IMST – Innovationen machen Schulen Top

Informatik kreativ unterrichten

SENSORPROGRAMMIERUG MIT KINECT

ID 853

Dr. Ernestine Bischof

HLW St. Veit an der Glan

St. Veit an der Glan, Februar 2013

INHALTSVERZEICHNIS

INHALTSVERZEICHNIS **2**

ABSTRACT **3**

1 EINLEITUNG **4**

1.1 Schulische Rahmenbedingungen 4

1.2 Lehrplan 4

1.3 Voraussetzungen von LehrerInnenseite 6

1.4 Projektziele 6

1.4.1 Ziele auf SchülerInnenebene 6

1.4.2 Ziele auf LehrerInnenebene 7

1.5 Zeitplan 7

2 DER KINECT-SENSOR **8**

2.1 Grundlegende Daten 8

2.2 Installation 9

3 PROJEKTVERLAUF **10**

3.1 Vorbereitung 10

3.2 Durchführung 10

3.3 Programmbeispiele 11

4 PROJEKTEVALUATION **15**

4.1 SchülerInnenfragebogen 15

4.2 Ergebnisse 15

4.3 Individuelle SchülerInnenrückmeldung und Lehrerinbeobachtung 17

5 REFLEXION UND AUSBLICK **18**

6 LITERATUR **19**

ABSTRACT

Im Rahmen des Themenprogramms „Informatik kreativ unterrichten“ wurden in einer 4. Klasse der Höheren Bundeslehranstalt für wirtschaftliche Berufe (HLW) St. Veit an der Glan 4 Stunden zur Programmierung mit dem Microsoft Kincet-Sensor der Xbox unterrichtet. Die SchülerInnen hatten keine Programmiervorkenntnisse. Unter Anleitung schafften sie es ein vorgegebenes Beispielprogramm aus dem Kinect Developer Toolkit und die Benutzeroberfläche zu verändern.

Die SchülerInnen bekamen einen Einblick in die Möglichkeiten der Programmierung und gaben positives Feedback zu den Unterrichtseinheiten.

Schulstufe:	12. Schulstufe (13. Schulstufe)
Fächer:	Angewandte Informatik
Kontaktperson:	Dr. Ernestine Bischof
Kontaktadresse:	Dr.-Arthur-Lemisch-Straße 15, 9300 St. Veit/Glan

1 EINLEITUNG

Das Projekt wurde an der Höheren Lehranstalt für Wirtschaftliche Berufe (HLW) St. Veit an der Glan in der 12. Schulstufe, Klasse 4AHW, sowie teilweise in einer Klasse der 13. Schulstufe (5BHW) durchgeführt. Schwerpunktthemen dieser Jahrgänge sind vorwiegend Webdesign, Netzwerke und Webanimationen.

Zwei Doppelstunden standen in der 4AHW für das Projekt zur Verfügung. Aufgrund des Interesses einer Kollegin wurde eine Projekteinheit auch in einer Doppelstunde in der 5BHW durchgeführt.

Es handelt sich bei der HLW um eine humanberufliche Schule. Wir haben an unserer Schule drei Schwerpunktzweige: Umweltökonomie und 3. Lebende Fremdsprache sowie Englisch als Arbeitssprache.

Die SchülerInnen hatten bereits die Auszeichnungssprache HTML kennengelernt, aber noch keine Programmiererfahrung.

1.1 Schulische Rahmenbedingungen

Das Fach "Angewandte Informatik" wird an unserer Schule in allen Schwerpunktzweigen von der elften bis zur dreizehnten Schulstufe mit jeweils zwei Wochenstunden unterrichtet. Die Schulklassen werden im Informatikunterricht ab 12 Schülern in zwei Gruppen geteilt. Ursprünglich war die Projektteilnahme der gesamten 4AHW geplant. Durch die Gruppenteilung unterrichten zwei Lehrerinnen die Klasse in Angewandte Informatik. Die beiden Gruppen für das Projekt zusammenzufassen war aus organisatorischen Gründen nicht möglich. Angwandte Informatik wird an unserer Schule generell in Einzelstunden unterrichtet. Durch einen Stundentausch konnte ich für meine Informatik-Gruppe Doppelstunden organisieren. Für die andere Gruppe war dies leider nicht möglich.

Um mehr SchülerInnen am Projekt teilhaben zu lassen wurde noch eine 5. Klasse (13. Schulstufe) in das Projekt einbezogen.

1.2 Lehrplan

Der Lehrplan der HLW sieht, abgesehen von Skript-Sprachen die im Webdesign verwendet werden, keinen Programmierunterricht vor. So war es nötig zusätzliche Stunden für das Projekt „abzuzweigen".

Im allgemeinen Teil des Lehrplans für Angewandte Informatik findet man folgende Erläuterungen zum Lehrstoff:

> *„Aufgrund der Komplexität des Lehrstoffes und der hohen Anforderungen an die Lehrpersonen ist zur Steigerung der Unterrichtsqualität eine Spezialisierung empfehlenswert."*

Die Bezeichung „hohe Anforderungen an die Lehrpersonen" liegt vermutlich darin begründet, dass bis vor kurzem Informatik nicht von Absolventen eines Informatik-Lehramtsstudiums unterrichtet wurde. Auch an unserer Schule wird Informatik teilweise von Absolventen eines BWL-Studiums oder Wirtschaftspädagogikstudiums unterrichtet

Der Lehrstoff für den 4. Jahrgang (12. Schulstufe) verteilt sich wie folgt:

GEBIET	LERNINHALTE

Online-Inhalte	*Usability, Accessibilitiy und Design* • Die Website • Das Layout • Barrierefreies Webdesign *Einführung in HTML und CSS* • Aufbau einer HTML-Datei - Grundlagen • Erstellen von Cascading Style Sheets - Grundlagen *Webeditor* • Webpublishing mit einem aktuellen Webpublishingprogramm • Einbinden von Cascading Style Sheets • Spezielle Techniken, wie z. B. Formulare, Container *Webanimation* • Multimediale Inhalte und Animationen auf Webseiten
Content-Manage-nage-	• Wesentliche Merkmale von CMS • Funktionen von CMS am Beispiel von aktuellen Programmen
Projektmanagement	*Besonderheiten beim IT-Projektmanagement* • Hauptmerkmale von Projekten • Planungsphase, Beauftragungsphase • Projektdurchführungsphase • Projekthandbuch • Controlling • Projektabschlussphase • Workshop

Man sieht, dass Programmierung bei den Lehrinhalten nicht erwähnt wird. Üblicherweise lernen die SchülerInnen der 4. Klassen im Zusammenhang mit Formularen in meinem Unterricht auch immer JavaScript kennen.

Das Projekt zur Sensorprogrammierung wurde somit nicht im Rahmen des Lehrplans durchgeführt.

Auch in der Lehrstoffverteilung der 5. Klassen sucht man vergebens nach Programmierinhalten:

GEBIET	LERNINHALTE
Netzwerke	• Hard- und Softwareanforderungen • Topologien • IP-Adressierung • Arbeiten in einer Client-Server-Umgebung • Grundlagen der Netzwerkverwaltung (Freigabe, Rechte, Netz-werktools) • Internet-Zugänge • Workshop
Rechtl. Grundla-gen	• Domainrecht • Linkrecht • Vertragsrecht • Verschlüsselungen

GEBIET	LERNINHALTE
Auswirkungen der IT	• Individuum • Gesellschaft • Arbeitswelt
Wirtschaft und tertiäre Bildungswege	*Neue Medien und Technologien* • aktuelle Entwicklungen im Hardwarebereich (Geräte, Leistungsstandards usw.) • aktuelle Entwicklungen im Bereich der Softwarelösungen für den PC (neue Versionen, Open Source Produkte usw.) *Aktuelle Standardsoftware* • aktuelle Textverarbeitungs-, Kalkulations- und Präsentationssoftware • Wiederholung und Vertiefung der Kenntnisse anhand komplexer Beispiele • Wissenschaftliche Arbeiten

1.3 Voraussetzungen von LehrerInnenseite

Durch das absolvierte Informatik-Lehramtsstudium der Projektnehmerin waren Programmierkenntnisse von Leherinnenseite vorhanden. In das Thema Sensorprogrammierung (mit Kinect) musste ich mich dennoch erst intensiv einarbeiten. Hilfestellung dabei bot Herr DI Bonifaz Kaufmann von der Alpen-Adria-Universität Klagenfurt.

Eine ursprünglich mit einer HTL geplante Kooperation wurde durch den erhöhten organisatorischen Aufwand nicht eingegangen.

Da Lehrpersonen an unserer Schule grundsätzlich keine Installationsrechte haben, musste ich mir erst vom Netzwerkadministrator die Erlaubnis holen, die benötigte Software zu installieren.

1.4 Projektziele

Im Folgenden werden die geplanten Projektziele beschrieben.

1.4.1 Ziele auf SchülerInnenebene

Globalziel

Die SchülerInnen erhalten einen Einblick in die Programmierung.

Detailziele

Die SchülerInnen erhalten einen Einblick in die Programmierung.

Die SchülerInnen können einfache Elemente im Programmcode verstehen.

Die SchülerInnen können einfache Elemente im Programmcode ändern. (z.B. Farben, Richtung, etc.)

Die SchülerInnen verstehen die Funktionsweise und den Aufbau des Kinect-Sensors.

Die SchülerInnen verstehen wie man auf die Sensoren im Programm zugreifen kann.

Die SchülerInnen erhalten Anregungen sich näher mit Kernthemen der Informatik zu beschäftigen.

1.4.2 Ziele auf LehrerInnenebene

Globalziel

Die Projektnehmerin arbeitet sich in die Programmierung der Kinect ein.

Andere Lehrpersonen der Schule lernen die Möglichkeiten der Sensorprogrammierung kennen.

Detailziel

Die Projektnehmerin lernt die Programmiersprache C# kennen.

Die Projektnehmerin lernt den Kinect-Sensor kennen.

1.5 Zeitplan

Der grobe Zeitplan zu Projektbeginn war wie folgt geplant:

Oktober/November:	Vorbereitung der Unterrichtseinheiten
	Fragebogenerhebung
Dezember:	Durchführung von ca. 4 Unterrichtseinheiten (in zwei Klassen 4AHW, 5BHW)
	Fragebogenerhebung
Jänner/Februar:	Nutzung der Einheiten und angeschafften Hardware in weiteren Klassen
	Fragebogenauswertung, Bericht verfassen

Der Zeitplan wurde wie geplant eingehalten. Durch die Gruppenteilung sind allerdings weniger SchülerInnen und eigentlich nur eine Klasse am Projekt beteiligt. Im Februar wurde die Rohfassung des Berichts fertiggestellt.

Der Fragebogen wurde mit zu bewertenden Aussagen ausgegeben und war ein Selbsteinschätzungsfragebogen.

2 DER KINECT-SENSOR

2.1 Grundlegende Daten

Der Kinect-Sensor ist ursprünglich eine Hardware zur Steuerung der Spielkonsole Xbox 360. Kinect wurde von der Firma Microsoft und der israelischen Firma PrimeSense entwickelt. Xbox-Spieler können durch Körperbewegungen die Spielsoftware bedienen.

Der Kinect-Sensor beinhaltet einen Infrarot-Tiefensensor, eine Farbkamera und ein 3D-Mikrofon. Die Tiefenmessung funktioniert durch ein Punktemuster und hat eine Wiederholrate von 30 Hz. Zur Positionierung der Kamera dient ein kleiner Elektromotor der im Standfuß eingebaut ist. (vgl. Kofler 2011, Borenstein 2012 und Webb & Ashely 2012)

Abbildung 1: Beschreibung des Kinectsensors (vgl. Borenstein 2012, p. 10)

Abbildung 2 zeigt das von der Infrarotkamera abgebildete Muster. Man erkennt den Entfernungsunterschied zwischen der Wand und den davor befindlichen Gegenständen durch einen kleinen „Absatz und Helligkeitsunterschiede.

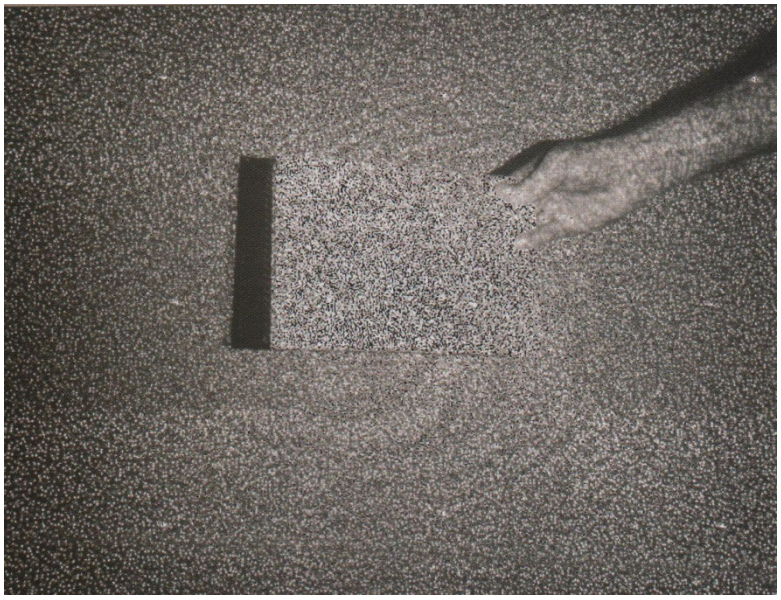


Abbildung 2: Bildpunkte der Infrarotkamera (Borenstein 2012, p. 4)

2.2 Installation

Für die Schule wurden 5 Kinect-Sensoren für den PC (nicht für die Xbox) angekauft. Die Geräte können an der USB-Schnittstelle angeschlossen werden. Die Software ist auf der Microsoft Homepage frei verfügbar. Visual Studio Express kann für den Privat- und Unterrichtsgebrauch gratis aktiviert werden oder 30 Tage kostenlos getestet werden. In diesem Unterkapitel wird beschrieben, wie die Installation auf den Schulcomputern vorgenommen wurde.

1) Installation von Microsoft Visual Studio Express 2012: http://www.microsoft.com/visualstudio/deu/products/visual-studio-express-products
2) Visual Studio schließen
3) Installation von KinectSDK1.6: http://www.microsoft.com/en-us/kinectforwindows/develop/new.aspx
4) Kinect zuerst beim Strom anstecken, dann erst den USB-Stecker beim PC anstecken
5) Kinect Developer Toolkit installieren: http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx
6) Jetzt können bereits vorgefertigte Beispiele ausprobiert werden.

Für den Unterricht wurde ein einfaches Beispiel für die Nutzung des Tiefensensors ausgewählt. Die Dokumentation zum Beispiel ist auf http://msdn.microsoft.com/en-us/library/hh855380 zu finden.

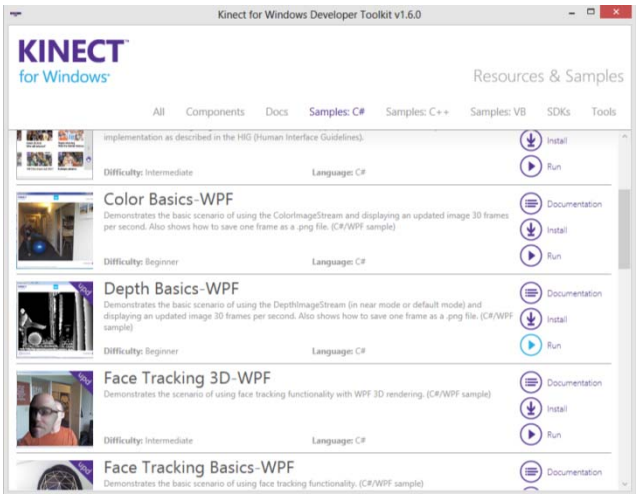


Abbildung 3: Beispiele aus dem Kinect Developer Toolkit

Abbildung 3 zeigt Programmierbeispiele aus dem Kinect Developer Toolkit. Die Beispiele eigenen sich gut dafür im Unterricht verändert oder erweitert zu werden.

3 PROJEKTVERLAUF

3.1 Vorbereitung

Um den Vorbereitungsaufwand auf Lehrerinnen-Seite zu minimieren wurde auf die Unterstützung von Herrn DI Bonifaz Kaufmann zurückgegriffen, da er schon viel Erfahrung mit der Arbeit mit der Kinect sammeln konnte. Die Unterrichtseinheiten wurden gemeinsam vorbereitet und die Kaufentscheidung wurde nach einem beratenden Gespräch erleichtert. Es wurde das Beispiel Depth Basics-WBF (siehe Abbildung 3) als Grundlage gewählt. Somit war bereits fertiger Programmcode vorhanden, der gemeinsam mit den SchülerInnen erweitert werden konnte. Ursprünglich war nur eine gemeinsame Vorbereitung mit Herrn Kaufmann geplant; die Unterrichtseinheiten sollten von der Projektnehmerin durchgeführt werden.

Um die Einheit für die SchülerInnen interessanter zu gestalten kam Herr Kaufmann für eine Doppelstunde an die Schule. Die zweite Unterrichtseinheit wurde von der Projektnehmerin als Vertiefung gestaltet.

Die Schwierigkeit bei der Sensorprogrammierung mit der gegebenen Klasse war, SchülerInnen ohne jegliche Programmiererfahrung in die Programmierung einzuführen. Die SchülerInnen sollten in kurzer Zeit ein kleines Programm gestalten.

Durch die Verwendung bestehenden Programmcodes konnte innerhalb einer Doppelstunde ein für die SchülerInnen motivierendes Ergebnis erzielt werden.

3.2 Durchführung

Die im Folgenden beschriebenen Unterrichtseinheiten wurden in der 4. Klasse (4AHW, Gruppe1) durchgeführt. In der 5. Klasse (5BHW, gesamte Klasse) wurde nur der 1. Teil der Unterrichtseinheit (eine Doppelstunde) durchgenommen.

Die Gruppe der 4. Klasse besteht aus 10 Lernenden (4 Schüler und 6 Schülerinnen). Die 5. Klasse besteht aus 23 Lernenden (4 Schüler und 19 Schülerinnen).

Die erste Unterrichtseinheit war wie folgt aufgebaut:

Die Klasse wurde in Kleingruppen geteilt, sodass jeweils 2-3 Lernende an einem Sensor arbeiteten. Aufgrund meiner bisherigen Vorerfahrungen habe ich die SchülerInnen in geschlechtshomogene Gruppen aufgeteilt. Die SchülerInnen erhielten etwa 20 Minuten lang einen Überblick, wie der Ki-

Abbildung 4: Virtual Dressing Room (http://www.youtube.com/watch?v=1jbvnk1T4vQ)

nect-Sensor aufgebaut ist und welche Anwendungsbeispiele es abseits der Xbox-Spiele, die einige von ihnen kannten, noch gibt. Ein Anwendungsbeispiel für einen Onlineshop, in dem man die Kleider „probieren“ kann, zeigt Abbildung 4.

3.3 Programmbeispiele

Abbildung 5: Kurzvortrag als theoretischer Input zu Projektbeginn

Die SchülerInnen installierten das Beispiel Depth Basics-WBF und öffneten es in Visual Studio. Die grundlegende Funktion des Programms wurde gemeinsam besprochen.

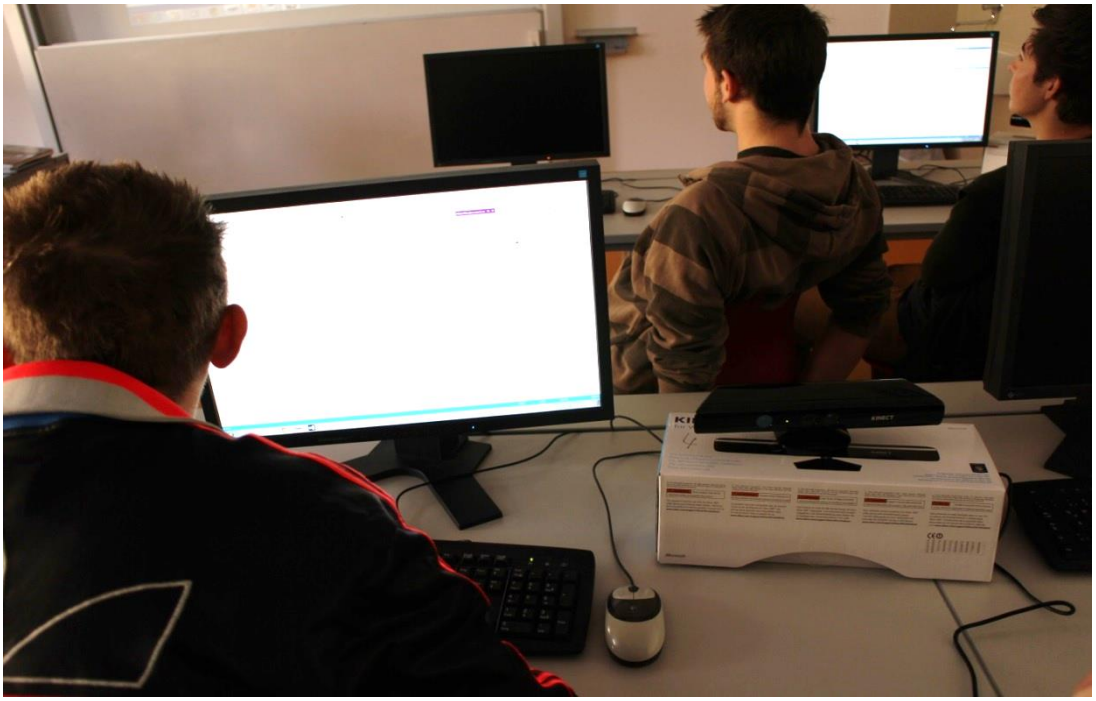


Abbildung 6: Installation des Programmbeispiels

Nachdem der bestehende Programmcode erklärt war, wurden nur in der Methode `SensorDepthFrameReady` Änderungen vorgenommen.

Aufgabe für die SchülerInnen war es das Programm so zu erweitern, dass in einer gewissen Entfernung die Pixel in einer Farbe eingefärbt werden. Die SchülerInnen erinnerten sich sofort an das RGB-Farbmodell, das sie schon in der 3. Klasse bei der Bildbearbeitung sowie in HTML verwendet hatten.

```
int countRedPixels = 0;
    for (int i = 0; i < this.depthPixels.Length; ++i)
    {
       // Get the depth for this pixel
       short depth = depthPixels[i].Depth;

       byte intensity = (byte)(depth >= minDepth && depth <= maxDepth ? depth : 0);

       if (depth >= 1000 && depth <= 1200)
       {
       // Write out blue byte
       this.colorPixels[colorPixelIndex++] = 0;

       // Write out green byte
       this.colorPixels[colorPixelIndex++] = 0;

       // Write out red byte
       this.colorPixels[colorPixelIndex++] = 255;

       countRedPixels++;
       }
       else
       {
       // Write out blue byte
       this.colorPixels[colorPixelIndex++] = intensity;

       // Write out green byte
       this.colorPixels[colorPixelIndex++] = intensity;

       // Write out red byte
       this.colorPixels[colorPixelIndex++] = intensity;
       }

       ++colorPixelIndex;
       }

       countRed.Content = countRedPixels;
```

Der zweite Schritt war, einen Pixelzähler einzubauen. Ab einer gewissen erreichten Pixelanzahl war das erste Level erreicht.

```
         if (countRedPixels > 120000)
         {
               levelCompleted.Visibility = System.Windows.Visibility.Visible;
         }
         else
         {
               levelCompleted.Visibility = System.Windows.Visibility.Hidden;
         }
```

Zusätzlich zur Änderung des Programmcodes musste auch die Benutzeroberfläche im Visual Studio Designer angepasst werden (siehe Abbildung 7). Das Label steht für die jeweilige Pixelanzahl die laufend aktualisiert wird. „Level completed!“ erscheint sobald die vorgegebene Pixelanzahl erreicht wurde (siehe Abbildung 8).

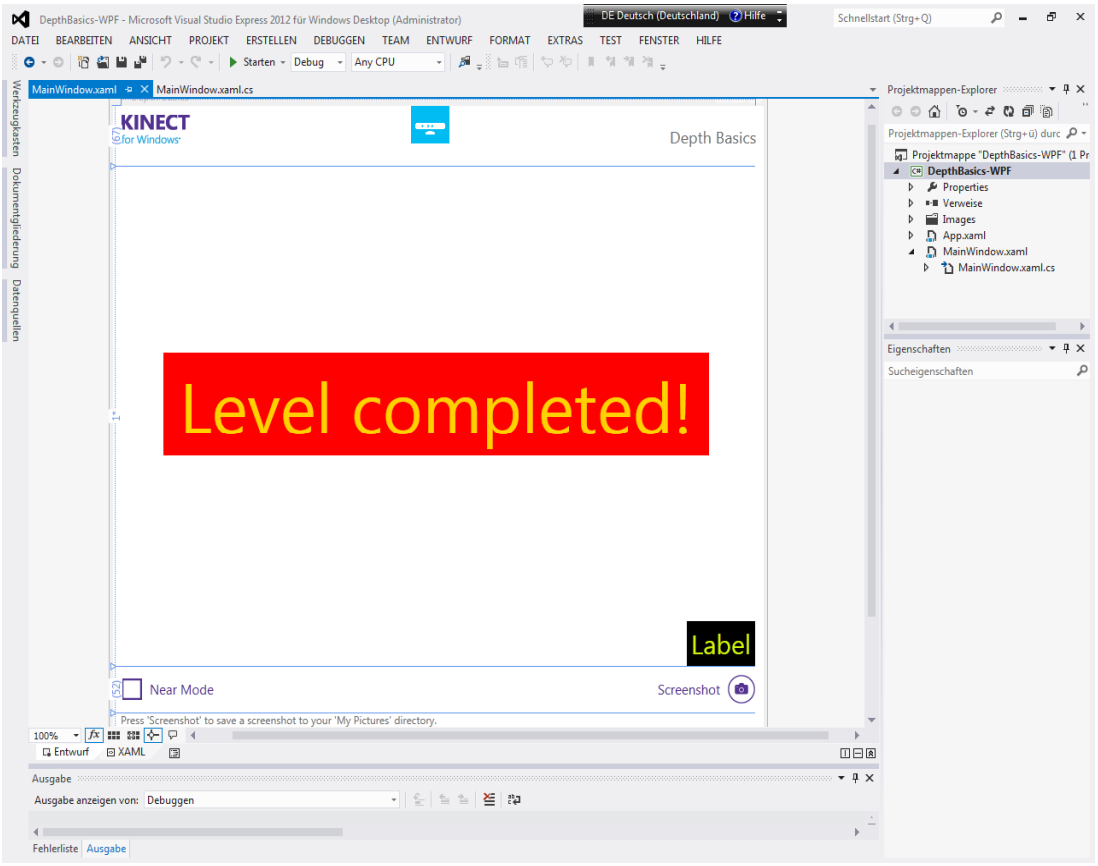


Abbildung 8: Designoberfläche in Visual Studio

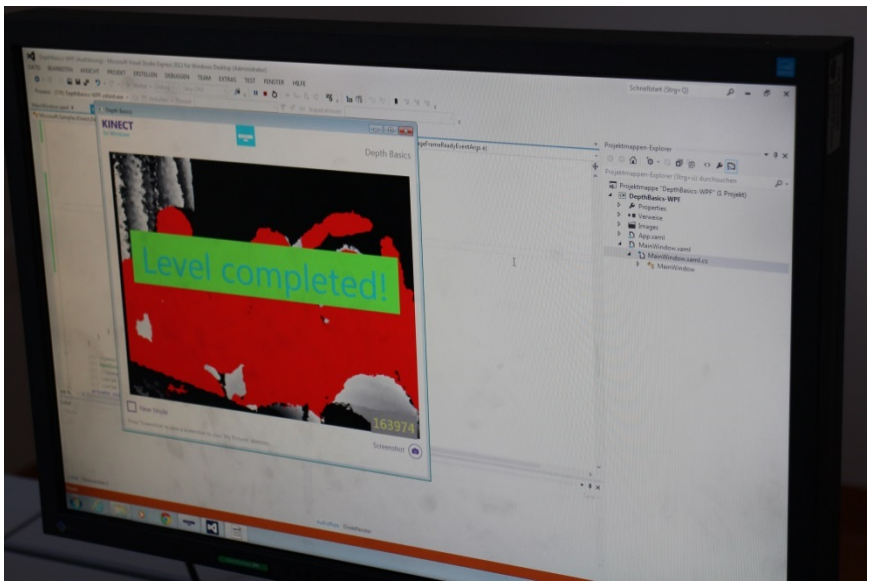


Abbildung 7: Benutzeroberfläche des fertigen Spiel-Ausschnitts

In der zweiten Doppelstunde sollte das Programm zu einem Spiel erweitert werden. Da die SchülerInnen noch keinerlei Programmiererfahrung hatten wurde diese Doppelstunde allerdings dazu verwendet, kleinere Änderungen (z.B. Farben) vorzunehmen und Kontrollstrukturen (IF-Anweisung, FOR-Schleife) nochmals zu wiederholen.

Die SchülerInnen hatten bereits die WENN-Funktion in Excel kennengelernt und erkannten selbst Ähnlichkeiten mit der IF-Anweisug.

Die Abbildungen 9 und 10 zeigen die Schülerinnen und Schüler beim Austesten ihrer Programme.

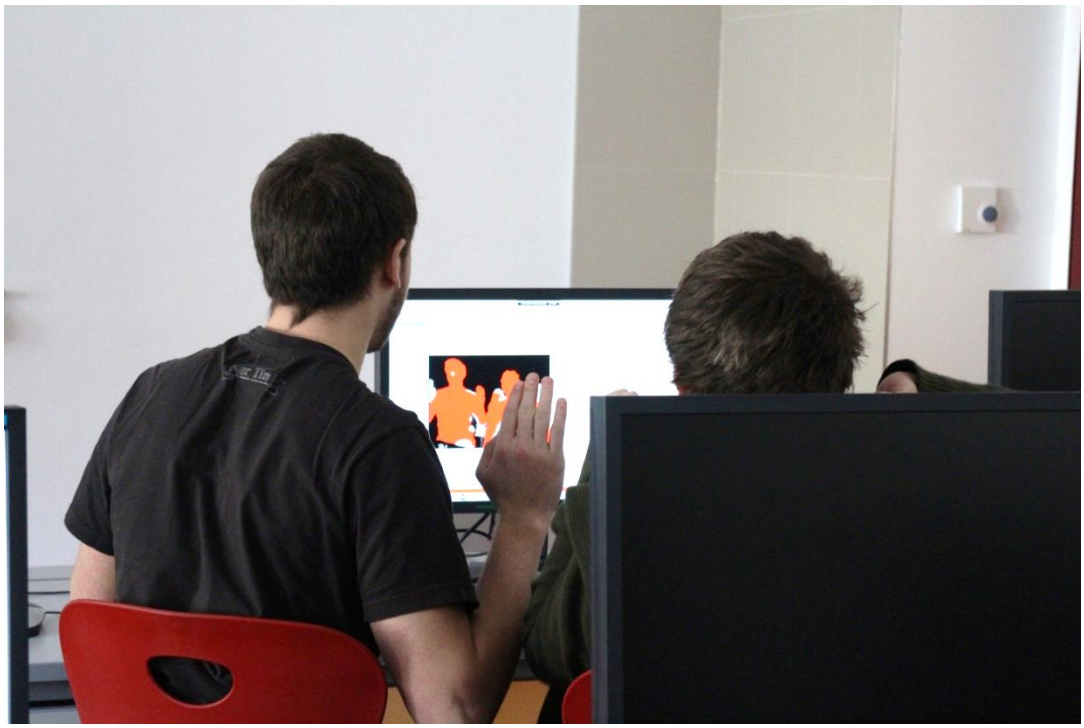


Abbildung 9: Austesten des Programms

Abbildung 10: Das Austesten des Programms macht sichtbar Spaß

4 PROJEKTEVALUATION

Zur Projektevaluation wurde den SchülerInnen vor und nach der Unterrichtseinheit ein Bogen mit Aussagen ausgegeben. An der Befragung nahmen die 10 SchülerInnen (6 weiblich, 4 männlich) der 4AHW teil. Aufgrund der sehr kurzen (und leider auch unvollständigen) Unterrichtseinheit in der 5BHW (19 weiblich, 4 männlich) wurden die SchülerInnen dieser Klasse nicht befragt.

4.1 SchülerInnenfragebogen

Da es an unserer Schule keinen Programmierunterricht, gibt war der Inhalt der Unterrichtseinheiten nur Teil der Benotung im Rahmen der Mitarbeit. Der Inhalt des Projektes sollte kein Teststoff sein. Um für die Projektevaluation dennoch vergleichbare Ergebnisse zu erhalten, wurden die SchülerInnen gebeten ihr Vorwissen einzuschätzen. Nach den Unterrichtseinheiten bekamen sie die gleichen Fragen nochmal gestellt. Die letzte Aussage („Das Arbeiten mit der Kinect hat mir gefallen.“) wurde beim zweiten Fragebogen noch ergänzt. Die SchülerInnen bewerteten die Aussagen anonym mit Schulnoten und gaben an, inwiefern sie auf sie zutreffen.

Selbst-Einschätzung

Vergib Schulnoten von 1 bis 5. (1..trifft zu, 5..trifft gar nicht zu):

Ich kenne den Kincet-Sensor von der Xbox.	
Ich kann einfache Elemente im Programmcode einer Programmiersprache verstehen.	
Ich kann einfache Elemente und somit die Ausführung des Programms verändern.	
Ich verstehe die Funktionsweise der Kinect.	
Ich verstehe wie man die Sensoren „anspricht“.	
Ich weiß, was in der Programmierung eine IF-Anweisung und eine Schleife ist.	
Das Arbeiten mit der Kinect hat mir gefallen.	

4.2 Ergebnisse

Die Tabelle stellt die Ergebnisse der Evaluation dar. Zur besseren Lesbarkeit wird in der Tabelle nur die Nummer der Frage angegeben und die Fragen werden separat aufgelistet.

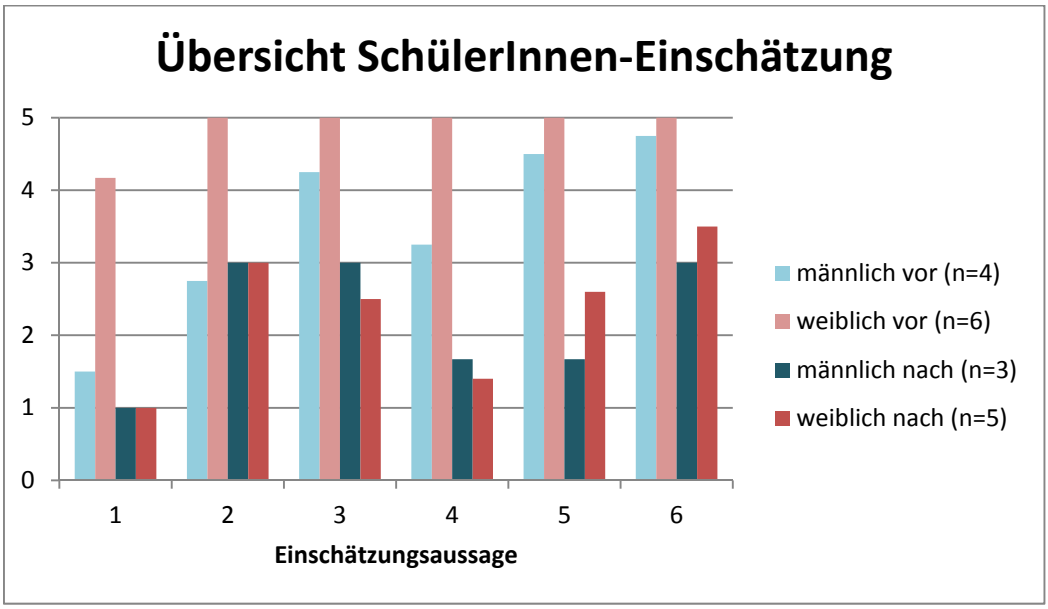
1) Ich kenne den Kincet-Sensor von der Xbox.
2) Ich kann einfache Elemente im Programmcode einer Programmiersprache verstehen.
3) Ich kann einfache Elemente und somit die Ausführung des Programms verändern.
4) Ich verstehe die Funktionsweise der Kinect.

5) Ich verstehe wie man die Sensoren „anspricht“.
6) Ich weiß, was in der Programmierung eine IF-Anweisung und eine Schleife ist.
7) Das Arbeiten mit der Kinect hat mir gefallen.

Bei der zweiten Ausgabe der Fragebögen fehlten leider eine Schülerin und ein Schüler.

Frage	männlich vor (n=4)	weiblich vor (n=6)	männlich nach (n=3)	weiblich nach (n=5)
1	1,5	4,17	1	1
2	2,75	5	3	3
3	4,25	5	3	2,5
4	3,25	5	1,67	1,4
5	4,5	5	1,67	2,6
6	4,75	5	3	3,5
7	-	-	1	1

Das Diagramm stellt die Werte der Tabelle in übersichtlicher Form dar:



Den Burschen der Gruppe war das Gerät Kinect größtenteils bereits sehr gut bekannt. Auf Nachfragen meinten die Burschen, dass sie die Kinect von den Spielen mit der Xbox kennen. Den Mädchen der Gruppe war die Kinect noch eher unbekannt. Nach den Unterrichtseinheiten war das Gerät offensichtlich allen bekannt.

Interessant gestalten sich die Antworten zur Aussage 2. Die Burschen gaben vor dem Projekt eine bessere Bewertung (2,75), als nach dem Projekt (3). Auch hier war eine Rückfrage von Lehrerinnen-Seite nötig. Die Schüler gaben an, dass sie vor der Projektdurchführung unter „Programmcode“ auch an HTML-Code dachten und daher angaben, sich auszukennen. Nach den Unterrichtseinheiten und dem Einblick in den Programmcode von C# war erst klar, was mit Programmcode gemeint ist. Die Aussagen hätten vermutlich umformuliert oder vorher erklärt werden müssen. Die Antworten der Mädchen lagen im erwarteten Bereich (vorher 5, nachher 3).

Aussage 3 zeigt, dass kaum Vorkenntnisse der SchülerInnen vorhanden sind, nach der Durchführung der Unterrichtseinheiten sich die SchülerInnen jedoch in der Lage sehen, einfache Elemente im Pro-

grammcode zu verändern. Diese SchülerInnen-Einschätzung ist aber sicherlich zu hinterfragen. Programmieren kann in so kurzer Zeit nicht gelernt werden.

Die Aussage zur Funktionsweise der Kincet zeigt, dass sich die Burschen schon vor dem Projekt mit deren Funktionsweise beschäftigt haben, die Mädchen noch nicht. Die Aussagen wurden nach den Unterrichtseinheiten im Bereich zwischen 1,4 bis 1,67 bewertet.

Sehr ähnlich sieht die Bewertung der Aussage 5 aus. Hier zeigt sich jedoch, dass sich die Mädchen auch nach den Unterrichtseinheiten weniger zutrauen, als die Burschen.

Die Aussage zum Verständnis von Schleifen und IF-Anweisungen wurden von den SchülerInnen nach der Unterrichtseinheit durchschnittlich bewertet.

Gefallen haben die Unterrichtseinheiten allen Schülerinnen und Schülern.

4.3 Individuelle SchülerInnenrückmeldung und Lehrerinbeobachtung

Die Beobachtung der SchülerInnen während der Unterrichtseinheit hat gezeigt, dass alle Gruppen mit Begeisterung gearbeitet haben. Die Aufteilung in geschlechtshomogene Gruppen hat gezeigt, dass auch in diesen Gruppen einige Personen dominanter auftreten als andere. In einer Burschengruppe gab es immer wieder Diskussionen zwischen den beiden, wie man etwas richtig macht.

Die Arbeit in den Mädchengruppen schien harmonischer. Einige waren dennoch dominanter als andere. Beobachtungen wie sie in „Genderaspekte in der Medienbildung“ (vgl. G@me 2008) berichtet werden, konnten in der Gruppe nicht gemacht werden. Die Schülerinnen zeigten keine höheren kreativen Interessen als die Schüler.

Die verbalen Rückmeldungen der SchülerInnen waren sehr positiv. Besonders der Besuch eines Experten wurde von den Lernenden positiv erwähnt. Sie waren stolz, dass ein Experte an unsere Schule kam um speziell nur mit ihrer Gruppe zu arbeiten.

Die SchülerInnen haben auch einen Bericht für die Schulhomepage bzw. den Jahresbericht verfasst:

> *„Am 13.12.2012 kam Herr DI Bonifaz Kaufmann vom Institut für Informatik-Systeme der Alpen-Adria-Universität Klagenfurt zu uns in die Schule, um uns das System des Kinectsensors näher zu bringen. Durch seine fachspezifische Kompetenz erlernten wir das Grundsystem der Kinect sehr schnell, sodass wir bald darauf unser eigenes Spiel programmierten. Zum ersten Mal bekamen wir einen Einblick in die Programmiersprache C#, welche auf den ersten Blick sehr kompliziert wirkt, doch in Wirklichkeit sehr logisch ist. Nachdem wir unser Spiel programmiert hatten, probierten wir es aus. Bei jedem funktionierte es einwandfrei. Es war ein riesen Spaß. Wir bedanken uns bei Frau Prof. Bischof und Herrn DI Kaufmann für diesen Einblick in die Programmierung der Kinect.“* Simon Falger, Thomas Fercher, Patrick Jury (4AHW)

5 REFLEXION UND AUSBLICK

Die Projektdurchführung erfolgte im Bewusstsein, dass „Programmieren lernen“ in zwei Doppelstunden nicht möglich ist. Es bestand somit das Ziel, den SchülerInnen einen Einblick in die Programmierung zu bieten und zugleich die Hoffnung, dass die SchülerInnen einfache Elemente der Programmierung verstehen. Vergleicht man die Bewertung der SchülerInnen vor und nach den Unterrichtseinheiten, so schätzen sie ihre Kenntnisse nach Projektdurchführung besser ein. Über den tatsächlichen Erkenntnisgewinn der SchülerInnen kann dennoch schwer eine Aussage getroffen werden, da ein konventionelles Abtesten des vermittelten Stoffs bewusst vermieden wurde um die SchülerInnen in entspannter Atmosphäre und ohne Prüfungsdruck arbeiten zu lassen. Auch wenn die SchülerInnen angeben sich Änderungen im Programm zuzutrauen, würden sie dazu wohl noch Anleitung von Lehrerinnenseite benötigen.

Auf Lehrerinnenebene war die Einarbeitung in die Sensorprogrammierung jedenfalls sehr interessant und durch die Unterstützung von außen massiv erleichtert. Zwei Kolleginnen wurden im Detail über das Projekt informiert und interessierten sich für die erstellten Programme. Mit einer Kollegin wurde vereinbart in einer Doppelstunde die 5. Klasse am Projekt teilhaben zu lassen.

Auf Schulebene kann berichtet werden, dass das Projekt auch beim Informationsabend im Jänner den interessierten Eltern aber auch KollegInnen vorgestellt wurde.

Ich persönlich sehe den Erfolg des Projektes darin, dass die SchülerInnen ohne Hinweise von Lehrpersonen Querverbindungen zu anderen bereits in Informatik behandelten Themen fanden. Sie erkannten beispielsweise den Zusammenhang zwischen WENN-Funktionen in der Tabellenkalkulation und der IF-Anweisung. Auch die weitere Anwendungsform der RGB-Farbwerte im Programmcode lies die SchülerInnen Querverbindungen bilden.

Das Projekt war eine nette Erfahrung für Lehrerin und SchülerInnen. Der hohe Vorbereitungsaufwand für die vier Unterrichtsstunden, und der im Vergleich dazu relativ geringe Output zeigen allerdings, dass es an einer humanberuflichen Schule für SchülerInnen ohne Programmierkenntnisse nur bedingt sinnvoll ist ein so komplexes Thema zu behandeln. Für den zukünftigen Unterricht werde ich wohl weiterhin die Programmiereinführung mit JavaScript vornehmen, da dies auch thematisch besser in das Unterrichtskonzept passt.

6 LITERATUR

Buch:

Borenstein, Greg (2012). *Making Things See*. Sebastopol: O'Reilly.

Webb, Jarrett & Ashley, James (2012). *Beginning Kinect Programming with Microsoft Kinect SDK*. New York: Apress Springer.

G@me 2008: *Gender Awareness in Media Education – Genderaspekte in der Medienbildung. Zusammenfassung von Berichten und Untersuchungen*. Projektbericht.

Internet:

Lehrplan IOM/AINF Humanberufliche Schulen: http://www.abc.berufsbildendeschulen.at (Stand vom 4.2.2013).

Kofler, Matthias (2011). Inbetriebnahme und Untersuchung des Kinect Sensors. Online unter http://rrt.fh-wels.at/publications/technical_papers/MP1_Kinect_Kofler_2011.pdf [1.2.2013]

ANHANG

Programmcode (entnommen vom Kinect Developer Toolkit und erweitert)

```
//------------------------------------------------------------------------------
// <copyright file="MainWindow.xaml.cs" company="Microsoft">
//     Copyright (c) Microsoft Corporation.  All rights reserved.
// </copyright>
//------------------------------------------------------------------------------

namespace Microsoft.Samples.Kinect.DepthBasics
{
    using System;
    using System.Globalization;
    using System.IO;
    using System.Windows;
    using System.Windows.Media;
    using System.Windows.Media.Imaging;
    using Microsoft.Kinect;

    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        /// <summary>
        /// Active Kinect sensor
        /// </summary>
        private KinectSensor sensor;

        /// <summary>
        /// Bitmap that will hold color information
        /// </summary>
        private WriteableBitmap colorBitmap;

        /// <summary>
        /// Intermediate storage for the depth data received from the camera
        /// </summary>
        private DepthImagePixel[] depthPixels;

        /// <summary>
        /// Intermediate storage for the depth data converted to color
        /// </summary>
        private byte[] colorPixels;

        /// <summary>
        /// Initializes a new instance of the MainWindow class.
        /// </summary>
        public MainWindow()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Execute startup tasks
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void WindowLoaded(object sender, RoutedEventArgs e)
        {

            levelCompleted.Visibility = System.Windows.Visibility.Hidden;

            // Look through all sensors and start the first connected one.
            // This requires that a Kinect is connected at the time of app startup.
            // To make your app robust against plug/unplug,
            // it is recommended to use KinectSensorChooser provided in Microsoft.Kinect.Toolkit
            foreach (var potentialSensor in KinectSensor.KinectSensors)
            {
                if (potentialSensor.Status == KinectStatus.Connected)
                {
                    this.sensor = potentialSensor;
                    break;
                }
            }

            if (null != this.sensor)
            {
                // Turn on the depth stream to receive depth frames
                this.sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);

                // Allocate space to put the depth pixels we'll receive
```

```
                this.depthPixels = new DepthImagePixel[this.sensor.DepthStream.FramePixelDataLength];

                // Allocate space to put the color pixels we'll create
                this.colorPixels = new byte[this.sensor.DepthStream.FramePixelDataLength * sizeof(int)];

                // This is the bitmap we'll display on-screen
                this.colorBitmap          =          new          WriteableBitmap(this.sensor.DepthStream.FrameWidth,
this.sensor.DepthStream.FrameHeight, 96.0, 96.0, PixelFormats.Bgr32, null);

                // Set the image we display to point to the bitmap where we'll put the image data
                this.Image.Source = this.colorBitmap;

                // Add an event handler to be called whenever there is new depth frame data
                this.sensor.DepthFrameReady += this.SensorDepthFrameReady;

                // Start the sensor!
                try
                {
                    this.sensor.Start();
                }
                catch (IOException)
                {
                    this.sensor = null;
                }
            }

            if (null == this.sensor)
            {
                this.statusBarText.Text = Properties.Resources.NoKinectReady;
            }
        }

        /// <summary>
        /// Execute shutdown tasks
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void WindowClosing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            if (null != this.sensor)
            {
                this.sensor.Stop();
            }
        }

        /// <summary>
        /// Event handler for Kinect sensor's DepthFrameReady event
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void SensorDepthFrameReady(object sender, DepthImageFrameReadyEventArgs e)
        {
            using (DepthImageFrame depthFrame = e.OpenDepthImageFrame())
            {
                if (depthFrame != null)
                {
                    // Copy the pixel data from the image to a temporary array
                    depthFrame.CopyDepthImagePixelDataTo(this.depthPixels);

                    // Get the min and max reliable depth for the current frame
                    int minDepth = depthFrame.MinDepth;
                    int maxDepth = depthFrame.MaxDepth;

                    // Convert the depth to RGB
                    int colorPixelIndex = 0;

                    int countRedPixels = 0;
                    for (int i = 0; i < this.depthPixels.Length; ++i)
                    {
                        // Get the depth for this pixel
                        short depth = depthPixels[i].Depth;

                        // To convert to a byte, we're discarding the most-significant
                        // rather than least-significant bits.
                        // We're preserving detail, although the intensity will "wrap."
                        // Values outside the reliable depth range are mapped to 0 (black).

                        // Note: Using conditionals in this loop could degrade performance.
                        // Consider using a lookup table instead when writing production code.
                        // See the KinectDepthViewer class used by the KinectExplorer sample
                        // for a lookup table example.
                        byte intensity = (byte)(depth >= minDepth && depth <= maxDepth ? depth : 0);

                        if (depth >= 1000 && depth <= 1200)
```

```
                    {
                        // Write out blue byte
                        this.colorPixels[colorPixelIndex++] = 0;

                        // Write out green byte
                        this.colorPixels[colorPixelIndex++] = 0;

                        // Write out red byte
                        this.colorPixels[colorPixelIndex++] = 255;

                        countRedPixels++;
                    }
                    else
                    {
                        // Write out blue byte
                        this.colorPixels[colorPixelIndex++] = intensity;

                        // Write out green byte
                        this.colorPixels[colorPixelIndex++] = intensity;

                        // Write out red byte
                        this.colorPixels[colorPixelIndex++] = intensity;
                    }


                    // We're outputting BGR, the last byte in the 32 bits is unused so skip it
                    // If we were outputting BGRA, we would write alpha here.
                    ++colorPixelIndex;
                }

                countRed.Content = countRedPixels;

                if (countRedPixels > 120000)
                {
                    levelCompleted.Visibility = System.Windows.Visibility.Visible;
                }
                else
                {
                    levelCompleted.Visibility = System.Windows.Visibility.Hidden;
                }

                // Write the pixel data into our bitmap
                this.colorBitmap.WritePixels(
                    new Int32Rect(0, 0, this.colorBitmap.PixelWidth, this.colorBitmap.PixelHeight),
                    this.colorPixels,
                    this.colorBitmap.PixelWidth * sizeof(int),
                    0);
            }
        }
    }

    /// <summary>
    /// Handles the user clicking on the screenshot button
    /// </summary>
    /// <param name="sender">object sending the event</param>
    /// <param name="e">event arguments</param>
    private void ButtonScreenshotClick(object sender, RoutedEventArgs e)
    {
        if (null == this.sensor)
        {
            this.statusBarText.Text = Properties.Resources.ConnectDeviceFirst;
            return;
        }

        // create a png bitmap encoder which knows how to save a .png file
        BitmapEncoder encoder = new PngBitmapEncoder();

        // create frame from the writable bitmap and add to encoder
        encoder.Frames.Add(BitmapFrame.Create(this.colorBitmap));

        string time = System.DateTime.Now.ToString("hh'-'mm'-'ss", CultureInfo.CurrentUICulture.DateTimeFormat);

        string myPhotos = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);

        string path = Path.Combine(myPhotos, "KinectSnapshot-" + time + ".png");

        // write the new file to disk
        try
        {
            using (FileStream fs = new FileStream(path, FileMode.Create))
            {
                encoder.Save(fs);
            }
```

```
                this.statusBarText.Text = string.Format("{0} {1}", Properties.Resources.ScreenshotWriteSuccess,
path);
            }
            catch (IOException)
            {
                this.statusBarText.Text = string.Format("{0} {1}", Properties.Resources.ScreenshotWriteFailed, path);
            }
        }

        /// <summary>
        /// Handles the checking or unchecking of the near mode combo box
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void CheckBoxNearModeChanged(object sender, RoutedEventArgs e)
        {
            if (this.sensor != null)
            {
                // will not function on non-Kinect for Windows devices
                try
                {
                    if (this.checkBoxNearMode.IsChecked.GetValueOrDefault())
                    {
                        this.sensor.DepthStream.Range = DepthRange.Near;
                    }
                    else
                    {
                        this.sensor.DepthStream.Range = DepthRange.Default;
                    }
                }
                catch (InvalidOperationException)
                {
                }
            }
        }
    }
}
```

Benutzeroberfläche (entnommen aus dem Kinect Developer Toolkit und erweitert)

```
<Window x:Class="Microsoft.Samples.Kinect.DepthBasics.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Depth Basics" Height="735" Width="770" Loaded="WindowLoaded" Closing="WindowClosing">

    <Window.Resources>
        <SolidColorBrush x:Key="MediumGreyBrush" Color="#ff6e6e6e"/>
        <SolidColorBrush x:Key="KinectPurpleBrush" Color="#ff52318f"/>
        <SolidColorBrush x:Key="KinectBlueBrush" Color="#ff00BCF2"/>
        <Style TargetType="{x:Type Image}">
            <Setter Property="SnapsToDevicePixels" Value="True"/>
        </Style>
        <Style TargetType="{x:Type Button}" x:Key="SnapshotButton" >
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type Button}">
                        <Grid>
                            <StackPanel Orientation="Horizontal" Background="Transparent">
                                <TextBlock x:Name="SnapText" Text="{TemplateBinding Content}" TextAlignment="Left"
VerticalAlignment="Center" Foreground="{StaticResource KinectPurpleBrush}" FontSize="15" />
                                <Grid Margin="9,0,0,0">
                                    <Image x:Name="SnapNormal" Source="Images\SnapNormal.png" Stretch="None" Horizon-
talAlignment="Center"/>
                                    <Image x:Name="SnapHover" Source="Images\SnapHover.png" Stretch="None" Horizonta-
lAlignment="Center" Visibility="Collapsed"/>
                                </Grid>
                            </StackPanel>
                        </Grid>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsMouseOver" Value="true">
                                <Setter Property="Visibility" Value="Collapsed" TargetName="SnapNormal"/>
                                <Setter Property="Visibility" Value="Visible" TargetName="SnapHover"/>
                                <Setter Property="Foreground" Value="{StaticResource KinectBlueBrush}" Target-
Name="SnapText"/>
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
        <Style TargetType="{x:Type CheckBox}" x:Key="SquareCheckBox" >
            <Setter Property="Template">
                <Setter.Value>
```

```
                    <ControlTemplate TargetType="{x:Type CheckBox}">
                        <Grid>
                            <StackPanel Orientation="Horizontal" Background="Transparent">
                                <Grid x:Name="SquareCheckBoxChecked">
                                    <Image  x:Name="CheckedNormal"  Source="Images\CheckedNormal.png"  Stretch="None"
HorizontalAlignment="Center"/>
                                    <Image x:Name="CheckedHover" Source="Images\CheckedHover.png" Stretch="None" Hor-
izontalAlignment="Center" Visibility="Collapsed"/>
                                </Grid>
                                <Grid x:Name="SquareCheckBoxUnchecked" Visibility="Collapsed">
                                    <Image        x:Name="UncheckedNormal"        Source="Images\UncheckedNormal.png"
Stretch="None" HorizontalAlignment="Center"/>
                                    <Image  x:Name="UncheckedHover"  Source="Images\UncheckedHover.png"  Stretch="None"
HorizontalAlignment="Center" Visibility="Collapsed"/>
                                </Grid>
                                <TextBlock  x:Name="SquareCheckBoxText"  Text="{TemplateBinding  Content}"  TextAlign-
ment="Left"     VerticalAlignment="Center"   Foreground="{StaticResource   KinectPurpleBrush}"   FontSize="15"     Mar-
gin="9,0,0,0"/>
                            </StackPanel>
                        </Grid>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsChecked" Value="false">
                                <Setter Property="Visibility" Value="Collapsed" TargetName="SquareCheckBoxChecked"/>
                                <Setter Property="Visibility" Value="Visible" TargetName="SquareCheckBoxUnchecked"/>
                            </Trigger>
                            <Trigger Property="IsMouseOver" Value="true">
                                <Setter Property="Visibility" Value="Collapsed" TargetName="CheckedNormal"/>
                                <Setter Property="Visibility" Value="Collapsed" TargetName="UncheckedNormal"/>
                                <Setter Property="Visibility" Value="Visible" TargetName="CheckedHover"/>
                                <Setter Property="Visibility" Value="Visible" TargetName="UncheckedHover"/>
                                <Setter   Property="Foreground"   Value="{StaticResource   KinectBlueBrush}"   Target-
Name="SquareCheckBoxText"/>
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
    </Window.Resources>

    <Grid Name="layoutGrid" Margin="10 0 10 0">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <DockPanel Grid.Row="0" Margin="0 0 0 20">
            <Image DockPanel.Dock="Left" Source="Images\Logo.png" Stretch="None" Margin="0 10 0 5"/>
            <TextBlock    DockPanel.Dock="Right"    Margin="0    0    -1    0"    VerticalAlignment="Bottom"    Fore-
ground="{StaticResource MediumGreyBrush}" FontFamily="Segoe UI" FontSize="18">Depth Basics</TextBlock>
            <Image Grid.Column="1" Source="Images\Status.png" Stretch="None" HorizontalAlignment="Center" Margin="0 0
0 5"/>
        </DockPanel>
        <Viewbox Grid.Row="1" Stretch="Uniform" HorizontalAlignment="Center"/>
        <Button  Grid.Row="2"  Style="{StaticResource  SnapshotButton}"  Content="Screenshot"  Height="Auto"  Horizonta-
lAlignment="Right"      VerticalAlignment="Center"      Margin="10      10      0      10"      Name="buttonScreenshot"
Click="ButtonScreenshotClick" />
        <CheckBox  Grid.Row="2"  Style="{StaticResource  SquareCheckBox}"  Content="Near  Mode"  Height="Auto"  Horizonta-
lAlignment="Left"      VerticalAlignment="Center"      Margin="0      10      10      10"      Name="checkBoxNearMode"
Checked="CheckBoxNearModeChanged" Unchecked="CheckBoxNearModeChanged"/>
        <StatusBar  Grid.Row="3"  HorizontalAlignment="Stretch"  Name="statusBar"  VerticalAlignment="Bottom"  Back-
ground="White" Foreground="{StaticResource MediumGreyBrush}">
            <StatusBarItem Padding="0 0 0 10">
                <TextBlock Name="statusBarText" Margin="-1 0 0 0">Press 'Screenshot' to save a screenshot to your 'My
Pictures' directory.</TextBlock>
            </StatusBarItem>
        </StatusBar>
        <Label  x:Name="countRed"  Content="Label"  HorizontalAlignment="Right"  Grid.Row="1"  VerticalAlignment="Bottom"
RenderTransformOrigin="3.934,-1.773" FontSize="30" Background="Black" Foreground="#FFD2F10A" Panel.ZIndex="1"/>
        <Image    x:Name="Image"    HorizontalAlignment="Left"    Height="560"    Grid.Row="1"    VerticalAlignment="Top"
Width="742"/>
        <Label  x:Name="levelCompleted"  Content="Level  completed!"  HorizontalAlignment="Left"  Margin="49,209,0,0"
Grid.Row="1" VerticalAlignment="Top" FontSize="80" Width="639" Background="Red" Foreground="#FFFFD100" HorizontalCon-
tentAlignment="Center"/>
    </Grid>
</Window>
```