электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.354

Kinect как средство естественного взаимодействия человек-компьютер

Романов А.В., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
кафедра «Системы обработки информации и управления»

Научный руководитель: Филиппович Ю. Н., преподаватель
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
chernen@bmstu.ru

Введение

В настоящее время все большую актуальность приобретает задача построение естественного человеко-машинного интерфейса – то есть такого, который прозрачен для пользователя и базируется на естественных операциях [1]. Большинство существующих интерфейсов используют какие-либо устройства и нуждаются в том, что бы пользователь был обучен работать с ними. В естественном пользовательском интерфейсе, даже если и требуется обучение, оно происходит гораздо быстрее и требует меньше усилий.

Одно из устройств, призванных обеспечить естественный пользовательский интерфейс –контроллер Kinect, выпущенный Microsoft 4 ноября 2010 года для игровой приставки Xbox 360, а позднее, 1 февраля 2012 года, была выпущена версия для персонального компьютера и специальное SDK для разработки приложений, лицензионное соглашение которого позволяло коммерческое использование [2].

Технические характеристики Kinect

Kinect представляет из себя набор датчиков и сенсоров и, как видно из рис. 1, содержит следующие элементы:

- ✓ Color Sensor – представляет собой камеру, способную выдавать цветное изображение с максимальным разрешением 1280x960 при 12 кадрах в секунду, или 640x480 при 30 кадрах в секунду, зона обзора составляет 57 градусов по горизонтали и 43 градуса по вертикали
- ✓ IR Emitter и IR Depth Sensor– излучатель и приемник инфракрасных лучей соответственно. ИК-приемник на основании принятых отраженных лучей, выпущенных ИК-излучателем, способен построить карту глубины разрешением 640x480 при 30 кадрах в секунду, углы обзора аналогичны камере.

✓ Microphone Array – массив микрофонов. Из-за того, что условия работы Kinect являются неблагоприятными – подразумевается, что устройство будет установлено под/над телевизором, который при работе издает некоторый шум, причем его громкость для Kinect значительно выше громкости человеческого голоса, то для уверенного распознавания голоса, а также для определения направления на его источник, в Kinect используется целых 4 микрофона – 1 с одной стороны, и 3 с другой стороны.

✓ Tilt Motor – так как возможно установка Kinect и ниже телевизора, и выше него, то для коррекции угла наклона используется мотор, способный наклонять устройство на угол ±270 по вертикали

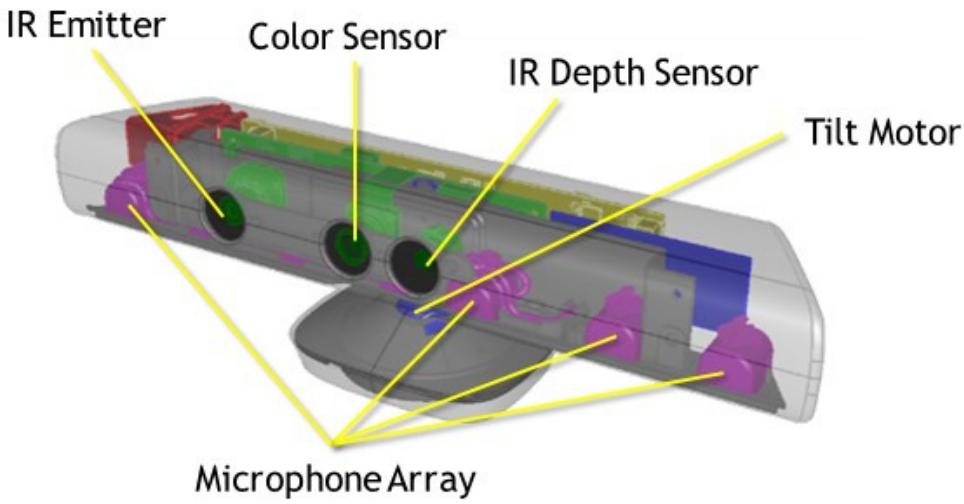

Рис. 1. Внутренне устройство Kinect [5]

Важным отличием Kinect для Windows от версии для Xbox 360 является дистанция работы устройства – если в версии для игровой приставки предполагалось значительное удаление пользователя от телевизора, и диапазон работы составляет 0,8-4 м., то при работе с персональным компьютером пользователь расположен существенно ближе, и Kinect для Windows может работать в двух режимах – в режиме по умолчанию и в ближнем режиме; в режиме по умолчанию рабочий диапазон аналогичен версии для Xbox 360, а в ближнем режиме минимальное расстояние составляет 0,4 м., а максимальное – 3 м [3].

Kinect способен распознавать до 6 человек в кадре одновременно с выделением их центра масс, из них для 2 человек может быть распознан скелет из 20 точек [3], см. рис. 2

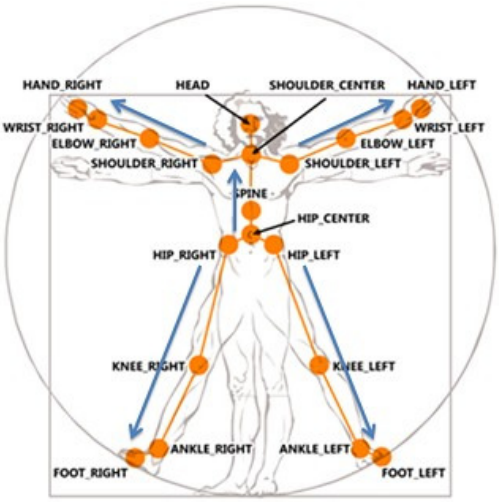

Рис. 2. Точки скелета, которые может распознать Kinect [4]

SDK для работы с Kinect

Существует два основных SDK для разработки с использованием Kinect – официальный SDK, который доступен для свободной загрузки на официальной странице (http://www.microsoft.com/en-us/kinectforwindows/develop/), а также SDK, выпущенный некоммерческой организацией Open Natural Interaction (http://www.openni.org/). Недостатком последнего является слабая документация, среди преимуществ можно назвать поддержку аналога Kinect от компании Asus – Asus Xtion (http://ru.asus.com/Multimedia/Motion_Sensor/Xtion_PRO/), а также открытый исходный код.

SDK OpenNI предоставляет базовые возможности, однако, что бы, например, получить скелет пользователя, разработчику необходимо использовать библиотеку NITE с закрытым исходным кодом, совместимую с OpenNI, выпущенную компанией PrimeSense (http://www.primesense.com/), главным участником Open Natural Interaction. SDK OpenNI рассчитан на использование с языком C++, однако, до версии 2 в комплекте поставлялись библиотеки-обертки для использования с C# и Java. Версия 2 сейчас находится в стадии бета-тестирования, и в комплекте поставки такие библиотеки отсутствуют.

Пример приложения с распознаванием жестов

Рассмотрим пример консольного приложения для Windows на C#, использующее Kinect или Asus Xtion при помощи OpenNI версии 1.5.4.0. Данное приложение будет служить для перелистывания слайдов в PowerPoint с помощью жеста «махание рукой».

После создания проекта необходимо установить в свойствах проекта на вкладке «Построение» параметр «Конечная платформа» в соответствии с используемой версией

библиотеки, x86 или x64, а также добавить в проект файл с конфигурацией со следующим содержанием:

```
<OpenNI>
          <Licenses>
          </Licenses>
          <Log writeToConsole="true" writeToFile="false">
                    <LogLevel value="3"/>
                    <Masks>
                              <Mask name="ALL" on="true"/>
                    </Masks>
                    <Dumps>
                    </Dumps>
          </Log>
          <ProductionNodes>
                    <GlobalMirror on="true"/>
                    <Node type="Depth" name="MyDepth">
                              <Query>
                              </Query>
                              <Configuration>
                                        <!-- Uncomment to set requested mode
                                        <MapOutputMode xRes="640" yRes="480" FPS="30"/>
                                        -->
                              </Configuration>
                    </Node>
                    <Node type="Gesture" />
          </ProductionNodes>
</OpenNI>
```

Как видно из приведенного выше, в данном файле содержаться настройки OpenNI, например, разрешение сенсора, а также перечислены все типы узлов, которые предполагается использовать – в данном случае Depth, т.е. глубина, и Gesture – т.е. жесты.

Для реализации функционала перелистывания слайдов в PowerPoint будет использоваться эмуляция нажатия клавиш «стрелка вниз» путем посылки активному окну сообщения WM_KEYDOWN и WM_KEYUP, для чего следует объявить WinAPI-функции и используемые константы:

```
#region Win32 API
const UInt32 WM_KEYDOWN = 0x0100;
const UInt32 WM_KEYUP = 0x0101;
const int VK_RIGHT = 0x27;
const int VK_DOWN = 0x28;
[DllImport("user32.dll")]
public static extern IntPtr GetForegroundWindow();
[DllImport("user32.dll")]
static extern bool PostMessage(IntPtr hWnd, UInt32 Msg, int wParam, int lParam);
#endregion
```

Вместо посылки сообщений возможен вариант взаимодействия с использованием пространства имен Microsoft.Office.Interop.PowerPoint, но, в данном случае, из соображений простоты был выбран данный метод.

Теперь объявим переменные, необходимые для работы с OpenNI:

```
private static string SENSOR_CONFIG = @"../../SensorConfig.xml";
private static Thread workThread;
private static Context context;
private static ScriptNode scriptNode;
private static DepthGenerator depthGenerator;
private static GestureGenerator gestureGenerator;
private static bool isActive = true;
```

Здесь SENSOR_CONFIG – путь к файлу конфигурации, workThread – поток, в котором будут получаться данные от сенсора, isActive – признак активности этого потока, остальные переменные – служебные, назначение их будет понятно из кода далее.

Для начала работы следует создать контекст с помощью функции Context.CreateFromXmlFile, затем получить узлы для глубины и жестов и вызвать у них метод StartGenerating:

```
context = Context.CreateFromXmlFile(SENSOR_CONFIG, out scriptNode);
depthGenerator = context.FindExistingNode(NodeType.Depth) as DepthGenerator;
gestureGenerator = context.FindExistingNode(NodeType.Gesture) as GestureGenerator;
gestureGenerator.AddGesture("Wave");
gestureGenerator.GestureRecognized += gestureGenerator_GestureRecognized;
depthGenerator.StartGenerating();
gestureGenerator.StartGenerating();
```

Как видно из кода выше, у объекта типа GestureGenerator присутствует метод AddGesture, который принимает строку с название жеста (в данном случае Wave, что соответствует жесту «махание рукой»), и событие GestureRecognized, которые вызывается при распознавании жеста.

Для получения и обновления данных с сенсора необходимо постоянно вызывать метод WaitOneUpdateAll у объекта типа Context – для этого и служит поток workThread:

```
workThread = new Thread(worker);
workThread.Start();
. . .
static private void worker()
{
        while (isActive)
        {
                context.WaitOneUpdateAll(depthGenerator);
        }
}
```

После этого все, что необходимо сделать – в обработчике события GestureRecognized найти активное окно, запомнить его, что бы не повторять поиск в дальнейшем, и послать ему последовательно сообщения WM_KEYDOWN и WM_KEYUP с кодом нажатой клавиши «стрелка вниз»:

```
static void gestureGenerator_GestureRecognized(object sender, GestureRecognizedEventArgs e)
{
        if (activeWindow == IntPtr.Zero)
                activeWindow = GetForegroundWindow();

        PostMessage(activeWindow, WM_KEYDOWN, VK_DOWN, 0);
        Thread.Sleep(200);
        PostMessage(activeWindow, WM_KEYUP, VK_DOWN, 0);
}
```

Теперь, если скомпилировать и запустить эту программу на компьютере, к которому подключен Kinect или Asus Xtion, а потом открыть в режиме показа презентацию PowerPoint, то, если осуществить жест «махание рукой», текущий слайд переключится на следующий!

http://sntbul.bmstu.ru/doc/593126.html

Некоторые рекомендации по оптимизации взаимодействия с Kinect

Для оптимальной работы сенсоры рекомендуется соблюдать следующие условия [6]:

- ✓ Размещение Kinect на высоте 0,6-1,8 м. от пола на расстоянии не менее 30 см. от динамиков
- ✓ Минимальное боковое или фоновое освещение
- ✓ Не рекомендуется попадание прямого солнечного освещения на пользователя или Kinect
- ✓ Яркое и равномерное освещение помещения
- ✓ Отсутствие предметов между сенсором и пользователем
- ✓ Не рекомендуется одевать мешковатую одежду
- ✓ Темная одежда может поглощать инфракрасное излучение, поэтому рекомендуется одеваться в светлые тона
- ✓ При выполнении жеста следует стоять лицом к сенсору и не следует выходить из его поля зрения – см. рис 3

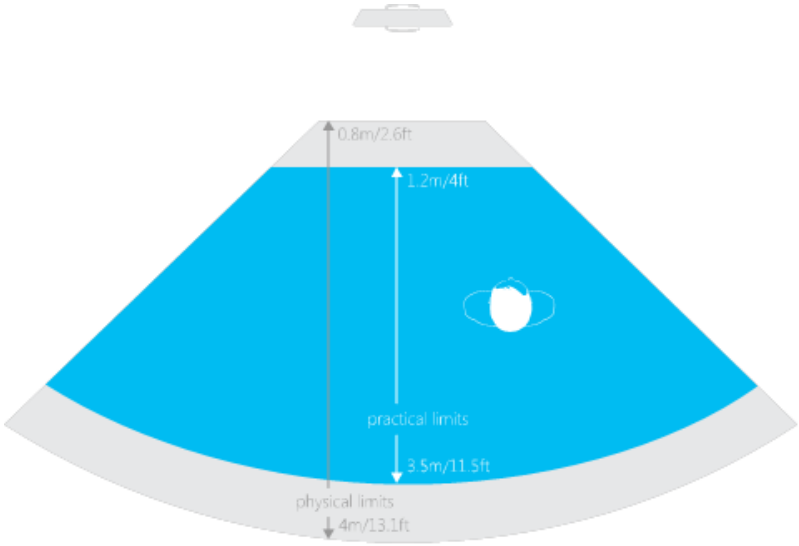

Рис. 3. Поле зрения Kinect по горизонтали [3]

Список литературы

1. Natural user interface // Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org/wiki/Natural_user_interface (дата обращения: 11.03.2013)
2. Kinect // Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org/wiki/Kinect (дата обращения: 25.02.2013).
3. Skeletal Tracking // MSDN. URL: http://msdn.microsoft.com/en-us/library/hh973074.aspx (дата обращения: 11.03.2013).

4. Tracking Users with Kinect Skeletal Tracking // MSDN. URL: http://msdn.microsoft.com/en-us/library/jj131025.aspx (дата обращения: 11.03.2013)
5. Kinect for Windows SDK. Часть 1. Сенсор // Хабрахабр. URL: http://habrahabr.ru/post/150955/ (дата обращения: 25.02.2013).
6. Устранение неполадок с отслеживанием телодвижений // Поддержка Xbox Kinect - Xbox.com http://support.xbox.com/ru-RU/xbox-360/kinect/body-tracking-troubleshoot (дата обращения: 11.03.2013).