文章编号：1003－1251(2017)02－0017－05

# Unity3D 中的 Kinect 主角位置检测与体感交互

陈鹏艳，王洪源，陈慕羿

（沈阳理工大学 信息科学与工程学院，沈阳 110159）

**摘 要：**将 Kinect 与 Unity3D 联合，使玩家在三维虚拟场景里通过肢体控制游戏。根据 Kinect 的骨骼跟踪原理，对用户进行实时骨骼跟踪，利用 Kinect SDK 获取 Kinect 记录的关节点三维数据，实现硬件设备与 Windows 平台的连通。通过 CMU 的 Kinect Wrapper 实现 Kinect SDK 与 Unity3D 之间的数据交互。测试时，在 Unity3D 中建立三维虚拟场景，通过 Kinect Wrapper 中的 KinectModelControllerV2 脚本对场景内的人物模型进行控制，并在 Unity3D 平台上实时确定用户的三维位置信息，完成 Kinect 与 Unity3D 的体感交互，可以扩展应用于高级人机交互应用，如虚拟现实系统、作战模拟训练等方面。

**关 键 词：**Kinect；Unity3D；位置检测；体感交互

**中图分类号：**TP391 **文献标志码：**A

## Protagonist Position Detection and Somatosensory Interaction Based on Kinect in Unity3D

CHEN Pengyan，WANG Hongyuan，CHEN Muyi

（Shenyang Ligong University，Shenyang 110159，China）

**Abstract**：Kinect combined with Unity3D will take players into three-dimensional virtual scene and control game through their body. According to the principle of the skeleton tracking，the user′s skeleton is tracked in real-time by using Kinect SDK，which obtains joint three-dimensional position records by Kinect to implement communication of hardware devices and Windows platform. Kinect Wrapper of CMU will implement data interaction between the Kinect SDK and Unity3D. Three-dimensional virtual scene is built in Unity3D when testing by using the ControllerV2 script in Kinect Wrapper，which controls the character models and determines user′s real-time three-dimensional position on Unity3D playform，and somatosensory interaction between Kinect and Unity could be extended for advanced human-computer interaction applications like virtual reality system and combat simulation training.

**Key words**：Kinect；Unity3D；position detection；somatosensory interaction

**收稿日期：**2016－05－30

**基金项目：**辽宁省教育厅一般项目(L2013083)

**作者简介：**陈鹏艳(1992—)，女，硕士研究生；通讯作者：王洪源(1963—)，男，教授，研究方向：模拟仿真训练系统、区域数据链路体系结

Unity3D(也称 Unity)是一款跨平台次世代游戏引擎,以其强大的跨平台特性,独特的技术优势与绚丽的 3D 渲染效果而闻名出众[1],其人机交互主要依靠鼠标键盘进行,不具备检测玩家位置、肢体动作并用于交互的功能。

Kinect 是微软公司于 2009 年 6 月发布的 Xbox 360 的体感周边外设,是功能强大的 3D 外接体感摄影机,被誉为第三代人机交互的划时代之作[2]。Kinect 利用微软剑桥研究院研发的基于深度图像的人体骨骼追踪算法,不需要使用任何道具即可完成玩家整个动作的识别与捕捉,并能记录人体关节点的三维位置信息。

研究将 Kinect 与 Unity 联合,可以使用户身临其境地在三维虚拟场景里通过肢体或语言控制游戏,给体验者带来更真实的沉浸感。在 Unity 中建立 3D 虚拟场景,通过 Kinect SDK 获取 Kinect 硬件采集到的原始数据,Kinect SDK NUI API 对画面进行识别并对人体骨骼进行分类,获得 20 个人体骨骼关节点的三维坐标信息[3],利用 CMU Kinect Wrapper for Unity3D 实现 Unity 与 Kinect 间的数据交互,人物模型控制及各关节点三维位置信息的实时跟踪。用户在 Unity 中三维位置信息的确定有利于满足高级游戏设计需求,并可以应用于虚拟现实系统、作战模拟训练系统中。

## 1 Kinect 骨骼跟踪

如图 1 所示,Kinect 共有 3 个摄像头,中间为 RGB 摄像头,用来获取 640 × 480 的彩色图像,每秒钟最多获取 30 帧[4];左侧为红外线发射器,右侧为红外线 CMOS 摄像机所构成的 3D 结构光深度感应器,两侧不对称地分布着麦克风阵列,采用四元线性麦克风阵列技术,底座配有传动马达。Kinect 传感器设备提供三类原始数据信息,包括深度数据流、彩色视频流、原始音频数据,整体可实现骨骼跟踪、影像识别、语音识别等功能[5]。

**图 1 Kinect 实物图**

骨骼跟踪是 Kinect“体感操作”的基础,Kinect 的两个深度传感器会产生并接收随机分布的红外光线,通过 Kinect Primesense 芯片获取深度数据流,以计算出视场范围内每个像素的深度值,得到深度数据并从中提取出物体的主体和形状,与背景图像进行分割,根据这些信息对人体各部位进行分类匹配,从而获得人体关节点的三维坐标。Kinect 对人体部位的分类可以通过对像素逐一判断来确定人体关节点,系统会根据“骨骼跟踪”的 20 个关节点来生成一幅骨架系统,准确评估人体实际位置。图 2 为 Kinect 记录的深度图像和彩色图像,主要关节点连线组成的“火柴人”为 Kinect 识别的人体骨架系统。

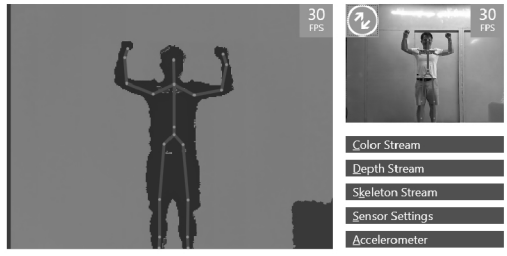


**图 2 人体深度、彩色图像及骨架系统**

骨骼位置信息的获取依赖于 Kinect SDK 的核心 NUI API,Kinect SDK 允许开发者借助微软 Visual Studio 2010,利用 C + + 或 C#等语言开发相关应用,可以使 Kinect 在 Windows 平台释放无限潜能。Kinect SDK 的设备驱动程序首先从硬件读取原始数据,包括图像数据、深度数据和音频数据,然后在 NUI 类库中进行计算,得到骨骼点位置,声源位置信息等,而 Kinect 应用则通过与 NUI 类库中的接口进行交互,来获取所需数据。目前,Kinect for Windows SDK 中的骨骼 API 可以提供至多两位用户的位置信息,包括详细的姿势和骨骼点的三维坐标信息[6]。在 SDK 中每个骨骼点都采用 Joint 类型表示,每一帧的 20 个骨骼点组成基于 Joint 类型的集合,包括三个属性:

1) JointType 骨骼点的类型,是一种枚举类型,列举了 20 个骨骼点的特定名称,如“HEAD”表示头部骨骼点。

2) Position SkeletonPoint 类型表示骨骼点的位置信息,是一个结构体,包含 $x$、$y$、$z$ 三个数据成员,用以存储骨骼点的三维坐标。

3) TrackingState JointTrackingState 类型也是一种枚举类型,表示该骨骼点的追踪状态。其中,Tracked 表示正确捕捉到该骨骼点,NotTracked 表示没有捕捉到骨骼点,Inferred 表示状态不确定[4]。

## 2　在 Unity 中进行骨骼绑定

Unity 是 Unity Technologies 提供的跨平台游戏开发工具和专业的游戏引擎,高效的开发模块,强大的渲染效果和可扩展能力,备受广大开发者青睐,便于实现实时游戏动画、三维虚拟场景、游戏逻辑设计等多元化游戏开发。Unity 支持通用语言脚本,可在一个平台上开发,多平台发布,使用 DirectX 和 OpenGL 图形引擎,提供高度优化的图形渲染管道,并且内置物理引擎,模拟刚体、关节等物理效果。此外,Unity 可兼容多种外部资源,可与 3dsMax、Maya 等程序协同工作。通过以上的功能特点,Kinect 传感器可与 Unity 结合实现人体三维位置检测。由于 Unity 不直接支持 Kinect SDK,因此本文利用 CMU Kinect Wrapper for Unity3D 实现 Unity 与 Kinect 间的数据交互。

Kinect Wrapper for Unity package 是一个对 Kinect 与 Unity 非常实用的中间件,其中包含一些脚本和场景示例。如:KinectExample 示例场景;KinectPrefab 预制件;Kinect Model Controller V2 脚本用于操纵模型骨骼,实现人物模型控制;KinectPointController 是对应于 KinectPointMan 的骨骼控制器;Display Depth 脚本用于获取深度图像;DisplayColor 脚本用于获取 RGB 图像;Kinect Recorder 用于记录用户动作,为 Kinect 模拟器(Emulator)产生回放文件;KinetEmulator 模拟 Kinect 设备,与 KinectRecorder 产生的回放文件一起工作;KinectSensor 用于获取 Kinect 中的数据;DeviceOrEmulator 用于设置使用 Kinect 物理设备或者模拟设备;SkeletonWrapper 脚本用于抓取骨骼数据;DepthWrapper 用于获取深度图像数据;KinectInterop 用于从 Microsoft Kinect SDK 中获取数据[7]。为了便于实现 Kinect 与 Unity 交互时场景的切换,在场景中放入 Kinect Prefab,使人物模型在切换场景时能正常使用。Kinect Prefab 包含开发体感项目的必需脚本,但不包含控制模型所需的控制器。因此,要为人物模型添加角色控制器 KinectModelControllerV2 脚本,并将 Bip 骨骼与人物模型在 Untiy 的 Inspector 外部变量中进行绑定,通过测试,人物模型能非常相似地模仿出用户动作。

## 3　三维位置的获取

使用应用程序获取下一帧骨骼数据的方式同获取彩色图像和深度图像的数据一样,都是通过调用回调函数并传递一个缓存实现的,获取骨骼数据调用的是 OpenSkeletonFrame( ) 函数。新的帧数据一旦准备好,系统会将其复制到缓存中,如果新的应用程序获取帧的速度大于帧数据准备的速度,只有等待下一个帧数据的触发。NUI 骨骼 API 提供了两种应用模型,分别是轮询模型和事件模型,下面对两种类型做简要介绍:

(1)轮询模型是最简单的获取帧数据的方法,通过调用 SkeletonStream 类的 OpenNextFrame( ) 函数即可实现。函数声明为 Public SkeletonFrame OpenNextFrame( int millisecondsWait ) 可以传递参数指定等待下一帧骨骼数据的时间,当新的数据准备好或是超出等待时间时,OpenNextFrame( ) 函数才会返回。

(2)事件模型以事件驱动的方式获取骨骼数据,更加灵活、准确。应用程序传递一个事件处理函数给 SkeletonFrameReady 事件,该事件定义在 KinectSensor 类中。当事件触发时,调用事件的属性 FrameReadyEventArgs 获取数据帧[4]。

本文采用事件模型获取用户的关节点位置信息,主要代码如下:

```
void _kinect_SkeletonFrameReady ( object sender,
SkeletonFrameReady EventArgs e)
{
    using( SkeletonFrame skeletonFrame = e. Open-
SkeletonFrame( ) )
    {
    if( skeletonFrame = = null)
    return;
    Skeleton s = Get Closet Skeleton
                ( skeletonFrame) ;
    if( s = = null)
```

```
        return;
        if( s. ClippedEdges = = FrameEdges. None)
        return;
        SkeletonPoint head =
        s. Joints[ JointType. Head]. Position;
            }
}
```

Kinect SDK NUI API 中包含彩色图像二维坐标、深度图像空间坐标、骨骼跟踪空间坐标,这三个坐标系的坐标和度量并不一致,通过 NUI 坐标转换,可以实现深度图像空间和骨骼空间坐标的转换。Kinect SDK 中提供了相关的 API 做相关转换,并且定义了 ColorImagePoint(彩色图像二维坐标点)、SkeletonPoint(骨骼跟踪三维坐标点)、DepthImagePoint(深度图像三维坐标点)三种点类型。被"骨骼跟踪"的用户位置由 $X$、$Y$、$Z$ 坐标系表示,该坐标系是三维的,以米为单位。$Z$ 轴表示红外摄像头光轴,与图像平面垂直。光轴与图像平面的交点,即为图像坐标系的原点。图 3 为测量结果。

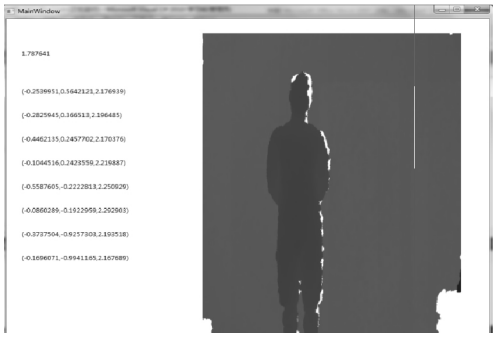

**图 3 用 Kinect 测量的人体身高及主要关节点三维数据**

表 1 为在用户相对于 Kinect 的不同位置,Kinect 测得的人体身高。

**表 1 在相对于 Kinect 的不同位置测量人体身高 m**

| 头部相对高度 | 左脚相对高度 | 相对补偿高度 | 被测人体高度 |
|---|---|---|---|
| 0.5429 | -0.9282 | 0.2882 | 1.7593 |
| 0.5535 | -0.9015 | 0.2998 | 1.7549 |
| 0.5452 | -0.9566 | 0.2844 | 1.7863 |
| 0.5642 | -0.9257 | 0.2977 | 1.7876 |
| 0.5592 | -0.9064 | 0.2988 | 1.7645 |
| 0.5422 | -0.9574 | 0.2887 | 1.7883 |
| 0.5453 | -0.9506 | 0.2883 | 1.7842 |



被测的人体实际身高为 1.78m,根据测量对比,利用 Kinect 测量的人体身高误差为 -0.025m 至 0.008m 之间,证明 Kinect 进行的人体关节点位置跟踪具有一定的准确性,并伴有少量误差。

在 Unity 中进行测试时,可以通过手动封装 DLL 以及使用 Kinect Wrapper for Unity Package 的方法获取 Kinect SDK 中的彩色图像数据流,深度数据流。本文选用后一种方法,在 Kinect ModelController 脚本中添加相关程序,实时获取用户三维位置信息。

Kinect 所测量的深度图像坐标系和骨骼跟踪坐标系都是 Kinect 的摄像头坐标系,原点为红外摄像头中心,SkeletonWrapper 脚本中使用位移矩阵,调用 Matrix4x4 成员函数 SetTRS,void SetTRS(Vector3 pos,Quaternion q,Vector3 s)实现从 Kinect 摄像头坐标系向屏幕世界坐标系的变换。

```
Matrix4x4 trans = new Martrix4x4( );
trans. SetTRS ( newVector3 ( - kinect. getkKinect-
Center( ). x,kinect. get
                SensorHeight ( ) - kinect. getKinectCenter
( ). y, - kinect. getKinectCenter( ). z),Quaternion. i-
dentity,Vector3. one);
Matrix4x4 rot = new Martrix4x4( );
Quaternion quat = new Quaternion( );
double theta = Mathf. Atan( ( kinect. getLookAt( ).
y + kinect. getKinectCenter( ). y - kinect. getSenso-
rHeight( ) );
float kinectAngle = ( float) ( theta * (180/Mathf.
PI) );
quat. eulerAngles = new Vector3( - kinectAngle,0,
0);
rot. SetTRS( Vector3. zero,quat,Vector3. one);
Matrix4x4 flip = Matrix4x4. identity;
flip[2,2] = -1;
//Kinect 的旋转变换矩阵补偿,转换成一个新的中心
kinectToWorld = flip * trans * rot;
```

## 4 测试结果

如图 4 所示,在测试过程中,用户通过 Unity 可以控制场景中的人物模型,使人物模型跟随人

体活动。选取右手 Hand_Right 为跟踪目标,当用户的右手从向下自然垂下到缓慢向上举起的过程中,Kinect 检测的右手运动轨迹的三维坐标如表 2 所示。

**图 4　用户控制 Unity 中人物模型运动**

**表 2　Hand_Right 运动轨迹**

| *X* 轴 | *Y* 轴 | *Z* 轴 |
|---|---|---|
| -40.4 | 6.5 | -68.7 |
| -40.3 | 6.5 | -68.7 |
| -40.1 | 6.8 | -68.6 |
| -39.9 | 7.1 | -68.5 |
| -39.9 | 7.3 | -68.4 |
| -39.9 | 7.6 | -68.3 |
| -40.0 | 7.9 | -68.3 |
| -40.1 | 8.1 | -68.3 |
| -40.3 | 8.4 | -68.3 |
| -40.4 | 8.4 | -68.4 |

右手在体侧向下自然下垂到向上伸展运动的过程中,*X* 轴的数值对称,起始和向上伸直的最终坐标稳定在 -40.4 附近;当右臂伸直平举时,*X* 轴坐标最大达到 -39.9,*Y* 轴的数值在 6.5 ~ 8.4 范围内不断增加;在底端和顶端与平举时的差值均为 1m 左右,与用户的实际右臂长度基本一致。

通过对测量数据进行分析可得,Kinect 的骨骼坐标检测技术,在 Unity 平台中能实时确定用户的三维位置信息,对各关节点的位置均有准确捕捉,并且人物模型受用户控制,能非常近似地模拟出动作变化,满足游戏设计需求。

## 5　结束语

利用 Kinect 传感器实现人体骨骼跟踪以及关节点三维数据的获取,在 Unity 中与人物模型进行骨骼绑定,通过人体动作控制 Unity 中人物模型的活动,并在 Unity 中实时获取用户的三维位置信息,完成体感的交互。基于 Kinect 和 Unity 的人体骨骼控制和三维位置信息获取,能实时捕捉人体动作 并完成人物定位,带给用户良好的沉浸感,可广泛应用于模拟训练系统,医疗康复系统,大型体感游戏等。

## 参考文献:

[1] 张金钊,孙颖,王先清,等. Unity3D 游戏开发与设计案例教程[M]. 北京:清华大学出版社,2015.

[2] 王森. Kinect 体感程序设计入门:使用 C#和 C + + [M]. 北京:科学出版社,2014.

[3] Jamie Shotton,Andrew Fitzgibon,Andrew Blake,et al. Real-Time Human Pose Recognition in Parts from a Single Depth Image[J]. Communications of the ACM,2013,56(1):116 - 124.

[4] 吴国斌,李斌,阎骥洲. KINECT 人机交互开发实践[M]. 北京:人民邮电出版社,2013.

[5] 张志常,范婷. 基于 Kinect 的 MoCA 康复认知评估系统设计[J]. 渤海大学学报:自然科学版,2015,3(1):49 - 52.

[6] 余涛. Kinect 应用开发实战,用最自然的方式与机器对话[M]. 北京:机械工业出版社,2012.

[7] 姚翠莉,袁璠,彭飞翔,等. Kinect 在 Unity 平台上的开发实例[J]. 计算机光盘软件与应用,2014(12):68 - 72.

(责任编辑:马金发)