Tam Hanna

Microsoft KINECT

Programmierung des Sensorsystems

dpunkt.verlag

Tam Hanna befasst sich seit mehr als einer Dekade mit der Programmierung und Anwendung von Computersystemen im mobilen und stationären Bereich. Zusätzlich betreibt sein Unternehmen eine Gruppe von Webseiten über Handcomputer und liefert Inhalte für Fachzeitungen und Fachbücher.

Tam Hanna

Microsoft KINECT®

Programmierung des Sensorsystems

dpunkt.verlag

Tam Hanna
tamhan@tamoggemon.com

Lektorat: Dr. Michael Barabas
Sprachliche Bearbeitung: Thomas Pohlmann
Copy-Editing: Marita Böhm, München
Satz: reemers publishing services gmbh, Krefeld
Herstellung: Birgit Bäuerlein
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN:
Buch 978-3-86490-030-3
PDF 978-3-86491-383-9
ePub 978-3-86491-384-6

1. Auflage 2013

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Vorwort

Ich kam zum Kinect wie die Jungfrau zum Kind: Im Rahmen der Anschaffung einer Xbox 360 für Entwicklungszwecke kam der Sensor als Teil des Bundles mit ins Haus.

In vielerlei Hinsicht ist der Kinect eine Weiterentwicklung des Trends zur direkten Interaktion zwischen Anwender und Computer, der von Touchscreen-Geräten eingeleitet wurde. Berührungsempfindliche Bildschirme ermöglichen die Interaktion durch manuellen Kontakt. Der Kinect erweitert das Interaktionsspektrum in die Bereiche Mimik und Gestik. Deshalb verschob sich mein Interesse nach Abschluss des Xbox-Projekts rasch in Richtung des Sensors.

Dieses Buch möchte Ihnen eine schnelle Einführung in die Welt der Natural User Interfaces geben. Richtig genutzt, erweitert diese Technologie Ihre Produkte um faszinierende neue Einsatzszenarien.

An wen richtet sich dieses Buch?

Die Entwicklung von Anwendungen für den Kinect erfolgt primär mithilfe des von Microsoft angebotenen SDK für Windows. Ein Großteil dieses Buchs befasst sich mit dem Erstellen von Programmen in der .NET-Sprache C#. Diese Beispiele lassen sich ohne allzu großen Aufwand in Visual Basic übersetzen. Die native Programmierung mit C++ wird nicht besprochen.

Obwohl diese Plattform von Microsoft nicht unterstützt wird, ist der Kinect-Sensor auch unter Linux einsetzbar. Daher befasst sich dieses Buch auch mit freenect und OpenNI. Als Entwicklungsumgebung kommt hier das Cross-Platform-Framework Qt zum Einsatz.

Als Leser benötigen Sie keine allzu großen Vorkenntnisse. Wenn Sie die Syntax von C# bzw. C++ verstehen, sollten Sie mit den Beispielen keinerlei Probleme haben. Kenntnisse in Qt sind vorteilhaft.

Aufbau und Lesehinweise

Kinect-Sensoren kommunizieren über Datenströme mit ihren Host-PCs. Dabei kommt immer dasselbe Konzept zum Einsatz. Wenn Sie eine Art von Datenstrom verstanden haben, kommen Sie auch mit allen anderen Datenstromtypen ohne große Probleme zurecht. Deshalb ist es ratsam, zumindest die ersten drei Kapitel am Stück zu lesen. Wer sich für Windows oder Linux nicht interessiert, mag die Ausführungen zur entsprechenden Plattform überspringen.

Die ausgearbeiteten Beispielprojekte zeigen praktische Anwendungen des Sensors. Sie werden beide unter Windows entwickelt. Die besprochenen Konzepte lassen sich aber auch unter Linux einsetzen. Wenn Sie sich von der Thematik nicht angesprochen fühlen, können Sie die beiden Beispielprojekte ohne Konsequenzen überspringen.

Im ersten Kapitel beschäftigen Sie sich mit der Hardware und der Geschichte des Sensors. Das zweite Kapitel beschreibt die Einrichtung des Sensors unter Windows und Linux.

Erste programmiertechnische Gehversuche unternehmen Sie im dritten Kapitel. Dort geht es um die Auswertung des Farbdatenstroms. Die unter Windows beschriebene API zeigt das generelle Konzept der Datenstromverarbeitung. Für Linux erfolgt die Beschreibung anhand der veralteten Version von freenect, die aber nach wie vor weit verbreitet ist.

Das darauf folgende erste Beispielprojekt (Kapitel 4) führt Sie in die Welt der Bildverarbeitung ein. Dort entsteht eine kleine Kameraanwendung, die die eingelesenen Bilder nachschärft und mit einem Kontrastverstärker verbessert.

Im fünften Kapitel beschäftigen Sie sich mit Tiefenströmen und der Korrelation zwischen Tiefen- und Farbdaten. Für Linux-Programmierer ist dieses Kapitel interessant, da es die neue Strom-API der freenect-Bibliothek vorführt. Im dazugehörigen zweiten Beispielprojekt geht es um Methoden zur Rauschminderung im Tiefenstrom. Zusätzlich erhalten Sie eine kurze Vorstellung von Histogrammen.

Skelettverfolgung ist eine der wichtigsten Funktionen des Kinect. In Kapitel 7 wird diese Technologie unter Windows vorgestellt. Kapitel 8 gibt einen Überblick über interessante Funktionen des Kinect-SDK für Windows.

Im neunten Kapitel beschäftigen Sie sich mit dem Mikrofon-Array und der Spracherkennung. Kapitel 10 beschreibt die Gesichtsverfolgungsfunktion. Kapitel 11 stellt Ihnen die OpenNI-API vor. Dabei handelt es sich um einen quelloffenen Standard zur Kommunikation mit Sensoren. Der primäre Anwendungszweck dafür ist die Skelettverfolgung unter Linux.

Abschließend erhalten Sie in Kapitel 12 noch einen Ausblick auf die angekündigte zweite Generation von Kinect-Sensoren und neue Möglichkeiten in der Version 1.7 des Kinect-SDK

Beispielcode

Dieses Buch enthält eine Vielzahl von Codebeispielen[1], die die besprochenen Konzepte für Sie illustriert. Um Ihnen umständliches Hin- und Herblättern zu ersparen, druckt der Verlag die Listings so komplett wie möglich ab. Dadurch werden manche Codeabschnitte mehrfach abgedruckt. Aber es ist meine feste Überzeugung, dass Herumblättern weitaus lästiger wäre. Wenn Sie das anders sehen, würde ich mich über einen Leserbrief freuen!

Über den Autor

Ich befasse mich seit dem Jahr 2004 mit (seinerzeit sogenannten) Handcomputern. Mein Einstieg in die IT-Industrie erfolgte zu einer Zeit, als der Besitz eines Smartphones zur sofortigen sozialen Exkommunikation (internetsüchtig!) führte und Applikationen für Handys noch richtig teuer waren.

Als begeisterter Leser von Fachbüchern (internetsüchtig!!!) ist es mir seit jeher ein Anliegen, ebenfalls Bücher für das interessierte Fachpublikum zu schreiben. Im Laufe der Jahre hat sich meine Tätigkeit so von der reinen Anwendungsentwicklung auch in andere Bereiche wie Beratung und das Verfassen von Fachartikeln verschoben.

Danksagung

Zu guter Letzt – der Brite würde an dieser Stelle »last, but not least« schreiben – sollen hier noch einige Personen aufgelistet werden, ohne deren Hilfe dieses Buch niemals fertig geworden wäre.

Am wichtigsten sind mit Sicherheit Sie, liebe Leserin, lieber Leser. Ohne Sie hätte die Arbeit an diesem Buch keinen Sinn gehabt. Ich danke Ihnen vielmals für Ihr Vertrauen und hoffe, dass das Buch Ihren Wünschen gerecht wird.

Gleich danach möchte ich mich bei meiner Lebensgefährtin bedanken. Ohne ihre bereitwillige Unterstützung hätte sich die Fertigstellung dieses Buchs auf den Sankt-Nimmerleins-Tag verschoben. Deshalb in der ihr eigenen Kürze: »Srdečná vďaka za všetko, Macky!«

Großer Dank gebührt auch dem Team des dpunkt.verlags. Insbesondere danke ich Herrn Dr. Michael Barabas für seine Bereitschaft, dieses Buch mit mir anzugehen. Besonderer Dank gebührt ihm zudem dafür, dass er den Veröffentli-

1. Der in Kapitel 11 vorgestellte Code basiert auf dem von Daniel Roggen erstellten QtKinect-Wrapper. Das vollständige Projekt steht unter der BSD-Lizenz und kann unter https://code.google.com/p/qtkinectwrapper/ heruntergeladen werden.

chungsplan für mich zweimal komplett überarbeitet hat, um die Aktualisierung des Lehrbuchs auf die erst während der Manuskripterstellung erschienenen SDK-Versionen 1.6 und 1.7 zu ermöglichen.

Natürlich dürfen an dieser Stelle auch die Kunden der Tamoggemon Holding k.s. nicht fehlen. Egal, ob Sie bei uns eine Applikation gekauft, Inhalte angefordert oder Beratungsdienstleistungen in Anspruch genommen haben – Sie geben meiner Arbeit Sinn. Danke dafür!

Tam Hanna

Juli 2013
Bratislava, Slowakei

Inhaltsverzeichnis

1 Kinect – was ist das?

Wenn Sie dieses Buch in Händen halten, wissen Sie sehr wahrscheinlich, dass es sich beim Kinect um ein von Microsoft vertriebenes Eingabegerät für Computersysteme handelt.

Aber: Wissen Sie auch, wie die einzelnen Hardwarekomponenten funktionieren? Wissen Sie, was das Projekt Natal war? Oder haben Sie sich schon immer gefragt, ob der Kinect in der Wiener U-Bahn beworben wurde?

Betrachten Sie dieses erste Kapitel nicht als Lehrstoff, sondern als Einführung in die Thematik. Wenn Sie diese Einführung aus Zeitmangel überspringen wollen, lesen wir uns später wieder. Wer wie ich Nebensächlichkeiten liebt, ist hier genau richtig.

1.1 Entfernungen messen

Klassische Entfernungssensoren für Gestenerkennung arbeiteten mit zwei bekannten Schemata: Kontrasterkennung und Signallaufzeit (Time-of-flight).

Die Schwächen der Kontrasterkennung lassen sich an Abbildung 1–1 ablesen. Die graue Farbe des Hemds ist kaum von der weißen Leinwand im Hintergrund zu unterscheiden. Ohne Tiefeninformationen (sprich nur mit Kontrasterkennung) ist es praktisch unmöglich, den Torso des Autors vom Hintergrund zu lösen. Zudem lässt sich der Abstand zum Sensor anhand der Farben allein nicht bestimmen. Auch wenn die Wand 20 Meter von der Kamera entfernt ist, bleibt sie weiß und wird nicht blau – obwohl diese Vorstellung in der Antike weit verbreitet war.

Anders als die Kontrasterkennung liefert die Time-of-flight-Methode echte Tiefendaten. Dabei kommt eine auf den ersten Blick primitive Methode zum Einsatz: Eine Lichtquelle feuert Infrarot- oder Lasersignale ab, die vom Objekt reflektiert und von einem Sensor eingefangen werden.

Abb. 1–1 *Weiß auf grau: das Ende der Kontrasterkennung*

Anhand der Signallaufzeit kann man sodann auf die Entfernung des reflektierenden Punkts schließen. Diese auf den ersten Blick primitive Technologie ist nicht ohne, da der durchschnittliche CMOS-Sensor nicht in der Lage ist, Laufzeiten zu ermitteln.

Der geringe Preis der Hardware wurde erst durch eine Entwicklung von PrimeSense möglich. Der Kinect ist – man mag es angesichts der enormen Marketinganstrengungen von Microsoft gar nicht glauben – streng genommen keine Entwicklung aus Redmond. Die im Sensor zum Einsatz kommende Technologie ist geheim. Es gibt außerhalb der diversen Patente kaum offizielle Beschreibungen, wie das System tatsächlich arbeitet.

Anhand der verfügbaren Informationen hat John MacCormick vom amerikanischen Dickinson College eine sehr interessante Präsentation erstellt, die kostenlos (da mit amerikanischem Steuergeld erstellt) heruntergeladen werden kann.[1]

Die Grundidee ist einfach: Der Kinect sendet das in Abbildung 1–2 gezeigte Punktemuster in die Umwelt. Das Punktemuster durchquert eine spezielle Linse, die die ausgehenden Lichtstrahlen deformiert. Diese Deformation hat den Effekt, dass sich die Form der einzelnen Punkte je nach dem Abstand zum Sensor verändert.

Der Kinect muss die Infrarotszene fotografieren und die Infrarotkügelchen erkennen. Anhand ihrer Form kann er danach auf die Entfernung der einzelnen Objekte schließen. Das Resultat davon ist eine Tiefenmatrix.

1. http://users.dickinson.edu/~jmac/selected-talks/kinect.pdf

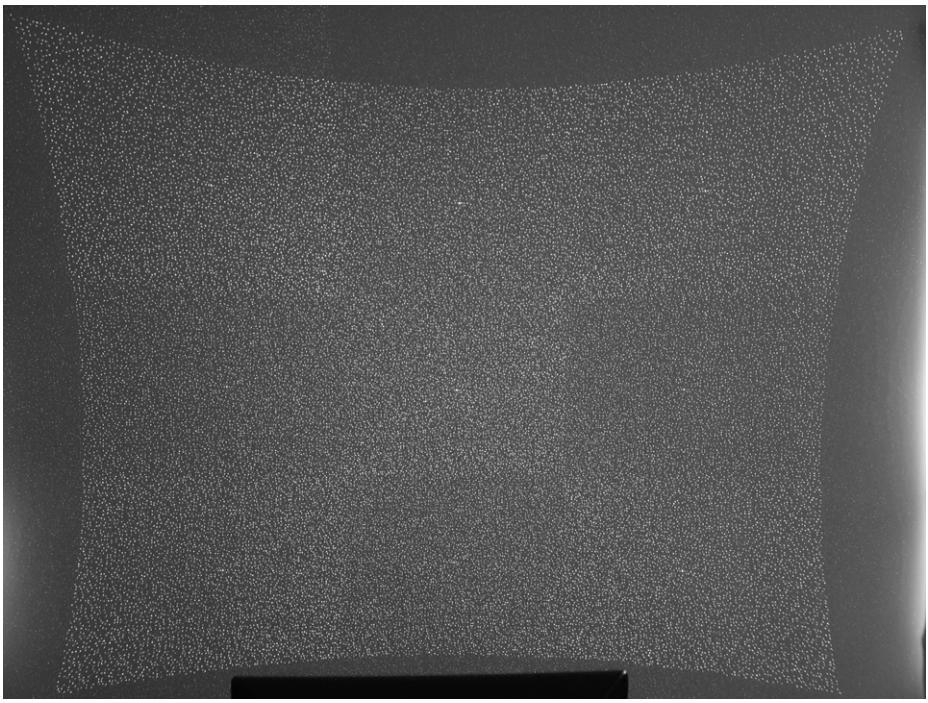

Abb. 1–2 *Die Kinect-Punktwolke (Quelle: http://www.futurepicture.org/?p=129)*

Microsoft bestätigt in MSDN (Microsoft Developer Network), dass der Kinect für die Tiefenerkennung Infrarot verwendet. Es ist also sehr wahrscheinlich, dass die hier gegebene Beschreibung zumindest annähernd korrekt ist.

1.2 Doing it the Microsoft Way

Die Entwicklung der Time-of-flight-Methode war für sich genommen noch kein besonderes Ereignis. Wirklich interessant wurde die Technologie erst, als die vier Gründer von PrimeSense mit einem Prototyp ihres Geräts zu Microsoft pilgerten.

Dort fand man sehr schnell Interesse an der Technologie. Am 30.5.2007 sprach Bill Gates auf der D5-Konferenz beiläufig von einer »Kamera zur Spielesteuerung«.

Aus damaliger Sicht war dies keine besondere Ankündigung. Kameras zur Spielesteuerung gab es schon damals für verschiedene Konsolen. Als Beispiel sei hier die 2003 für die PlayStation ausgelieferte EyeToy-Kamera genannt, oder auch die heute kaum noch bekannte U-Force-Kamera für das Nintendo Entertainment System, die sogar schon mit Infrarot arbeitete.[2]

2. http://en.wikipedia.org/wiki/U-Force

Der Erfolg der Nintendo Wii führte am 7.4.2008 zu einem weiteren »Aufflackern« in der Gerüchteküche: MTV News verlautbarte, dass Microsoft die Nintendo'sche Konsole angreifen wolle. Wie das erfolgen sollte, blieb damals offen.

Erst die Spielemesse E3 brachte 2009 weitere Einblicke. Microsoft kündigte auf seiner alljährlichen Pressekonferenz das Project Natal an. Das Project Natal wurde als »bewegungsempfindlicher« Sensor für die Xbox vorgestellt. Vom Namen Kinect wusste man damals noch nichts.

Für Entwickler war die Verlautbarung im Januar 2010 mit Abstand am wichtigsten. Microsoft kündigte an, dass der Kinect – anders als die diversen Prototypen und Systeme von PrimeSense – keinen integrierten Prozessor haben würde. Die Berechnungen sollten stattdessen von einem der Kerne der Xbox 360 erledigt werden, deren Gesamtleistung durch diese Zusatzbelastung um rund 10% »sinken« würde.

1.3 Reverse Engineering zugunsten von Unix

Die Auslieferung des Kinect am 4. November 2010 war ein signifikanter Erfolg für Microsoft. Das Gerät verkaufte sich wie geschnitten Brot. Es tauchten sogar in der Wiener U-Bahn Werbestände für das Produkt auf.[3]

Leider veröffentlichte Microsoft bei der Erstauslieferung der Hardware keinerlei Pläne für PC-Treiber. Dadurch sah sich das für Elektronikbasteleien bekannte Unternehmen Adafruit Industries inspiriert, eine Prämie von 1000 US-Dollar für den ersten funktionierenden Treiber auszuloben.

Microsoft reagierte darauf mit einer Ankündigung, gegen jede Form von Hacking gerichtlich vorzugehen. Adafruit verdoppelte daraufhin die Belohnung, die Technikpresse schrie auf – und die Sache entpuppte sich als großes Missverständnis.

Microsoft ging es primär um Raubkopien und »Cheaten« in Multiplayerspielen. Solange es nur um den Zugriff auf die vom Sensor gelieferten Daten ging, war das dem Unternehmen egal. Das USB-Interface wurde vielmehr als »offen« beschrieben.

Auch in Redmond war man nicht faul. Das im Juni 2011 erstmals ausgelieferte offizielle SDK wurde sogar von einem eigenen, für den PC-Einsatz zugelassenen Kinect begleitet, den Sie im nächsten Kapitel kennenlernen werden.

An dieser Stelle ist nur eine Sache wichtig: Es gibt zwei konkurrierende SDKs. Das eine stammt von Microsoft, das andere ist quelloffen. Für das eine System entwickelte Anwendungen sind mit dem jeweils anderen inkompatibel.

3. http://tamsppc.tamoggemon.com/2011/12/01/microsoft-kinects-at-austrian-underground-rail/

1.4 Innereien

Die innere Struktur des Kinect ist bei allen Modellen gleich. Eine vom amerikanischen Unternehmen iFixit veröffentlichte schrittweise Zerlegungsbeschreibung erlaubt uns, diese näher anzusehen.[4]

Anstatt die Hauptplatinen einzeln abzudrucken (ein Blick auf die URL http://www.ifixit.com/Teardown/Microsoft+Kinect+Teardown/4066/1 lohnt sich für Interessierte), soll hier das in Abbildung 1–3 gezeigte Flussdiagramm ausreichen. Es zeigt schematisch alle Baugruppen des Kinect.

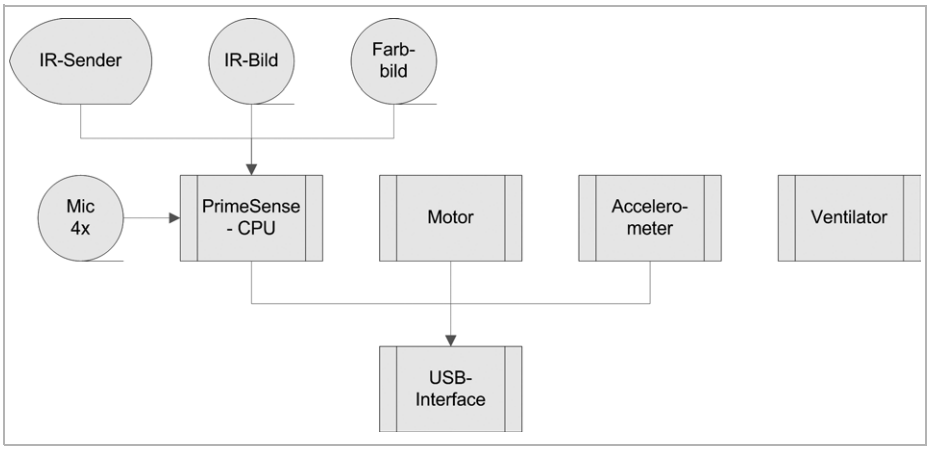

Abb. 1–3 *Der Kinect als Flussdiagramm; die Stromversorgung ist nicht abgebildet.*

Die Kommunikation zwischen Sensor und Host erfolgt über USB. Dass der Steckplatz nicht unbedingt standardisiert aussieht, wird im nächsten Kapitel besprochen.

Neben dem zur Kühlung eingebauten (und vom PC aus nicht steuerbaren) Ventilator gibt es an »simpler« Peripherie zusätzlich ein Accelerometer und einen Motor zur Anpassung der Sichtachse.

Die eigentliche Intelligenz befindet sich in der abgespeckten PrimeSense-CPU. Sie hat die Aufgabe, die verschiedenen im Kinect enthaltenen Sensoren miteinander abzustimmen und als Resultat einen Audio-, einen Tiefen- und einen Farbdatenstrom an den Host zurückzuliefern.

Die Aussage von Microsoft, der Kinect hätte KEINEN Prozessor, ist so zu verstehen, dass die Skelettverfolgung (also der Prozess der Umwandlung von Farb- und Tiefendaten in Skelettkoordinaten) ausschließlich durch den Host erfolgt.

4. http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/

1.5 Fazit

Die vom Kinect gebotenen Interaktionsmöglichkeiten sind – per se – nichts Neues. Die Stärke des Produkts liegt im geringen Preis und der hohen Verfügbarkeit. Nie zuvor gab es ein so leistungsfähiges und gleichwohl günstiges Sensorpaket in einem so praktischen Formfaktor.

2 Erste Schritte

Nach den historischen Ausführungen im ersten Kapitel ist es nun an der Zeit, die Anwendungsentwicklung für den Kinect-Sensor vorzubereiten.[5]

Das Installieren und Anschaffen eines Kinect ist komplizierter, als man auf den ersten Blick annehmen würde. Es gibt viele verschiedene Kinect-Typen, die sich insbesondere beim Anschließen an den Rechner unterschiedlich verhalten.

2.1 Der Kinect-Zoo

Prinzipiell gibt es drei Bauarten für unterschiedliche Einsatzzwecke, die sich in den Funktionen und beim Anschluss an den PC unterscheiden.

Wichtig ist, dass nur der Kinect für Windows mit der SDK-Runtime für Endanwender zusammenarbeitet. Wenn Sie Ihre Anwendung also auf einem PC ausführen wollen, auf dem das SDK nicht installiert ist, so darf der an diesem PC angeschlossene Kinect keinesfalls eine der Xbox-Versionen sein.

2.1.1 Kinect-Bundles

Der wahrscheinlich verbreitetste Sensor ist Teil der diversen Bundles, die aus einer Xbox 360S (slim edition), einem Kinect und eventuell einem oder mehrerer Spiele bestehen.

Die Xbox 360 hat einen speziellen Port auf ihrer Rückseite, der die Stromversorgung und die USB-Kommunikation in nur einem Kabel kombiniert. In Abbildung 2–1 ist der Stecker dargestellt. Dass Sie so einen Port an Ihrem PC nicht finden, ist offensichtlich.

5. In diesem Buch wird der Begriff Sensor auch als Synonym für den Kinect benutzt. Streng genommen ist das nicht ganz korrekt, da der Kinect eine ganze Reihe von Sensoren enthält.

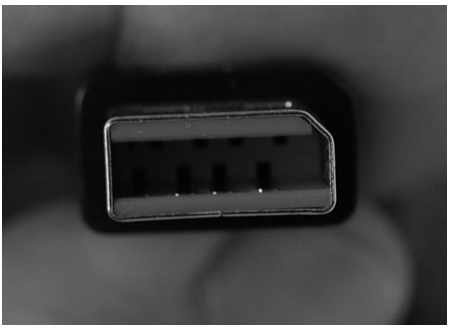

Abb. 2–1 *Das ist kein normierter USB-Stecker – bitte keinesfalls gewaltsam in einen USB-Port von PC oder Notebook rammen!*

Damit auch Besitzer alter Xbox-360-Modelle einen – z. B. von Freunden geliehenen – Kinect verwenden können, offeriert Microsoft ein spezielles Netzteil. Es wird mit dem Kinect verbunden und teilt sich in ein Steckernetzteil und ein normales USB-Kabel auf. Es ist ein von Microsoft wahrscheinlich unerwünschter Nebeneffekt, dass sich dieses Kabelsystem – USB ist nun mal USB – auch mit dem PC verbinden lässt.

Da dieses von Microsoft selbst vertriebene Netzteil nur schwer erhältlich (und ärgerlich teuer) ist, bietet es sich an, auf alternative Produkte zurückzugreifen. Der Autor verwendet für die Arbeit mit dem Kinect das in Abbildung 2–2 gezeigte Produkt, das in diversen Webshops erhältlich ist.

Abb. 2–2 *Billiger, tut es aber genauso …*

Übrigens: Wer ein Bundle aus Xbox 360S und Kinect kauft, um auch für die Xbox zu entwickeln, sollte keinesfalls eines der 4-GB-Pakete erwerben! Die XNA-Entwickleranwendung erfordert zwangsweise das Vorhandensein einer internen (!) Festplatte. Kauft man diese nur schwer erhältliche Komponente später nach, wird es (extrem) teuer.

2.1.2 Kinect für Xbox 360

Kinect Nummero zwei ist seltener. Ihn bekommt man im Handel als Kinect für Xbox 360. Er ist als Upgrade für existierende, also Nicht-S-Xboxen vorgesehen. Im Handel kostet er aktuell (ohne Spiele) etwas mehr als 100 Euro.

Da die klassische Xbox nur über USB-Ports verfügt, kommt dieser Kinect mit einem Netzteil zur Stromversorgung. Deshalb ist er zum Verwenden mit dem PC technisch gut geeignet – wäre da nicht das lizenzrechtliche Problem mit dem SDK bei kommerzieller Verwendung.

2.1.3 Kinect für Windows

Zu guter Letzt gibt es noch den Kinect für Windows. Er ist sozusagen der »Ferrari« unter den Kinect-Sensoren und kostete zum Zeitpunkt der Drucklegung dieses Buchs samt Versand um die 200 Euro. Ihn erkennt man sofort an der etwas anderen Aufschrift auf der Frontseite. Während auf normalen Kinects die Aufschrift »XBOX 360« prangt, steht bei der Windows-Edition »KINECT«.

Er bringt eine neue, »Near Mode« genannte Funktion mit. Sie ist für den Einsatz am Schreibtisch vorgesehen und reduziert die »minimale Abtastentfernung« von 0,8 auf 0,4 Meter. Bis heute ist es nicht völlig klar, ob es sich dabei um eine andere Hardware handelt oder ob die gesteigerte Reichweite ausschließlich durch Softwaremodifikationen entsteht.

Das Kinect-for-Windows-Team schreibt im offiziellen Kinect-Blog[6], dass die Änderungen am Sensor primär durch Firmware-Änderungen und Selektion entstehen. In der Praxis könnte das heißen, dass Microsoft aus der Produktion möglichst präzise arbeitende Sensoren auswählt und diese als Kinect für Windows vertreibt.

Für diese Theorie spricht, dass sehr viele Besitzer eines Kinect für Windows die Performance im Near Mode als unbefriedigend bezeichnen. Anscheinend arbeitet die Windows-Firmware mit »best effort«, während die Xbox-Firmware aus »Sicherheitsgründen« null liefert.

Jedenfalls ist der Kinect für Windows um rund 100 US-Dollar teurer, weil er für den »kommerziellen Einsatz« mit dem Microsoft SDK freigegeben ist. Das liegt vermutlich daran, dass Microsoft auf die Einnahmen durch den Verkauf von Spielen verzichten muss – und diesen »Verlust« in die Hardware einpreist.

---

6. http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/20/near-mode-what-it-is-and-isn-t.aspx

2.2 Das offizielle SDK

Microsoft bietet ein offizielles Kinect-SDK an. Es lässt sich mit diversen IDEs verwenden. Ich benutze in diesem Buch für die .NET-Entwicklung die 2010-Express-Edition von Visual C#.

Im ersten Schritt sollten Sie auf Ihre Workstation die gewünschte Entwicklungsumgebung installieren. Danach muss das SDK beschafft werden. Der nächste Schritt führt Sie also zur Download-Seite von Microsoft[7], wo Sie sich sowohl das SDK als auch das Toolkit herunterladen. Der Großteil der Beispiele wurde mit Version 1.5 und 1.6 des SDK entwickelt. Einen Ausblick auf Neuerungen in Version 1.7 finden Sie in Kapitel 12 »Kinect, der Zweite«.

2.2.1 Windows 7 und Co.

Wenn Sie unter Windows 7 oder Windows 8 entwickeln, können Sie das SDK sofort durch Ausführen der heruntergeladenen Datei installieren.

Nachdem Sie sowohl SDK als auch Developer Toolkit installiert haben, verbinden Sie den Kinect erst mit der Stromversorgung und danach per USB mit dem Computer.

Sobald die Hardwareerkennung abgeschlossen ist, beginnt die grüne LED am Kinect zu blinken. Im Gerätemanager erscheint ein korrekt mit Energie versorgter Sensor wie in Abbildung 2–3 gezeigt als Gruppe von vier USB-Geräten.

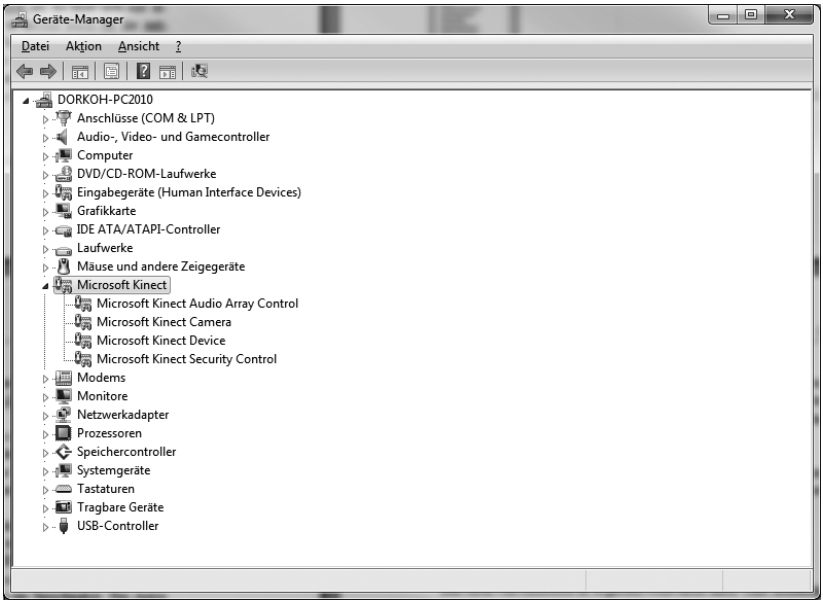

Abb. 2–3 *Die Arbeitsstation hat den Kinect erkannt.*

7. http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx