|  |  |
| --- | --- |
| 产品名称API Release Notes |  |
|  |  |
| 产品版本API Release Notes | Total 29 pages 共 29页 |
|  |

API Release Notes

|  |  |  |  |
| --- | --- | --- | --- |
| Prepared by  拟制 | Unity内容组 | Date  日期 | 2020/04/03 |
| Reviewed by  审核 |  | Date  日期 |  |
| Approved by  批准 |  | Date  日期 |  |



Shadow Creator Information Technology Co.,Ltd.

上海影创信息科技有限公司

All rights reserved

版权所有 侵权必究

Revision Record 修订记录

| Date  日期 | Revision Version  修订 版本 | Release Notes | Change Description  修改描述 | Author  作者 |
| --- | --- | --- | --- | --- |
| 2020/04/03 | V1.0.0 | API Release Notes | 初稿完成 | 王超群 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Catalog 目 录

1 前言 4

1.1 说明 4

2 SVR模块API\_SVR 4

3 InputSystem模块API\_InputSystem 8

4 InputSystem模块API\_InputSystem\_Head 12

5 InputSystem模块API\_InputSystem\_Bluetooth 15

6 InputSystem模块API\_InputSystem\_Gesture26Dof 22

# 

# 前言

## 说明

**a)**:API主要以脚本提供，API脚本命名规则：

API\_模块\_子模块\_备注.cs

其中：子模块和备注可无,例如：API\_SVR.cs或 API\_InputSystem\_Head.cs

**b)**:API主要提供的方法都是static，直接用ClassName.APIName即可调用

API以是属性或方法提供，其上///所示内容为API说明，例如：

/// <summary>

/// 设置眼镜进入模式,运行过程中可修改

/// </summary>

public static void SetTrackMode(TrackMode mode) {

if(SvrManager.Instance != null) {

SvrManager.Instance.settings.trackPosition = (mode == TrackMode.Mode\_6Dof ? true : false);

}

}

/// <summary>

/// 获取InputSystem支持的头部输入设备,头部输入设备包含一个Part，名曰：Head

/// </summary>

/// <returns>null表示不支持或者InputSystem未初始好</returns>

public static InputDeviceHead HeadDevice {

get {

if(InputSystem.Instant) {

return InputSystem.Instant.Head;

}

return null;

}

}

**C)**:每个API具有全局唯一编号

编号格式：API-No.X

备注：X以1开始，整数增加,每个模块X的范围已定，见如下

API\_SVR.cs X范围1-50

API\_InputSystem.cs X范围50-100

API\_InputSystem\_Head.cs X范围100-150

API\_InputSystem\_Bluetooth.cs X范围150-200

API\_InputSystem\_Gesture26Dof.cs X范围200-250

# SVR模块API\_SVR

## API-No.1

public static void SetTrackMode(TrackMode mode);

**说明：**

设置系统6DOF与3DOF模式

**参数：**

TrackMode.Mode\_3Dof表示3Dof模式

TrackMode.Mode\_6Dof表示6Dof模式

**返回值：**

无

**备注：**

此接口可在系统运行中动态设置

**所属：**

API\_SVR.cs

## API-No.2

public static bool IsSvrRunning();

**说明：**

获取系统运行状态

**参数：**

无

**返回值：**

True表示正在运行，系统休眠时返回false

**备注：**

无

**所属：**

API\_SVR.cs

## API-No.3

public static bool IsSvrInitialized();

**说明：**

获取系统是否初始化完成

**参数：**

无

**返回值：**

True表示系统初始化完成，FALSE表示系统初始化未完成

**备注：**

无

**所属：**

API\_SVR.cs

## API-No.4

public static void AddInitializedCallBack(Action action);

**说明：**

系统初始化完成后触发的回调

**参数：**

类型为Action的委托

**返回值：**

无

**备注：**

无

**所属：**

API\_SVR.cs

## API-No.5

public static void RemoveInitializedCallBack(Action action);

**说明：**

删除系统初始化完成的回调

**参数：**

类型为Action的委托

**返回值：**

无

**备注：**

无

**所属：**

API\_SVR.cs

## API-No.6

public static void SetRenderFrame(int frameRate = -1);

**说明：**

设置系统渲染帧率，眼镜端默认以75FPS运行

**参数：**

默认-1表示按系统Vsync频率渲染

**返回值：**

无

**备注：**

支持动态设置

**所属：**

API\_SVR.cs

## API-No.7

public static List<Camera> GetEyeCameras();

**说明：**

获取双眼对应的Camera

**参数：**

无

**返回值：**

List[0]表示左眼，List[1]表示右眼，注意有可能为null

**备注：**

无

**所属：**

API\_SVR.cs

## API-No.8

public static List<RenderTexture> GetRenderTexure();

**说明：**

获取双眼渲染出来的RenderTexture

**参数：**

无

**返回值：**

List[0]表示左眼，List[1]表示右眼，注意有可能为null

**备注：**

无

**所属：**

API\_SVR.cs

## API-No.9

public static Transform GetHead();

**说明：**

获取头部对应的组件

**参数：**

无

**返回值：**

头部对应的组件

**备注：**

无

**所属：**

API\_SVR.cs

## API-No.10

public static void SetPD(float offset = 0);

**说明：**

设置双眼瞳距

**参数：**

增加的偏移

**返回值：**

无

**备注：**

Awake时调用，Start后调用不保证有效

**所属：**

API\_SVR.cs

## API-No.11

public static void RecenterTracking();

**说明：**

重置定位

**参数：**

无

**返回值：**

无

**备注：**

可动态设置

**所属：**

API\_SVR.cs

# InputSystem模块API\_InputSystem

## API-No.50

public static InputSystem GetInstant();

**说明：**

获取InputSystem的单例，所有输入的管理类

**参数：**

无

**返回值：**

InputSystem的单例，注意有可能为null

**备注：**

无

**所属：**

API\_InputSystem.cs

## API-No.51

public static bool IsISInitialized();

**说明：**

InputSystem是否初始化完成

**参数：**

无

**返回值：**

True表初始化完成，False表示初始化未完成

**备注：**

无

**所属：**

API\_InputSystem.cs

## API-No.52

public static void AddInitializedCallBack(Action action);

**说明：**

InputSystem初始化完成时触发的回调

**参数：**

类型Action的回调

**返回值：**

无

**备注：**

注意需要和RemoveInitializedCallBack保持成对出现

**所属：**

API\_InputSystem.cs

## API-No.53

public static RemoveInitializedCallBack(Action action);

**说明：**

删除InputSystem初始化完成时的回调

**参数：**

类型Action的回调

**返回值：**

无

**备注：**

注意需要和AddInitializedCallBack保持成对出现

**所属：**

API\_InputSystem.cs

## API-No.54

public static void EnableInputDeivce(InputDeviceType inputDevice);

**说明：**

开启InputSystem中的某个设备

**参数：**

开启设备，支持设备见InputDeviceType

InputDeviceType.Head:Head组件

InputDeviceType.HandShank :手柄输入设备

InputDeviceType.Gesture26DofHand: 自由手势输入设备

**返回值：**

无

**备注：**

无

**所属：**

API\_InputSystem.cs

## API-No.55

public static void DisableInputDeivce(InputDeviceType inputDevice);

**说明：**

关闭InputSystem中的某个设备

**参数：**

关闭设备，支持设备见InputDeviceType

InputDeviceType.Head:Head组件

InputDeviceType.HandShank :手柄输入设备

InputDeviceType.Gesture26DofHand: 自由手势输入设备

**返回值：**

无

**备注：**

无

**所属：**

API\_InputSystem.cs

## API-No.56

继承PointerHandlers类,重写方法

**说明：**

监听某个按键事件接口

**参数：**

无

**返回值：**

无

**备注：**

引用命名空间： SC.InputSystem ;

支持的事件有：

Enter按键按下：OnSCPointerDown / OnSCPointerEnter

Enter按键松开：OnSCPointerUp / OnSCPointerExit

Enter键按下并拖拽：OnSCPointerDrag

两种输入设备的Enter键同时按下：OnSCBothHandPointerDown

两种输入设备的Enter键同时松开： OnSCBothHandPointerUp

两种输入设备的Enter键同时按下并拖拽：OnSCBothHandPointerDrag

在需要监听的事件的游戏对象上加上Collider并override相应方法即可

**所属：**

API\_InputSystem.cs

## API-No.57

继承PointerDelegate类,重写方法

**说明：**

监听某个按键事件的委托

**参数：**

无

**返回值：**

无

**备注：**

引用命名空间： SC.InputSystem ;

支持的事件有：

Enter按键按下：partEnterKeyDownDelegate

Enter按键松开：partEnterKeyUpDelegate

Enter键按下并拖拽：partEnterKeyDragDelegate

任意键按下：partAnyKeyDownDelegate

任意键长按（只有Bluetooth输入设备支持）：partAnyKeyLongDelegate

任意键松开：partAnyKeyUpDelegate

在需要监听的事件的游戏对象上override相应方法即可

**所属：**

API\_ InputSystem.cs

## API-No.58

public static GameObject Target;

**说明：**

输入设备检测的目标，优先级为Head/BTRight/BTLeft/GTRight/GTLeft

**参数：**

无

**返回值：**

返回输入设备检测的目标组件，目标有可能为null

**备注：**

无

**所属：**

API\_InputSystem.cs

## API-No.59

public static GameObject Gazer;

**说明：**

输入设备的射线起点，优先级为Head/BTRight/BTLeft/GTRight/GTLeft

**参数：**

无

**返回值：**

返回输入设备的射线起点的组件

**备注：**

无

**所属：**

API\_InputSystem.cs

## API-No.60

public static Vector3 Normal;

**说明：**

输入设备的射线方向，优先级为Head/BTRight/BTLeft/GTRight/GTLeft

**参数：**

无

**返回值：**

返回输入设备的射线方向

**备注：**

无

**所属：**

API\_InputSystem.cs

## API-No.61

public static Vector3 Position;

**说明：**

输入设备Focus的位置，优先级为Head/BTRight/BTLeft/GTRight/GTLeft

**参数：**

无

**返回值：**

返回输入设备Focus的位置，全局坐标

**备注：**

无

**所属：**

API\_InputSystem.cs

## API-No.62

public static InputDevicePartBase InputDeviceCurrent;

**说明：**

获取当前输入设备，优先级为Head/BTRight/BTLeft/GTRight/GTLeft

**参数：**

无

**返回值：**

返回当前输入设备组件

**备注：**

无

**所属：**

API\_InputSystem.cs

# InputSystem模块API\_InputSystem\_Head

## API-No.100

public static InputDeviceHead HeadDevice;

**说明：**

获取Head的输入设备管理类

**参数：**

无

**返回值：**

Head组件，注意有可能为null

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.101

public static InputDeviceHeadPart Head;

**说明：**

获取Head组件

**参数：**

无

**返回值：**

返回Head组件，注意有可能为null

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.102

public static Vector3 HeadEulerAngles;

**说明：**

获取Head的欧拉角

**参数：**

无

**返回值：**

Head的欧拉角，全局坐标

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.103

public static Vector3 HeadPosition;

**说明：**

获取Head组件的位置

**参数：**

无

**返回值：**

Head组件的位置，全局坐标

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.104

public static SCPointEventData HeadPointerEventData;

**说明：**

获取Head组件碰撞的信息集合

**参数：**

无

**返回值：**

碰撞的信息集合，注意有可能为null

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.105

public static GameObject HeadHitTarget;

**说明：**

获取Head组件碰撞的物体

**参数：**

无

**返回值：**

空表示未有碰撞物体

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.106

public static RaycastHit HeadHitInfo;

**说明：**

获取Head组件碰撞信息

**参数：**

无

**返回值：**

碰撞信息

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.107

public static GameObject HeadDragTarget;

**说明：**

获取Head组件拖拽的物体

**参数：**

无

**返回值：**

推拽的物体，注意有可能为null

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.108

public static bool IsHeadKeyDown(InputKeyCode inputKeyCode);

**说明：**

获取Head组件下某个按键是否Down，当前帧有效，下帧复位

**参数：**

指定按键，Head支持指定的按键为Enter/Back

**返回值：**

False表示未Down，true表示Down

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.109

public static bool IsHeadKeyUp(InputKeyCode inputKeyCode);

**说明：**

获取Head组件下某个按键是否Up，当前帧有效，下帧复位

**参数：**

指定按键，Head支持指定的按键为Enter/Back

**返回值：**

False表示未Up，true表示Up

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.110

public static InputKeyState HeadKeyState(InputKeyCode inputKeyCode);

**说明：**

获取Head组件下某个按键的状态，当前帧有效，下帧复位

**参数：**

指定按键，Head支持指定的按键为Enter/Back

**返回值：**

指定按键的状态

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.111

public static InputKeyState HeadKeyCurrentState(InputKeyCode inputKeyCode);

**说明：**

获取Head组件下某个按键的实时状态，参考UnityAPI:Input.GetKey

**参数：**

指定按键，Head支持指定的按键为Enter/Back

**返回值：**

指定按键的状态

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.112

public static void HeadAddKey(InputKeyCode inputKeyCode, InputKeyState inputKeyState);

**说明：**

给Head组件模拟发送一个按键

**参数：**

inputKeyCode：按键

inputKeyState：按键状态（DOWN、UP、LONG和NULL）

**返回值：**

无

**备注：**

发送按键后，下一帧生效，注意发送按键最好需要Down/Up成对出现

**所属：**

API\_InputSystem\_Head.cs

## API-No.113

public static void SetHeadRayCastDistance(float distance);

**说明：**

设置Head组件的检测Collider的范围半径，默认50米

**参数：**

检测Collider的范围半径，单位米

**返回值：**

无

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.114

public static void SetHeadEndPointerDistance(float distance);

**说明：**

设置Head组件的Focus未碰撞到Collider时的距离，默认3米

**参数：**

Focus未碰撞到Collider时的距离，单位米

**返回值：**

无

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

## API-No.115

public static Focus GetHeadFocus;

**说明：**

获取Head组件的Focus光标对象

**参数：**

无

**返回值：**

Head组件的Focus光标对象

**备注：**

无

**所属：**

API\_InputSystem\_Head.cs

# InputSystem模块API\_InputSystem\_Bluetooth

## API-No.150

public static InputDeviceHandShank BTDevice;

**说明：**

获取手柄输入设备管理类

**参数：**

无

**返回值：**

手柄输入设备，注意有可能为null

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.151

public static InputDeviceHandShankPart BTRight;

**说明：**

获取第一个连接的手柄组件

**参数：**

无

**返回值：**

第一个连接的手柄组件，注意有可能为null

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.152

public static InputDeviceHandShankPart BTLeft;

**说明：**

获取第二个连接的手柄组件

**参数：**

无

**返回值：**

第二个连接的手柄组件，注意有可能为null

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.153

public static Quaternion BTRotation(BTType type = BTType.Right);

**说明：**

获取手柄组件的旋转

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

手柄组件的旋转值的四元素，全局坐标

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.154

public static Vector3 BTPosition(BTType type = BTType.Right);

**说明：**

获取手柄组件的位置

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

手柄组件的位置，全局坐标

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.155

public static bool IsBTTpTouch(BTType type = BTType.Right);

**说明：**

获取手柄组件的是否触摸了TouchPanel

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

True表示触摸，false表示未触摸

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.156

public static Vector2 BTTpTouchInfo(BTType type = BTType.Right);

**说明：**

获取手柄组件的触摸的位置信息（x,y）

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

手柄组件的触摸的位置信息（x,y）

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.157

public static string BTName(BTType type = BTType.Right);

**说明：**

获取手柄组件的名字

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

手柄组件的名字，K02或K07

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.158

public static SCPointEventData BTPointerEventData(BTType type = BTType.Right);

**说明：**

获取手柄组件的碰撞信息集合

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

手柄组件的碰撞信息集合，注意有可能为null

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.159

public static GameObject BTHitTarget(BTType type = BTType.Right);

**说明：**

获取手柄组件碰撞的Collider物体

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

返回手柄组件碰撞的Collider物体

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.160

public static RaycastHit BTHitInfo(BTType type = BTType.Right);

**说明：**

获取手柄组件碰撞的Collider具体信息

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

手柄组件碰撞的Collider具体信息

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.161

public static GameObject BTDragTarget(BTType type = BTType.Right);

**说明：**

获取手柄组件拖拽的物体

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

手柄组件拖拽的物体

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.162

public static bool IsBTKeyDown(InputKeyCode inputKeyCode, BTType type = BTType.Right);

**说明：**

获取手柄组件是否按下某键

**参数：**

inputKeyCode：指定按键，HandShank支持Enter(Trigger)/Back/Funcion/Tp

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

True表示按下，false表示未按下

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.163

public static bool IsBTKeyUp(InputKeyCode inputKeyCode, BTType type = BTType.Right);

**说明：**

获取手柄组件是否触发某个Key Up

**参数：**

inputKeyCode：指定按键，HandShank支持Enter(Trigger)/Back/Funcion/Tp

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

True表示触发，false表示未触发

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.164

public static InputKeyState BTKeyState(InputKeyCode inputKeyCode, BTType type = BTType.Right);

**说明：**

获取type手柄组件inputKeyCode键的状态

**参数：**

inputKeyCode：指定按键，HandShank支持Enter(Trigger)/Back/Funcion/Tp

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

手柄组件inputKeyCode键的状态

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.165

public static InputKeyState BTKeyCurrentState(InputKeyCode inputKeyCode, BTType type = BTType.Right);

**说明：**

获取手柄组件inputKeyCode键的实时状态

**参数：**

inputKeyCode：指定按键，HandShank支持Enter(Trigger)/Back/Funcion/Tp

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

手柄组件inputKeyCode键的实时状态

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.166

public static void BTKeyAddKey(InputKeyCode inputKeyCode, InputKeyState inputKeyState, BTType type = BTType.Right);

**说明：**

模拟给手柄组件发送一个按键

**参数：**

inputKeyCode：指定按键，HandShank支持Enter(Trigger)/Back/Funcion/Tp

inputKeyState: 指定按键状态，注意发送按键最好需要Down/Up成对出现

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

无

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.167

public static void SetBTRayCastDistance(float distance, BTType type = BTType.Right);

**说明：**

设置手柄组件检测Collider的范围半径，默认50米

**参数：**

Distance：type手柄组件检测Collider的范围半径，单位米

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

无

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.168

public static void SetBTEndPointerDistance(float distance, BTType type = BTType.Right);

**说明：**

设置手柄组件未检测到Collider光束的半径，默认3米

**参数：**

Distance: type手柄组件未检测到Collider光束的半径,单位米

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

无

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.169

public static Focus GetBTFocus(BTType type = BTType.Right);

**说明：**

获取手柄组件的Focus光标对象

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

手柄组件的Focus光标对象，注意有可能为null

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.170

public static LineRenderer GetBTLine(BTType type = BTType.Right);

**说明：**

获取手柄组件的光束对象

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

手柄组件的光束对象，注意有可能为null

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.171

public static void EnableBT(BTType type = BTType.Right);

**说明：**

开启手柄组件

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

无

**备注：**

可动态设置

**所属：**

API\_InputSystem\_Bluetooth.cs

## API-No.172

public static void DisableableBT(BTType type = BTType.Right);

**说明：**

关闭type手柄组件

**参数：**

type:BTType.Right表示第一个连接的手柄组件(默认),BTType.Left表示第二个连接的手柄组件

**返回值：**

无

**备注：**

无

**所属：**

API\_InputSystem\_Bluetooth.cs

# InputSystem模块API\_InputSystem\_Gesture26Dof

## API-No.200

public static InputDevice26DofGesture Gesture26Dof;

**说明：**

获取手势输入设备管理类

**参数：**

无

**返回值：**

手势输入设备

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.201

public static InputDevice26DofGesturePart GTLeft;

**说明：**

获取手势左手组件

**参数：**

无

**返回值：**

左手组件

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.202

public static InputDevice26DofGesturePart GTRight;

**说明：**

获取手势右手组件

**参数：**

无

**返回值：**

右手组件

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.203

public static handInfo GThandInfo(GestureType type = GestureType.Right);

**说明：**

获取手势组件的handInfo结构，handInfo包含手势的各类数据

**参数：**

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

手势组件的handInfo结构

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.204

public static SCPointEventData GTPointerEventData(GestureType type = GestureType.Right);

**说明：**

获取手势组件碰撞信息集合

**参数：**

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

手势组件碰撞信息集合

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.205

public static GameObject GTHitTarget(GestureType type = GestureType.Right);

**说明：**

获取手势组件碰撞Collider物体

**参数：**

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

返回碰撞的物体，有可能为null

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.206

public static RaycastHit GTHitInfo(GestureType type = GestureType.Right);

**说明：**

获取手势组件碰撞Collider的RaycastHit结构

**参数：**

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

手势组件碰撞Collider的RaycastHit结构

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.207

public static GameObject GTDragTarget(GestureType type = GestureType.Right);

**说明：**

获取手势组件拖拽的物体

**参数：**

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

手势组件拖拽的物体

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.208

public static bool IsGTKeyDown(InputKeyCode inputKeyCode, GestureType type = GestureType.Right);

**说明：**

判断手势组件是否触发某个Key Down

**参数：**

inputKeyCode:指定按键,暂时只支持Enter按键

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

True表示触发，false表示未触发

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.209

public static bool IsGTKeyUp(InputKeyCode inputKeyCode, GestureType type = GestureType.Right);

**说明：**

判断手势组件是否触发某个Key Up

**参数：**

inputKeyCode:指定按键,暂时只支持Enter按键

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

True表示触发，false表示未触发

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.210

public static InputKeyState GTKeyState(InputKeyCode inputKeyCode, GestureType type = GestureType.Right);

**说明：**

获取手势组件某个Key的状态

**参数：**

inputKeyCode:指定按键,暂时只支持Enter按键

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

指定按键的状态

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.211

public static InputKeyState GTKeyCurrentState(InputKeyCode inputKeyCode, GestureType type = GestureType.Right);

**说明：**

获取手势组件某个Key的实时状态

**参数：**

inputKeyCode:指定按键,暂时只支持Enter按键

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

指定按键的实时状态

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.212

public static void GTKeyAddKey(InputKeyCode inputKeyCode, InputKeyState inputKeyState, GestureType type = GestureType.Right);

**说明：**

模拟手势组件发送一个按键

**参数：**

inputKeyCode:指定按键,暂时只支持Enter按键

inputKeyState：指定按键状态，注意确保Down/Up成对出现

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

无

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.213

public static void SetGTRayCastDistance(float distance, GestureType type = GestureType.Right);

**说明：**

设置手势组件检测Collider的半径范围，默认50米

**参数：**

Distance：指定检测半径，单位米

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

无

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.214

public static void SetGTEndPointerDistance(float distance, GestureType type = GestureType.Right);

**说明：**

设置手势组件未检测到Collider时，Focus的位置半径，默认3米

**参数：**

Distance：指定Focus未检测到Collider时半径，单位米

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

无

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.215

public static Focus GetGTFocus(GestureType type = GestureType.Right);

**说明：**

获取手势组件的Focus

**参数：**

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

手势组件的Focus

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.216

public static LineRenderer GetGTLine(GestureType type = GestureType.Right);

**说明：**

获取手势组件的光束

**参数：**

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

手势组件的光束

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.217

public static void EnableGT(GestureType type = GestureType.Right);

**说明：**

开启手势组件

**参数：**

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

无

**备注：**

可动态设置

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.218

public static void DisableableGT(GestureType type = GestureType.Right);

**说明：**

关闭手势组件

**参数：**

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

无

**备注：**

可动态设置

**所属：**

API\_InputSystem\_Gesture26Dof.cs

## API-No.219

public static Model26DofGesture.fingerUI[] GetFingerUI(GestureType type = GestureType.Right);

**说明：**

获取手势组件fingerUI的结构

**参数：**

type：GestureType.Right表示右手组件（默认值），GestureType.Left表示左手组件

**返回值：**

手势组件fingerUI的结构

**备注：**

无

**所属：**

API\_InputSystem\_Gesture26Dof.cs