2013-02-07 Winform游戏编程入门1:游戏循环的演化



Large Visitor Globe


visitor count
See more
CN 185,981 MY 120 IR 26 SY 13 PK 5 VE 2 CU 1 LB 1
US 5,116 MD 108 LT 26 IL 9 RS 5 KZ 2 AL 1 MU 1
TW 4,213 NL 92 MX 25 AE 9 PE 5 CM 2 KW 1 BA 1
JP 1,530 VN 79 UA 23 NO 8 LA 5 NP 2 SK 1 TJ 1
HK 1,321 NZ 73 IE 22 SA 8 CL 4 LV 2 NG 1 EE 1
SG 426 PH 59 ES 21 PT 8 BD 4 VG 2 FO 1 LU 1
CA 326 KH 53 BR 19 ZM 8 PS 4 ZA 2 UY 1 HR 1
KR 305 IN 40 HU 18 CZ 7 MM 4 CO 2 KG 1 SI 1
GB 284 IT 40 FI 18 BG 6 MD 3 NE 2 OM 1
DE 238 TH 39 ID 18 BE 6 BY 3 AO 2 QA 1
AU 237 TR 39 EG 17 AR 6 AT 3 CR 2 LK 1
FR 162 PL 37 DK 16 DZ 5 GR 3 BO 2 TT 1
RU 123 SE 33 RO 14 CH 5 MG 3 IS 2 GH 1
Pageviews: 618,751
FLAG counter
visitor percentage
See more
CN 92.19% MY 0.06% IR 0.01% SY 0.01% PK 0.00% VE 0.00% CU 0.00% LB 0.00%
US 2.54% MD 0.05% LT 0.01% IL 0.00% RS 0.00% KZ 0.00% AL 0.00% MU 0.00%
TW 2.09% NL 0.05% MX 0.01% AE 0.00% PE 0.00% CM 0.00% KW 0.00% BA 0.00%
JP 0.76% VN 0.04% UA 0.01% NO 0.00% LA 0.00% NP 0.00% SK 0.00% TJ 0.00%
HK 0.65% NZ 0.04% IE 0.01% SA 0.00% CL 0.00% LV 0.00% NG 0.00% EE 0.00%
SG 0.21% PH 0.03% ES 0.01% PT 0.00% BD 0.00% VG 0.00% FO 0.00% LU 0.00%
CA 0.16% KH 0.03% BR 0.01% ZM 0.00% PS 0.00% ZA 0.00% UY 0.00% HR 0.00%
KR 0.15% IN 0.02% HU 0.01% CZ 0.00% MM 0.00% CO 0.00% KG 0.00% SI 0.00%
GB 0.14% IT 0.02% FI 0.01% BG 0.00% MD 0.00% NE 0.00% OM 0.00%
DE 0.12% TH 0.02% ID 0.01% BE 0.00% BY 0.00% AO 0.00% QA 0.00%
AU 0.12% TR 0.02% EG 0.01% AR 0.00% AT 0.00% CR 0.00% LK 0.00%
FR 0.08% PL 0.02% DK 0.01% DZ 0.00% GR 0.00% BO 0.00% TT 0.00%
RU 0.06% SE 0.02% RO 0.01% CH 0.00% MG 0.00% IS 0.00% GH 0.00%
Pageviews: 618,750
FLAG counter