

УДК 004.354

Kinect как средство естественного взаимодействия человек-компьютер

***Романов А.В.**, студент*

*Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедры «Системы обработки информации и управления»*

*Научный руководитель: Филиппович Ю. Н., преподаватель
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана*

bauman@bmstu.ru

Kinect представляет собой периферийное устройство, общий вид которого приведен на рис. 1, выпущенное компанией Microsoft 4 ноября 2010 для игровой приставки Xbox 360, которое сразу вошло в книгу рекордов Гиннесса как самое быстро продаваемое бытовое устройство – за первые 60 дней было продано более 8 миллионов экземпляров. Первоначально Kinect был выпущен только для Xbox 360, но быстро появились переходники и драйверы для подключения его к компьютеру. Не смотря на то, что данный способ был неофициальным, появилось много разработок для использования Kinect на персональном компьютере, но из-за того, что лицензионное соглашение не позволяло его коммерческое использование, первоначально приложений, реализующих все возможности Kinect, было немного, но затем, с выпуском компанией Microsoft 1 февраля 2012 специальной версии Kinect – Kinect for Windows, а также специального SDK для разработки приложений и возможностью коммерческого использования, не стало препятствий для разработки и продажи программного обеспечения для персонального компьютера, использующего возможности, предоставляемые Kinect.



Рис 1. Общий вид контроллера Kinect

Kinect представляет из себя набор датчиков и сенсоров, которые позволяют управлять компьютером или игровой приставкой без использования какого-либо контроллера. Как видно из рис. 2, устройство 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 возможна установка Kinect и ниже телевизора, и выше него, то для коррекции угла наклона используется мотор, способный наклонять устройство на угол $\pm 27^{\circ}$ по вертикали

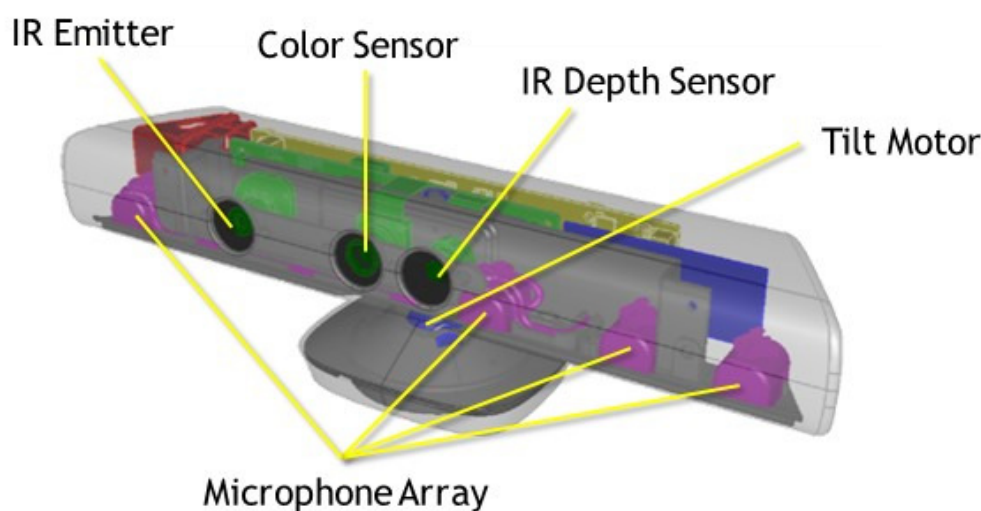


Рис 2. Внутренне устройство Kinect

Важным отличием Kinect for 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 for 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 – 4 м.

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, см. рис. 3

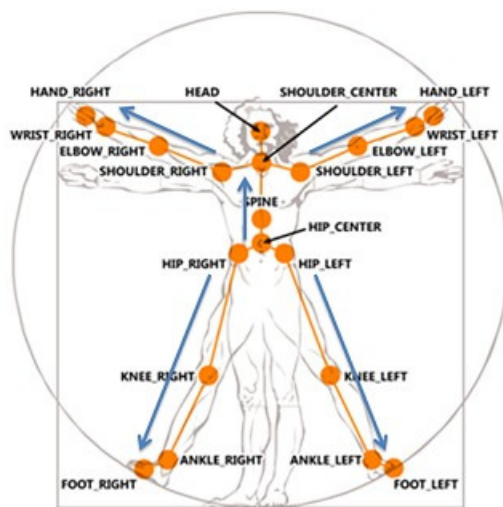


Рис 3. Точки скелета, которые может распознать Kinect

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.

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 содержатся 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 отправки 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
```

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 названием жеста (в данном случае `Wave`, что соответствует жесту «махание рукой»), и событие `GestureRecognized`, которое 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, чтобы 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!

Список литературы

1. Kinect // Wikipedia, the free encyclopedia. URL: <http://en.wikipedia.org/wiki/Kinect> (дата обращения: 25.02.2013г.).
2. Kinect for Windows SDK. Часть 1. Сенсор // Хабрахабр. URL: <http://habrahabr.ru/post/150955/> (дата обращения: 25.02.2013г.).
3. Контроллер Kinect попал в Книгу рекордов Гиннеса // 3D News – Daily Digital Digest. URL: <http://www.3dnews.ru/news/kontroller-kinect-popal-v-knigu-rekordov-ginnesa> (дата обращения: 25.02.2013г.).