# Entwurf

## Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 08.03.2012 | 1.0 | Erste Version des Dokuments | LE |
| 09.03.2012 | 1.1 | Korrekturen und Review Frameworks | DT |
| 02.04.2012 | 1.2 | Interaktion durch Handtracking hinzugefügt | LE |

## Design Entscheide

### Frameworks

Um eine Applikation mit Microsoft Kinect zu entwickeln, stehen die folgenden drei Frameworks zur Verfügung:

* Kinect for Windows SDK
* OpenNI
* OpenKinect

Nachfolgend einige Anmerkungen zu diesen Frameworks.

#### Framework 1: Kinect for Windows SDK[[1]](#footnote-1)

Das offizielle Kinect Framework von Microsoft für Windows wurde kurz vor Beginn dieser Arbeit, im Februar 2012, in der Version 1.0 herausgegeben. Wenn man beachtet, dass andere Frameworks schon eher, als Beispiel OpenNI Ende 2010, veröffentlicht wurden, ist dies relativ spät. Entsprechend sind für dieses Framework viel weniger Beispiele und Bibliotheken im Internet zu finden, wobei dafür die Qualität deren hoch ist.

Dieses Framework geht durch die Nutzwertanalyse (siehe I.2.3 Nutzwertanalyse) klar als Sieger hervor.

#### Framework 2: OpenNI[[2]](#footnote-2)

Dieses Framework wurde in der Version 1.0.0.23 im Dezember 2010 erstmals freigegeben und konzentriert sich, im Gegensatz zum Microsoft Kinect SDK, nicht nur auf Kinect als Eingabemöglichkeit, sondern allgemein auf Natural User Interfaces (NUI).

Um weitere Geräte anzusprechen und gerätespezifische Funktionen zu implementieren, lässt sich im Framework zusätzliche Middleware einsetzen. So wird mit NiTE[[3]](#footnote-3) von PrimeSense[[4]](#footnote-4) entwickelt, um das Skeleton Tracking durchzuführen.

#### Framework 3: OpenKinect[[5]](#footnote-5) / libfreenect[[6]](#footnote-6)

OpenKinect ist eine Community, die den libreenect Treiber entwickelt. Leider gibt es dafür aber keine erweiterten Funktionen wie Gestenerkennung oder Skeleton Tracking.

#### Nutzwertanalyse

Um herauszufinden, welches dieser drei Framework das passende für die Entwicklung der Video Wall Applikation ist, wurde eine Nutzwertanalyse durchgeführt.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nutzwertanalyse: Auswahl Kinect Framework | | | | | | |  |
|  |  | **Framework 1** | | **Framework 2** | | **Framework 3** | |
|  |  | **Kinect for Windows SDK** | | **OpenNI** | | **OpenKinect / libfreenect** | |
| **Kriterium** | **Gewichtung** | **Bewertung** | **Total** | **Bewertung** | **Total** | **Bewertung** | **Total** |
| **Cooperate Support, Weiterentwicklung, Community** | 5 | 5 | 25 | 3 | 15 | 3 | 15 |
| **Windows Integration und Installation** | 3 | 5 | 15 | 3 | 9 | 3 | 9 |
| **Linux / Mac Kompatibilität** | 1 | 1 | 1 | 5 | 5 | 5 | 5 |
| **C# / .NET Framework / Visual Studio Integration** | 5 | 5 | 25 | 3 | 15 | 1 | 5 |
| **Skeleton Tracking Qualität** | 5 | 5 | 25 | 3 | 15 | 1 | 5 |
| **Libraries für Gestenerkennung** | 3 | 3 | 9 | 5 | 15 | 1 | 3 |
| **Record / Replay Funktionalität** | 5 | 3 | 15 | 5 | 25 | 1 | 5 |
| **Dokumentation** | 5 | 5 | 25 | 3 | 15 | 1 | 5 |
| **Mit Framework realisierte Beispiele und Libraries (Quantität)** | 3 | 1 | 3 | 5 | 15 | 5 | 15 |
| **Mit Framework realisierte Beispiele und Libraries (Qualität)** | 3 | 5 | 15 | 1 | 3 | 1 | 3 |
| **Total Punkte** |  |  | **158** |  | **132** |  | **70** |
| **Rang** |  |  | **1** |  | **2** |  | **3** |
|  |  |  |  |  |  |  |  |
| Bemerkung: Die Gewichtungs- / Bewertungsskala geht von 1 (am schlechtesten) bis 5 (am besten). | | | | | | |  |

Tabelle - Nutzwertanalyse: Auswahl Kinect Framework

Aus der Analyse (siehe Tabelle 1 - Nutzwertanalyse: Auswahl Kinect Framework) geht das Framework 1: Microsoft Kinect SDK als Sieger vor dem Framework 2: OpenNI hervor.

#### Sensitivitätsanalyse

In der Sensitivitätsanalyse wird untersucht, wie stark sich eine Änderung auf das Gesamtergebnis auswirken würde.

Das Framework 3 wird auch bei Änderungen der Bewertung nicht als Sieger hervorgehen.

Zwischen dem Framework 1 und 2 ist der Bewertungsunterschied einiges kleiner. Da das Framework 1 von Microsoft aber über ein ausgeklügeltes, vorhersehendes Skeleton Tracking System[[7]](#footnote-7) verfügt, eine bessere Dokumentation besitzt und perfekte Windows, Visual Studio, C# und .NET Integration bietet, würde sich dieses Framework trotz Anpassungen an einzelnen Gewichtungen oder Bewertungen gegenüber dem Framework 2 durchsetzen. Demensprechend ist diese Nutzwertanalyse nicht sensitiv gegenüber Änderungen.

#### Weiteres

Bei der Nutzwertanalyse wurden zwar möglichst viele nummerisch bewertbare Kriterien untersucht, es fehlt aber noch der persönliche Eindruck. Für das Projekt Video Wall fällt die Entscheidung gefühlsmässig auf das Microsoft Framework, da auch die übrigen für die Arbeit eingesetzten Technologien von Microsoft sind und damit gerechnet werden muss, dass andere Frameworks nicht ohne Probleme mit den für das Projekt bereits festgesetzten Microsoft Technologien zusammenarbeiten können.

Weiter ist in der Bachelorarbeit „Kinect Bodyscanner“[[8]](#footnote-8) von Felix Egli und Michael Schnyder im Kapitel 3.3.1 Kinect Framework auf Seite 30 beschrieben, dass OpenNI für die Arbeit nur eine temporäre Lösung ist. Es ist geplant, auf das offizielle Framework von Microsoft zu wechseln, sobald dieses verfügbar sei. Auch diese Aussage spricht klar für das „Kinect for Windows SDK“.

## Interaktion durch Handtracking

In der Domain Analyse wurde evaluiert, dass der Benutzer mithilfe der Hand die Applikation bedienen kann („Meine Hand ist die Maus“). Wie das genau funktioniert, wird in diesem Kapitel erläutert.

### Kinect Daten

Eine der wichtigsten Features der Kinect SDK ist das sogenannte Skeleton Tracking. Das funktioniert so, dass mithilfe der Sensoren (Tiefensensor, Bildsensor, Infrarotsensor) versucht wird, ein menschliches Skelett zu erkennen, und zwar in Echtzeit. Es ist möglich, gleichzeitig von bis zu zwei Personen das Skelett zu verfolgen. Für das Handtracking ist aber nur ein Skelett notwendig, weshalb in nachfolgenden Darstellungen nur ein Skelett dargestellt wird.



Abbildung - Beispiel eines Skeletts

### Handtracking

Wie oben dargestellt ist es möglich, die einzelnen Punkte in einem Skelett zu erkennen. Somit kann auch die rechte Hand verfolgt und dargestellt werden. Damit sich der Benutzer der Applikation nicht zu stark bewegen muss, wird eine Grenze für das Handtracking festgelegt. Das sieht schematisch folgendermassen aus:



Abbildung - Skelett mit Zone (rot) für das Handtracking

Die Position der Hand muss auf den Bildschirm projetziert werden, wobei die rote Zone die Position der Hand auf dem Bildschirm darstellt. Also wenn der Benutzer die Hand oben rechts des roten Bereiches bewegt, so wird diese oben rechts auf dem Bildschirm angezeigt. Befindet sich die Hand unten links des roten Bereiches so wird die Hand unten links auf dem Bildschirm angezeigt. Befindet sich die Hand nicht im roten Bereich, so wird sie (analog zur Maus) am Rand des Bildschirms oder gar nicht auf dem Bildschirm angezeigt. Der Monitor würde mit dem Skelett aus Abbildung 2 - Skelett mit Zone (rot) für das Handtracking folgendermassen aussehen:



Abbildung - Beispiel Monitor mit Handtracking

Wie die genauen Masse der Zone für das Handtracking sind und wo sich die Zone genau befindet ist in der Entwicklungsphase noch genauer zu definieren und kann direkt dem Quellcode entnommen werden. Grundsätzlich ist klar, dass sich die Zone über der Hüfte des Skelettes befinden wird und etwa bis in die Mitte des Körpers gehen wird. Ebenfalls wird die Zone nicht weit über dem Kopf hinausragen.

1. <http://www.microsoft.com/en-us/kinectforwindows/> [↑](#footnote-ref-1)
2. <http://openni.org/> [↑](#footnote-ref-2)
3. <http://www.primesense.com/Nite/> [↑](#footnote-ref-3)
4. <http://www.primesense.com/> [↑](#footnote-ref-4)
5. <http://openkinect.org/wiki/Main_Page> [↑](#footnote-ref-5)
6. <https://github.com/OpenKinect/libfreenect> [↑](#footnote-ref-6)
7. <http://www.cs.dartmouth.edu/~cs104/BodyPartRecognition.pdf> [↑](#footnote-ref-7)
8. <http://eprints3.hsr.ch/180/1/Hauptdokument.pdf> [↑](#footnote-ref-8)