Tim Gerber

Luftpiano

Mit der Kinect V2 MIDI-Geräte steuern

Bereits wenige Programmzeilen genügen, um mit der jüngsten Version des 3D-Kamera-Sensors Kinect für Windows mit dem Finger in der Luft Klavier zu spielen. An dem Beispiel zeigt sich, wie leicht man an die Sensor-Daten kommt.

Vor Kurzem hat Microsoft das Software Developer Kit (SDK) 2.0 für die aktuelle Kinect für Windows freigegeben. Damit sollen vor allem Entwickler gelockt werden, Anwendungen für den Sensor zu programmieren. Zum Paket gehört die notwendige Lizenz, solche Software mit Teilen der Microsoft-eigenen Bibliotheken dann auch kommerziell zu vertreiben.

Interessant ist die neue Kinect für Windows aber nicht nur für kommerzielle Anwendungen. Sie kostet 200 Euro, und wer schon eine neue Kinect für die Xbox One besitzt, muss lediglich für 50 Euro einen Adapter für den PC-Anschluss dazu erstehen.

Das eigentlich Bestechende an der Kinect für Windows ist das überarbeitete SDK und die zugehörige Bibliothek. Damit ist das Nutzen der Sensordaten in selbst geschriebenen Programmen sehr einfach. Wir zeigen im Folgenden beispielhaft, wie man mit der Bewegung eines Fingers Soundeffekte erzeugen kann.

Die Kinect für Windows V2 ist schneller und präziser als das Vorgängermodell. Dadurch ist es ihr unter anderem möglich, neben „offen“ und „geschlossen“ noch einen dritten Status der Hände einer vor dem Sensor agierenden Person zu erkennen: einen ausgestreckten Finger. Grundsätzlich funktioniert das Erkennen von Akteuren mit der Vorgängerversion aber ähnlich. Allerdings sind im bisherigen SDK 1.8 die Namenskonventionen und einige weitere Details anders, sodass das hier vorgestellte Beispiel nicht eins zu eins übertragbar ist. Was im SDK 2.0 beispielsweise mit „Body“ bezeichnet wird, also alle Objekte, die Daten der Akteure und ihrer Positionen enthalten, heißt im SDK 1.8 und früher „Skeleton“. Die neuen Objekte sind ausgefeilter und auf die jüngste .NET-Version 4.5 ausgerichtet. Die Grundprinzipien aber sind in beiden Varianten dieselben. Wer für ein paar Experimente mit dem Kinect-Sensor keinen dreistelligen Betrag ausgeben will, bekommt die ältere Version für etwa 50 bis 70 Euro in Online-Auktionen. Man muss beim Kauf nur darauf achten, dass der für den Anschluss an den PC notwendige USB-Adapter dabei ist. Einen Artikel über die Portierung von Anwendungen für die Kinect 1 auf die Kinect 2 auf *heise developer* sowie Download-Adressen aller erwähnter Software finden Sie über den c't-Link am Ende des Artikels.

Wegen der höheren Auflösung und des schnelleren Datenstroms stellt die Kinect für Windows V2 auch erheblich höhere Anforderungen an die PC-Hard- und Software, nämlich mindestens Windows 8, USB 3.0 und eine DirectX-11-fähige Grafikkarte. Das SDK inklusive der Bibliothek gibt es bei Microsoft zum freien Download. Mit der Installation landet auch der nützliche SDK-Browser auf der Platte. Er listet alle verfügbaren Beispielprogramme auf und befördert deren Quelltexte auf einen Klick ebenfalls auf die Festplatte oder startet die Ausführung der bereits mitinstallierten Exe-Programme.

Die Kinect liefert dem PC einen Strom von Rohdaten. Die sehr komplexe Auswertung dieser Daten übernimmt die Microsoft-Bibliothek. Sie stellt verschiedene Objekte zur Verfügung, über deren Methoden und Eigenschaften Programmierer auf einfache Weise auf die Auswertung der Sensordaten für verschiedene Zwecke zugreifen können. Das sind zum Beispiel Pixeldaten der Kamera nebst Tiefeninformationen des Infrarot-Sensors, aber auch bereits komplexe Auswertungen etwa über erkannte Personen, die sich vor der Kinect bewegen. Diese Informationen aus den Rohdaten zu gewinnen ist eine beachtliche Leistung der Microsoft-Entwickler. Der Programmierer von Anwendungen für die Kinect braucht sich um diese Dinge nicht mehr zu kümmern, sondern greift einfach auf die Daten zu, die ihm die Objekte des Kinect-Frameworks mundgerecht liefern. Für jedes dieser Objekte listet der Kinect-Explorer Beispielprogramme in verschiedenen Versionen auf.

Für unser Experiment ist das Objekt `BodyFrameReader` von Interesse. Es liefert Daten über die erkannten Personen. Nur für die ersten beiden der insgesamt sechs möglichen Akteure sind die Daten aber so detailliert bis sprichwörtlich in die Fingerspitzen, wie wir es eingangs beschrieben haben. Mithilfe des Objekts kommt man ziemlich einfach an alle benötigten Daten heran, um die Position eines ausgestreckten Fingers der rechten Hand einer vor der Kinect agierenden Person auf den Zentimeter genau zu ermitteln.

Unser Beispiel haben wir mit der kostenlosen Express-Version von Visual Studio 13 in C# programmiert, aber auch in den anderen

Sprachen sollte es ohne Weiteres nachvollziehbar sein. Es ist als klassisches Desktop-Programm (Windows Form) gestaltet, man benötigt zum Nachvollziehen also auch die Desktop-Variante von Visual Studio Express und nicht die für Windows-Store-Programme vorgesehene Variante „Windows“.

Wenn Sie das Folgende nachvollziehen wollen, legen Sie zunächst ein neues Windows-Forms-Projekt in Visual Studio an. Mit einem Rechtsklick auf den Eintrag „Verweise“ im Projektmappen-Explorer öffnen Sie ein Kontext-Menü. Darin wählen Sie „Verweis hinzufügen …“ und wählen anschließend die Datei kinect.dll aus. Sie befindet sich im Ordner Programme\Microsoft SDKs\Kinect\MainV2\Assemblies. Anschließend fügen Sie unter den using-Direktiven am Anfang des Programms die Zeile `using Microsoft.Kinect;` ein. Wenn der Verweis zuvor korrekt angelegt wurde, müsste die Entwicklungsumgebung bereits mit ihrer Auto-Ausfüllfunktion den Namensraum (Namespace) der Kinect-Bibliothek erkennen. Andernfalls wird sie den Eintrag als Fehler ankringeln. Dann ist bei der Verknüpfung etwas schiefgegangen und Sie sollten die oben beschriebene Prozedur wiederholen.

Die Verwendung des BodyFrameReader-Objekts demonstriert die Beispiel-App „Body Basics“, die es wie die meisten Beispielprogramme des Kinect-SDK in C++, C#, Visual Basic und sogar in HTML gibt. Sie zeichnet ein symbolisches Skelett der vom Sensor erfassten Personen mit den markanten Knotenpunkten etwa an den Knien, den Schultern und auch den Händen. Mit farbigen Kreisen zeigt sie den Zustand der Hände als offen (grün), geschlossen (rot) oder geschlossen mit einem ausgestreckten Finger (blau) an. Die grafische Darstellung macht den Löwenanteil am Code des Beispielprogramms aus. Um das BodyFrame-Objekt ins eigene Programm einzubinden und darüber an die Sensordaten selbst zu kommen, benötigt man nur wenige Zeilen davon.

Zunächst braucht das Fensterobjekt Form1 ein paar Variablen für die verwendeten Kinect-Objekte:

```
private KinectSensor kinectSensor = null;
private BodyFrameReader bodyFrameReader = null;
private Body[] bodies = null;
```

Ein Doppelklick auf das Programmfenster in der Entwurfsansicht erzeugt eine Form1_Load-Routine, die bei Erzeugung des Programmfensters automatisch aufgerufen wird. Dort initialisieren Sie den Kinect-Sensor und registrieren die Ereignisbehandlung für das Event FrameArrived in der Ereignisbehandlungsschleife des Programms. Das Ereignis tritt, wie der Name sagt, immer dann auf, wenn ein komplettes Frame mit Sensordaten eingetroffen ist. Je nach Leistungsfähigkeit des PC schafft die Kinect 25 bis 30 Frames pro Sekunde. Die Ereignisbehandlungsroutine wird also 25 bis 30 Mal in der Sekunde aufgerufen, das sollte man später bedenken, wenn man dort Reaktionen auf bestimmte Sensorinformationen hineinprogrammiert.

Die im Kasten unten zu sehende Ereignisbehandlungsroutine erhält über das mitgelieferte Argument vom Typ BodyFrameArrivedEventArgs Zugriff auf den aktuellen Frame und fragt zunächst ab, ob die Daten gültig sind. Mit der Zeile `bodyFrame.GetAndRefreshBodyData(bodies);` werden die Daten des aktuellen Body-Frames in das Array bodies geladen. Man kann nun mit einer foreach-Schleife die Daten aller erkannten Personen auswerten. Nachdem über die Abfrage `if (body.IsTracked)` die erste Person identifiziert ist, bricht das Programm über ein break die foreach-Schleife ab, sodass weitere Personen nicht berücksichtigt werden.

Die Koordinaten der einzelnen Punkte speichert das Body-Objekt in einem Dictionary namens Joints. Der Zugriff auf die Elemente erfolgt über definierte Schlüssel-Werte, in unserem Fall mit `Joint joint = body.Joints[JointType.HandTipRight];`. Das Autovervollständigen der Visual-Studio-Umgebung hilft ungemein, um sich hier zum Beispiel alle Werte vom Joint@$>-Typ anzeigen zu lassen. Auf diese Weise kann man übrigens auch weitere im BodyFrame-Objekt schlummernde Informationen aufstöbern.

Wurde die Fingerspitze der rechten Hand erkannt (`if (joint.TrackingState == TrackingState.Tracked)`), holt man sich die Werte für dessen X- und Y-Position. Sie werden als vorzeichenbehaftete Gleitkommazahl in Metern geliefert, die Mitte des Erkennungsbereichs des Sensors ist dafür der Ausgangspunkt, Positionen links oder unterhalb der Mitte sind deshalb negativ. Für erste Experimente lässt man sich die Werte am besten nach dem Schema `label1.Text = joint.Position.X.ToString();` in einer Textkomponente anzeigen. Der güns-

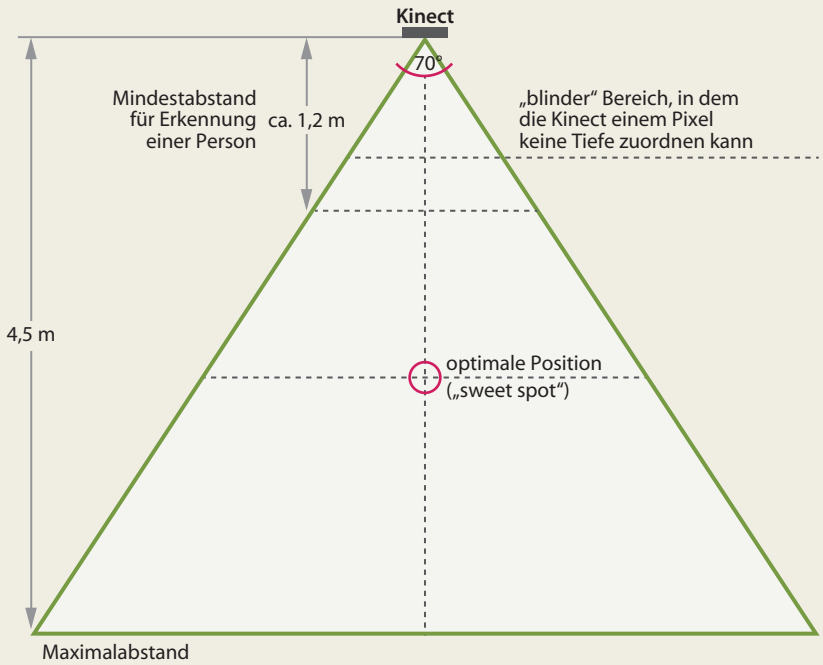
Die Kinect V 2 erkennt dank verbesserter Hardware auch die Position eines einzelnen Fingers und kann dadurch zum Klavierspielen in der Luft benutzt werden.

```
private void Reader_FrameArrived(object sender, BodyFrameArrivedEventArgs e)
{
    bool dataReceived = false;
    int count = 0;
    using (BodyFrame bodyFrame = e.FrameReference.AcquireFrame())
    {
        if (bodyFrame != null)
        {
            if (bodies == null)
            {
                bodies = new Body[bodyFrame.BodyCount];
            }
            bodyFrame.GetAndRefreshBodyData(bodies);
            dataReceived = true;
        }
    }
    if (dataReceived)
    {
        foreach(Body body in bodies)
        {
            if (body.IsTracked)
            {
                Joint joint = body.Joints[JointType.HandTipRight];
                if (joint.TrackingState == TrackingState.Tracked)
                {
                    int x = Convert.ToInt32((joint.Position.X * 100 + 200) / 3.3);
                    int y = Convert.ToInt32(joint.Position.Y * 100 + 100);
                    label1.Text = x.ToString();
                    label2.Text = y.ToString();
                    {
                        midiOut.Send(MidiMessage.StartNote(x, 127, 2).RawData);
                        playing = true;
                    }
                    if ((y > 90) && playing)
                    {
                        midiOut.Reset();
                        playing = false;
                    }
                }
                else
                {
                    label1.Text = "Nicht erkannt!";
                    midiOut.Reset();
                    playing = false;
                }
                break;
            }
        }
    }
}
```

Die Ereignisbehandlungsroutine Reader_FrameArrive **bildet den Dreh- und Angelpunkt des Beispielprogramms.**

Erfassungsbereich der Kinect V2

Das Luftklavier spielt sich am besten in ungefähr zwei Metern Abstand vom Sensor.



tigste Erfassungsbereich der Kinect ist zwischen etwas über einem und viereinhalb Metern (siehe oben stehendes Schaubild), für das hiesige Beispiel reicht ein Abstand von zwei Metern völlig aus.

Wenn das Programm so weit funktioniert und die Koordinaten des ausgestreckten Fingers auf dem Bildschirm angezeigt werden, kann man darangehen, in Abhängigkeit von Bewegung und Position Reaktionen zu programmieren: Wenn der Finger nach unten bewegt wird, soll das Programm einen Ton ausgeben und wenn er wieder gehoben wird, den Ton stoppen. Die Tonhöhe soll sich nach der X-Position des Fingers richten. Am Ende kann der Anwender also mit einem Finger in der Luft auf dem PC Klavier spielen wie mit dem in [1] vorgestellten Theremin.

Zur Tonerzeugung nutzt man den Windows-internen Software-Synthesizer, der über das Musical Instruments Device Interface (MIDI) angesprochen wird. Die Programmierung von MIDI-Kommandos über das Windows-API ist recht komplex. Auf Microsofts Plattform für Open-Source-Projekte namens CodePlex findet sich die neuere Bibliothek NAudio, mit deren Hilfe es sehr viel einfacher ist, MIDI-Befehle zum Abspielen eines bestimmten Tons an den internen Synthesizer zu schicken.

Das Einbinden der NAudio.dll geht analog der eingangs beschriebenen Einbindung der Kinect-Bibliothek vonstatten. Per `using NAudio.midi` bindet man den nötigen Namensraum ebenfalls ins Programm ein.

Mit `MidiOut midiOut = new MidiOut(0);` legen Sie ein Objekt für das MIDI-Ausgangsgerät an. Hinter dem Index 0 bei der Initialisierung verbirgt sich der Windows-Synthesizer, da er stets das erste, wenn nicht einzige MIDI-Ausgabegerät darstellt. Zentral für die Ausgabe der MIDI-Kommandos ist der Befehl `midiOut.Send(MidiMessage.StartNote(Note, Velocity, Kanal).RawData);`. Die Variablen geben den Notenwert und die Anschlagsintensität mit Werten zwischen 0 und 127 sowie den MIDI-Kanal von 1 bis 16) an. Auf Kanal 2 liegt beim Synthesizer ein Klavier, auf Kanal 10 ein Schlagzeug. Die meisten anderen Kanäle sind frei beziehungsweise geben ebenfalls Klaviersound wieder. Mit den Werten darf experimentiert werden.

Im Beispiel haben wir uns darauf beschränkt, auf Kanal 2 einen Ton mit voller Lautstärke (also Velocity = 127) auszugeben und nur die Höhe des Tons durch Rechts-Links-Verschiebungen des Fingers zu verändern. Dazu rechnet man die Koordinaten zunächst um. Die Zeile `int y = Convert.ToInt32(joint.Position.Y * 100 + 100);` rechnet den Wert für die Höhe des Fingers in Zentimeter um und erhöht ihn um einen Meter, sodass man in der Mitte, also ungefähr auf Bauchhöhe, Werte um die 100 Zentimeter erhält. Bei den Y-Werten ist das für dieses Beispiel recht willkürlich, denn sie wird nur benötigt, um eine Grenze festzulegen, bei deren Unterschreiten ein Ton ausgelöst werden soll. Anders sieht es mit den X-Werten aus, die den gespielten Ton angeben und sich also zwischen 0 und 127 bewegen müssen. Der Abstand der Tasten auf dem virtuellen Keyboard soll etwa 3,3 Zentimeter betragen, also teilt man den Zentimeter-Wert hier noch durch 3,3 und vergisst dabei die Klammern nicht: `int x = Convert.ToInt32((joint.Position.X * 100 + 100)/3.3);`. Dann stellt man mit `if (x > 127) x = 127;` und `if (x < 0) x = 0;` sicher, dass sich der Wert nicht aus dem für MIDI-Kommandos zulässigen Bereich hinausbewegt.

Wie erwähnt tritt das Ereignis bis zu 30 Mal in der Sekunde auf. So oft will man bestimmt keine Taste auf dem Klavier auslösen, denn das gäbe nur sinnlosen Krach. Deshalb stellt das Programm über ein paar Abfragen des Y-Wertes und die boolesche Variable `playing` sicher, dass ein Ton nur einmal ausgelöst wird und bei Heben des Fingers gestoppt wird. Erst danach wird bei erneutem Senken des Fingers ein neuer Ton ausgelöst.

Das war eigentlich schon alles. Mit dem Finger in der Luft Klavier zu spielen erfordert allerdings einiges an Übung. Aber das ist beim echten Klavier auch nicht anders. Auch ist die Funktionsweise des Programms denkbar simpel, es könnte durchaus noch die Bewegungsgeschwindigkeit auswerten und die ermittelte Anschlagsdynamik über den zweiten Parameter der MIDI-Send-Funktion weitergeben.

Ausblick

Das Beispielprogramm dient nur der Demonstration, wie leicht man die Daten der Kinect auslesen kann. Das Drumherum ist alles andere als ausgereift und dringend verbesserungswürdig. So fehlt die Behandlung möglicher Fehler; auch müssten ausgereifte Programme die Möglichkeit der Auswahl des MIDI-Geräts, des Kanals und Synthesizer-Programms und dergleichen bieten.

Auch wenn das Beispielprogramm alles andere als vollständig ist, zeigt es doch bereits, wie simpel die Gewinnung von Daten aus der Kinect ist. Das gilt übrigens nicht nur für jene des `BodyFrame`-Objekts. Auch auf die Pixeldaten der Kamera oder die Tiefeninformationen des Infrarotsensors kann man auf ähnlich einfache Weise in eigenen Programmen gut zugreifen. Eine Entdeckungsreise durch die verschiedenen Daten-Objekte des SDK mit dem Kinect-Explorer und den Beispielcodes in Visual Studio ist in jedem Fall erhellend und anregend und macht Lust auf eigene Experimente. Versuchen Sie doch mal, mit einer der auf Seite 172 vorgestellten LED-Lichtleiste einen farbigen Punkt zu erzeugen, der der Bewegung eines Fingers folgt! (tig)

Literatur

[1] Oliver Lau, Ätherisch, Handbewegungen erkennen mit dem Leap-Motion-Sensor, c't 20/13, S. 196

[2] Oliver Lau, Natürlich steuern, Einführung in Microsofts Kinect SDK, c't 24/11, S. 184

[3] Oliver Lau, Magic Eye Movies, Bewegte Autostereogramme mit der Microsoft Kinect, c't 19/12, S. 164, www.ct.de/-1673564

[4] Tam Hanna, Microsoft Kinect, Programmierung des Sensorsystems, d.punkt-Verlag Heidelberg, September 2013

[5] Tim Gerber, Ohne Noten, MIDI-Geräte per PC oder Smartphone steuern und konfigurieren, c't 5/13, S. 162

c't Beispielcode und mehr: ct.de/yprt