Kinect

Seminar *Objektorientiertes Programmieren mit .NET und C#*

Christoph Ihrke

Institut für Informatik
Software & Systems Engineering

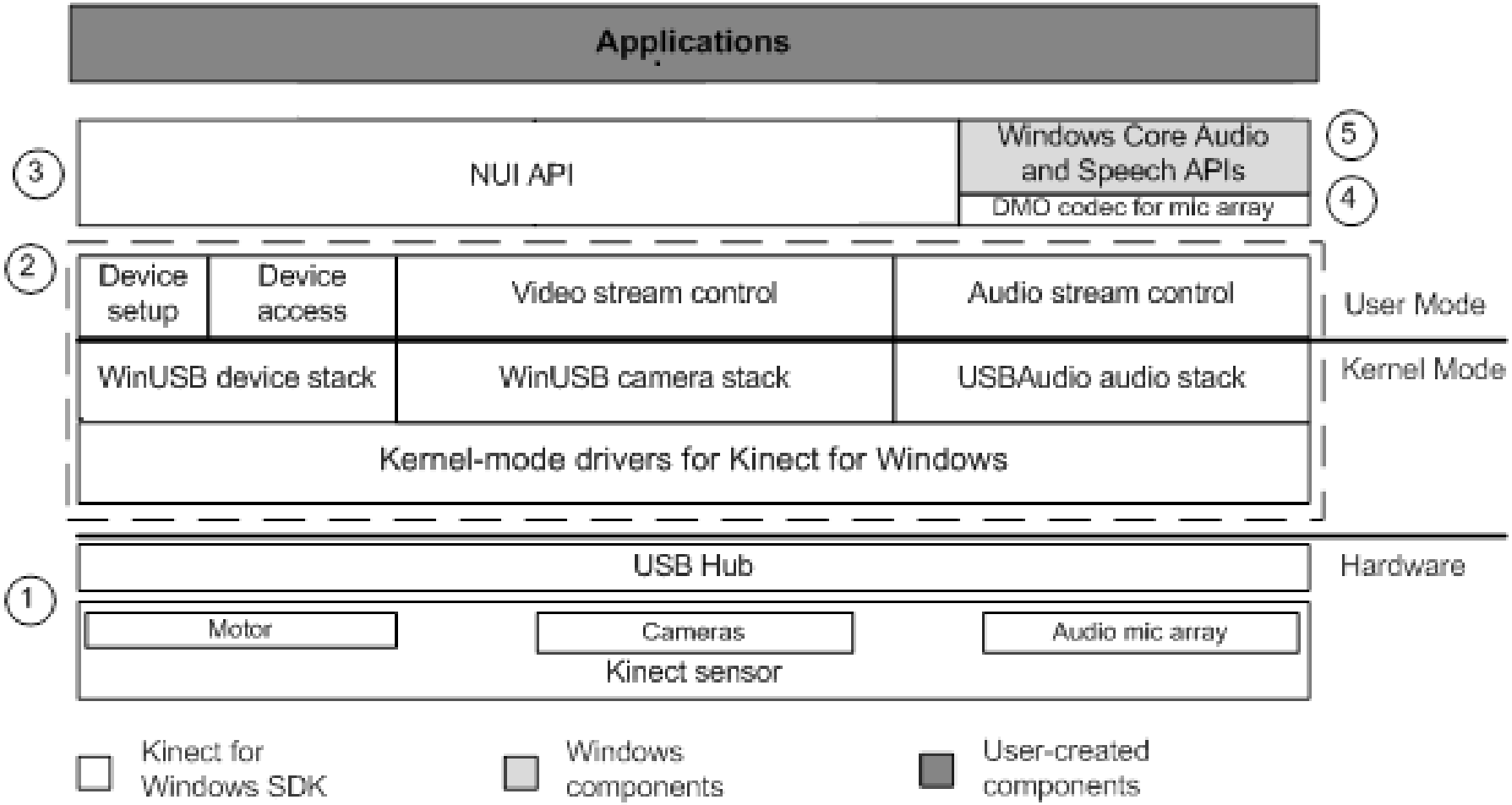
Agenda

- Historie
- SDK Aufbau
- Hardware (Kinectsensor)
- NUI API
  - Datenströme
  - Skeletal Tracking
  - Windows Core Audio & Speech API
- Anwendungsgebiete
- Fazit

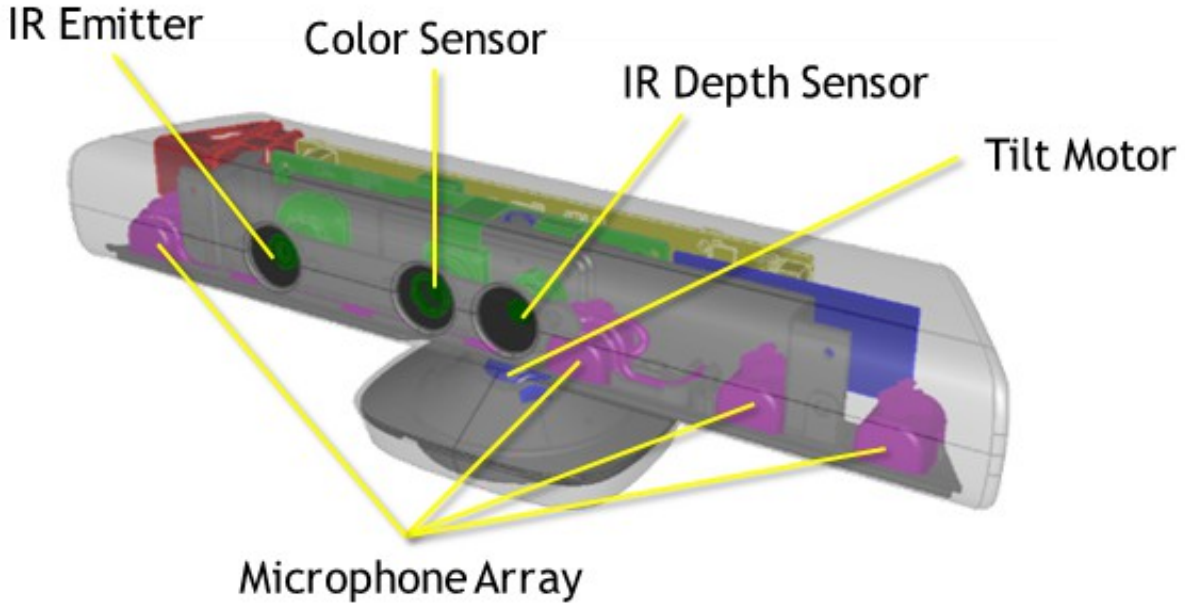
Historie

- 1. Juni 2009: Vorstellung des „Project Natal“ auf der E3
- 10. November 2010: Veröffentlichung von Kinect für Xbox 360
- 9. Januar 2012: Fertigstellung des Windows Kinect SDK 1.0
- Aktuell: Version 1.6 mit Support für Windows 8

SDK Aufbau


Applications
3
NUI API
Windows Core Audio and Speech APIs
DMO codec for mic array
5
4
2
Device setup
Device access
Video stream control
Audio stream control
User Mode
WinUSB device stack
WinUSB camera stack
USBAudio audio stack
Kernel Mode
Kernel-mode drivers for Kinect for Windows
USB Hub
Hardware
1
Motor
Cameras
Audio mic array
Kinect sensor
Kinect for Windows SDK
Windows components
User-created components

Hardware

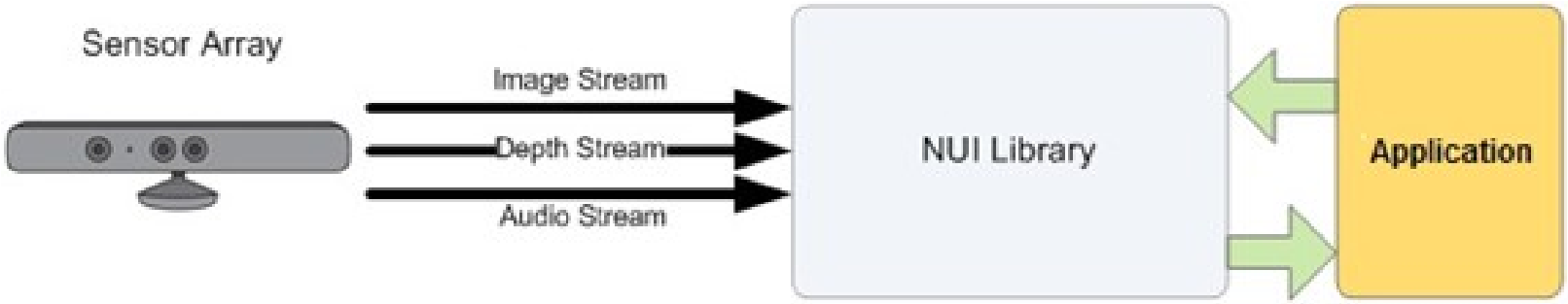


- Farbkamera (bis zu 1280 x 960 pixel x 32bit)
- Tiefensensor (640 × 480 pixel x 11bit)
  bestehend aus separaten Infrarot Sender und Empfänger
- 3D-Mikrofon
  4 einzelne Mikrofone mit je 24bit@16kHz
- Neigungssensor und -motor
  erhöht das Field of View in vertikale Richtung

NUI API

- NUI = Natural User Interface
- Datenströme
  - Image
  - Depth
  - Audio
  - Beispiel Implementierung
- Skeletal Tracking
  - Gelenke & Knochen
  - Beispiel Implementierung
- Spracherkennung
  - Speech API
  - Beispiel Implementierung

Datenströme



- Image: kontinuierlicher Strom von definierten Frames (zB RgbResolution1280x960Fps12)
- Depth: Tiefendaten(x,y), Werte in Millimeter zum Objekt
  Bitmap(x,y), Werte 1-6 entsprechend der Personenzugehörigkeit
- Audio: kontinuierlicher Audiostrom, mit der Möglichkeit nur bestimmte Richtungen und Echo herauszufiltern

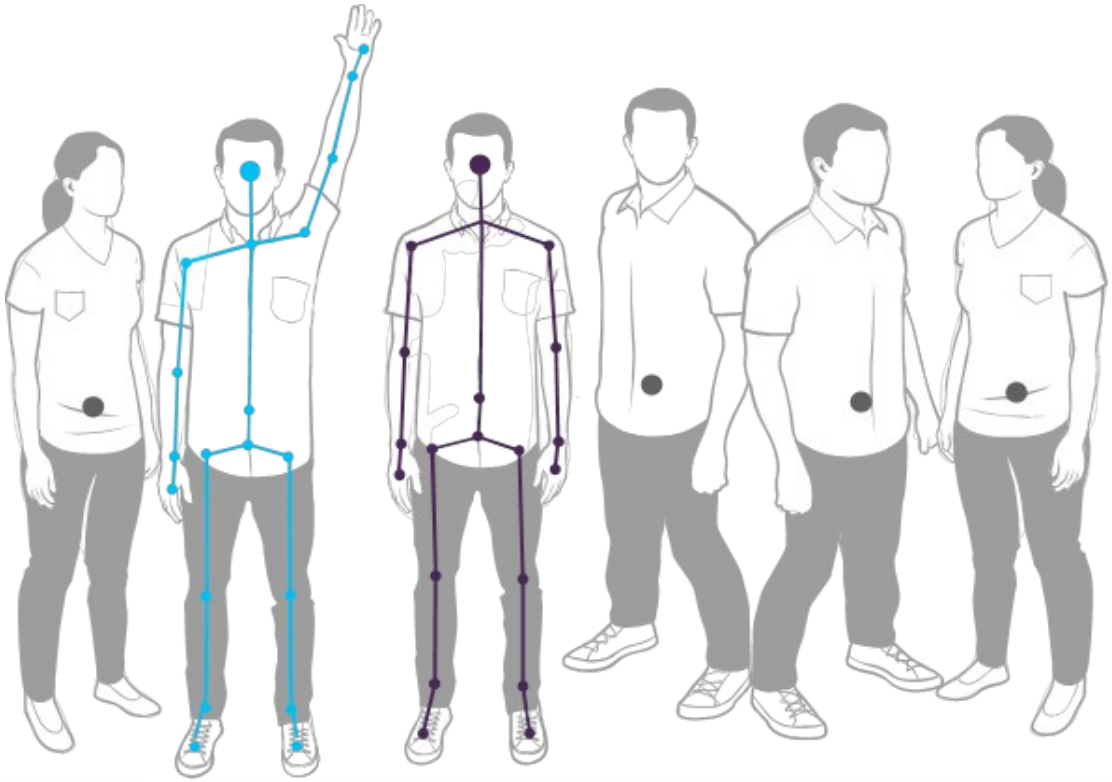
Beispiel Implementierung des Color Streams

```
//temporäres Array zum speichern der Bilddaten
byte[] rawColorPixels;
this.colorPixels = new byte[this.sensor.ColorStream.FramePixelDataLength];

//Eventhandler, löst aus wenn ein Color Frame verfügbar ist
private void SensorColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)  {
    using (ColorImageFrame colorFrame = e.OpenColorImageFrame())    {
        if (colorFrame != null)  {
        //Kopiere die Bilddaten in das temporäre Array
            colorFrame.CopyPixelDataTo(this.rawColorPixels);
        }
    }
 }
```

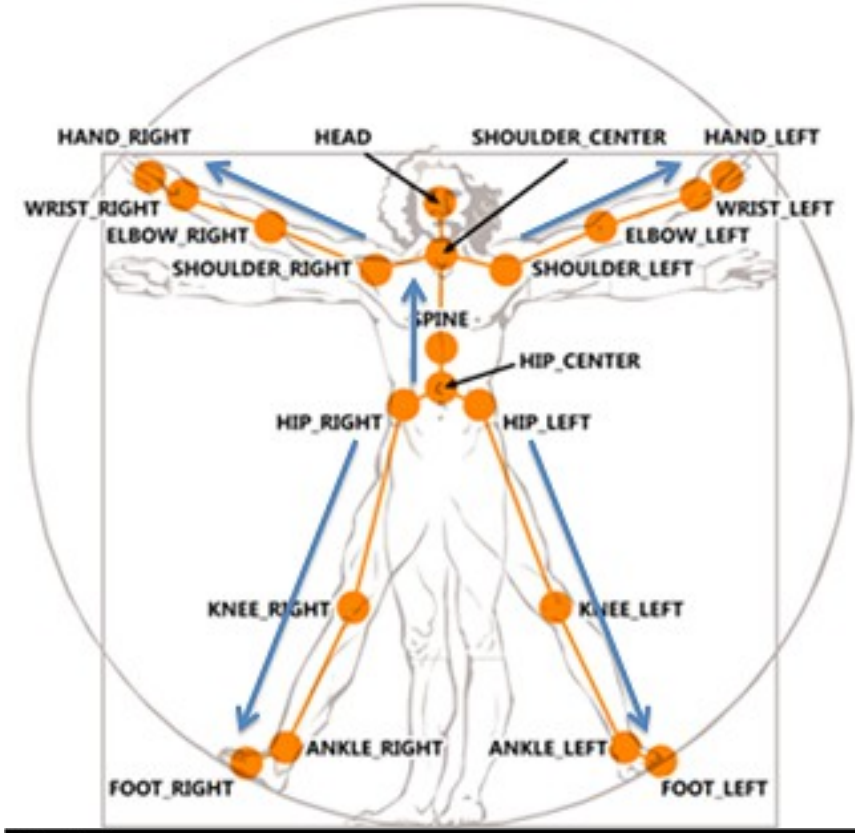
Skeletal Tracking

- Erkennung von bis zu 6 Personen
- 2 davon können detailliert erfasst werden



Gelenke & Knochen

- 2 unterscheidbare Personen mit bis zu 20 Gelenke
- Zustände „tracked“, „seated“ oder „position only“
- „tracked“ und „seated“ geben neben der Position auch die Orientierung des Gelenke zurück
- Hierarchisch verwaltet
- Knochen sind als Verbindung zweier Gelenke im Hierarchiebaum definiert



Beispiel Implementierung

```
//Eventhandler, löst aus wenn ein Skeleton Frame verfübar ist
private void kinect_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e) {
    using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame()) {
        if (skeletonFrame != null && this.skeletonData != null) {
            //rufe die skeletonData Informationen ab
            skeletonFrame.CopySkeletonDataTo(this.skeletonData);
            if (kinect.SkeletonStream.TrackingMode == SkeletonTrackingMode.Seated) {
                //zeichne die Knochen und Gelenke eine sitzenden Skelettes
                DrawSeatedSkeletons();
            }
            else {
                //zeichne die Knochen und Gelenke eine stehenden Skelettes
                DrawStandingSkeletons();
            }
        }
    }
}
```

Spracherkennung

- Microsoft Speech API (SAPI)
- Beam Technology
- Verfügbare Sprachen: de-DE, en-AU, en-CA, en-GB, en-IE, en-NZ, es-ES
  es-MX, fr-CA, fr-FR, it-IT, ja-JP
- Setzt eine Grammatik voraus

Beispiel Implementierung

```
//Erstelle eine Grammatik aus einer XML Datei
using (var memoryStream = new MemoryStream(Encoding.ASCII.GetBytes(Properties.Resources.SpeechGrammar))) {
    var g = new Grammar(memoryStream);
    speechEngine.LoadGrammar(g);
}

//Initialisierung
speechEngine.SpeechRecognized += SpeechRecognized;
speechEngine.SetInputToAudioStream(sensor.AudioSource.Start(), new SpeechAudioFormatInfo(EncodingFormat.Pcm,
16000, 16, 1, 32000, 2, null));
speechEngine.RecognizeAsync(RecognizeMode.Multiple);

private void SpeechRecognized(object sender, SpeechRecognizedEventArgs e) {
    //Schwellenwert, der die Spracherkennung abbricht, wenn er unterschritten wird
    const double ConfidenceThreshold = 0.3;

    if (e.Result.Confidence >= ConfidenceThreshold) {
        switch (e.Result.Semantics.Value.ToString()) {
            //Führe goStepForward() aus, wenn das Wort Vorwärts erkannt wird
            case "VORWÄRTS":
                goStepForward();
            ...
```

Anwendungsgebiete

- alternativer Spielekontroller der Xbox 360 für
  - Fitness- &
  - Tanzspiele
- Eingabegerät für „3D-Interfaces“
  - Kinect for Mediacenter
- günstiger Multifunktionssensor für Forschungsprojekte
  - Abstandssensor
  - „Vision“-sensor

Fazit

Vorteile

- neuartige Eingabemethode
- umfangreiches SDK
- günstiger Mehrzwecksensor

Nachteile

- (noch) keine Metro Unterstützung
- hohe Rechenleistung nötig
- begrenzte Performanz

Fragen?

Danke für eure Aufmerksamkeit