Karlsruher Institut für Technologie

Intelligente Sensor-Aktor-Systeme

Kinect Point Cloud to Unreal Engine 4

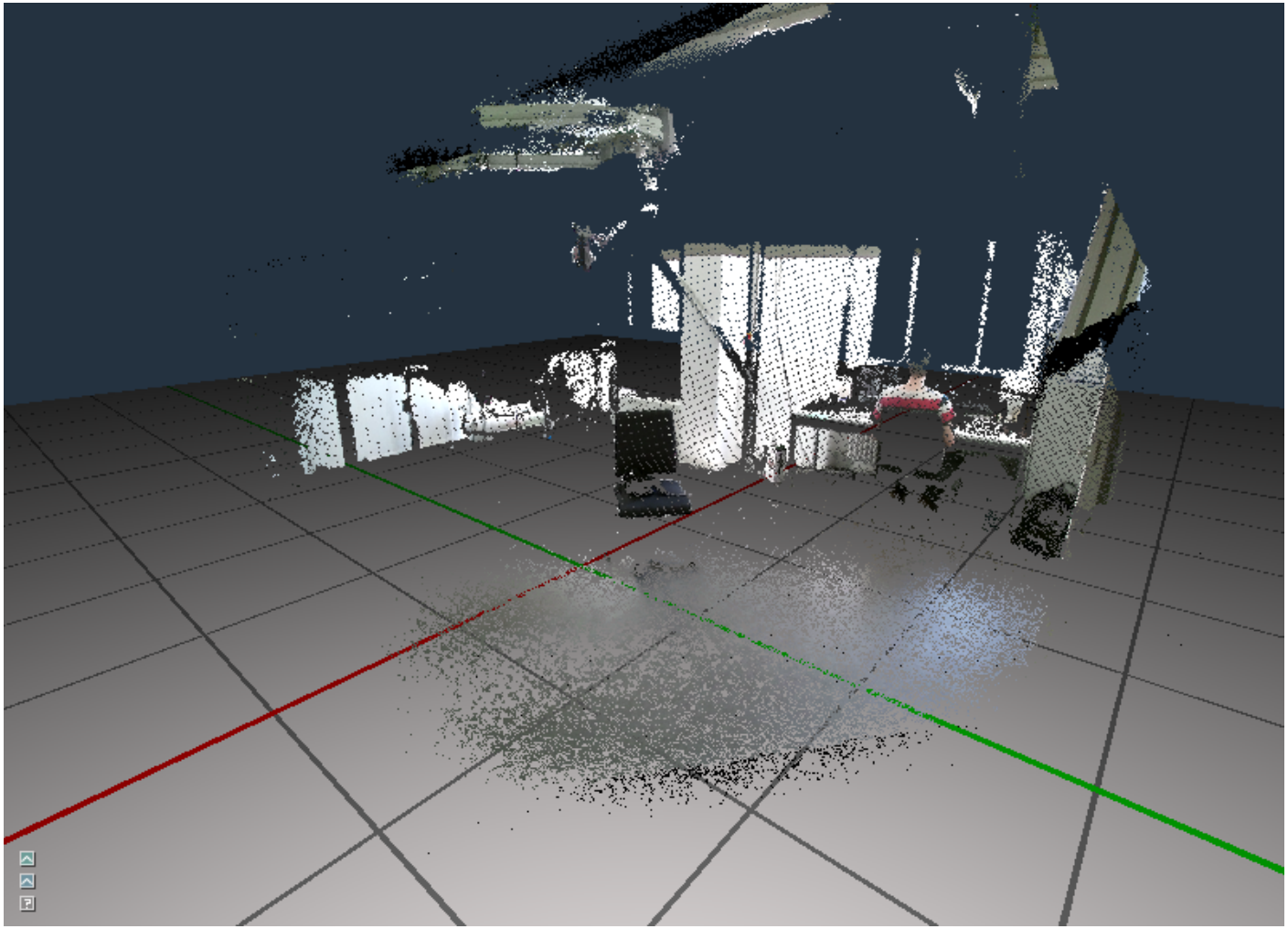
Marius Wodtke | 22. Juli 2016

SEMINAR: ANTHROPOMATIK PRAKTISCH ERFAHREN

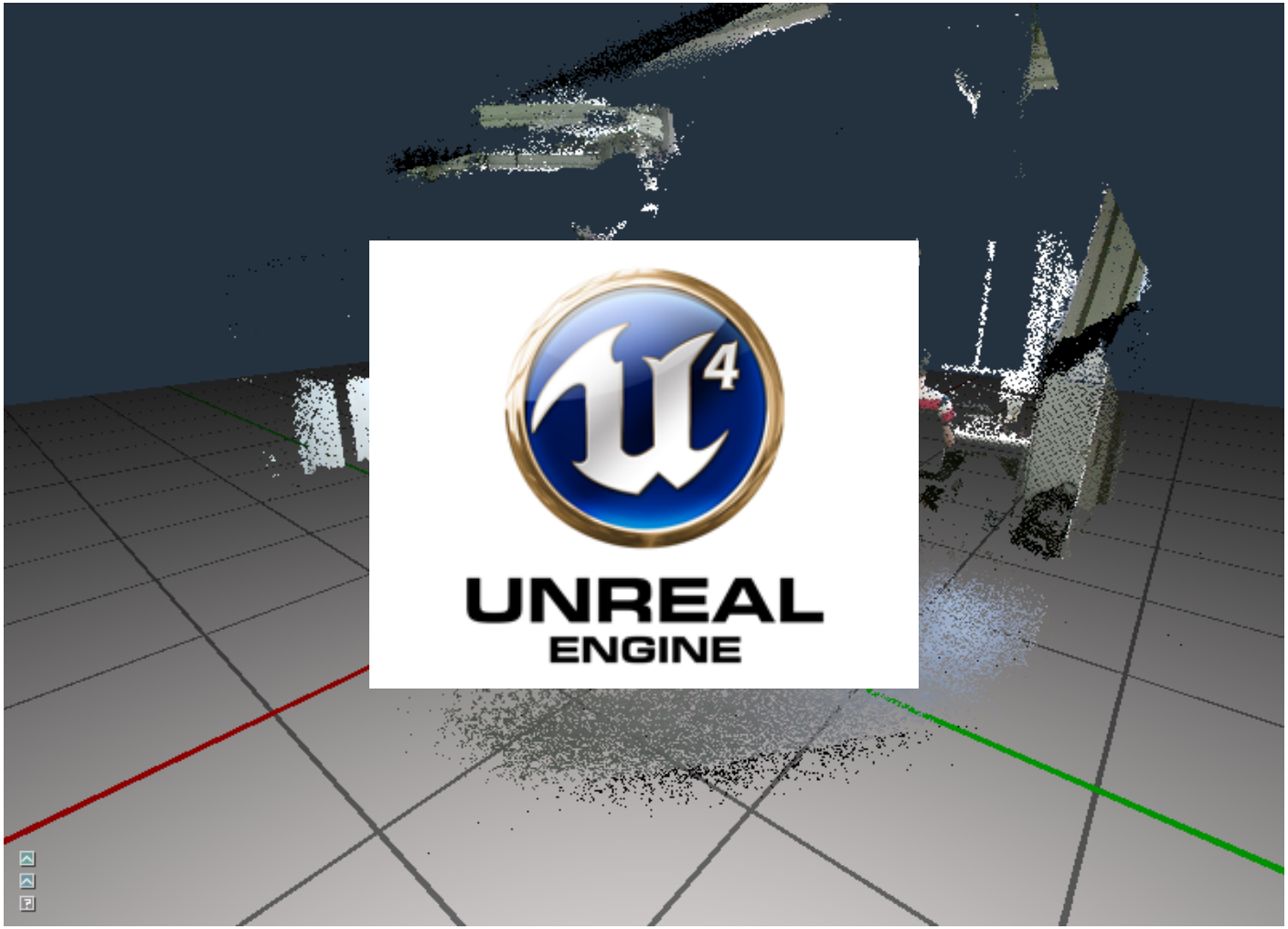
Gliederung

- Problemstellung
- Kinect
- Kinect 4 Windows Plugin
- PCViewer
- Berechnen der Wolke
- Demo
- Fazit

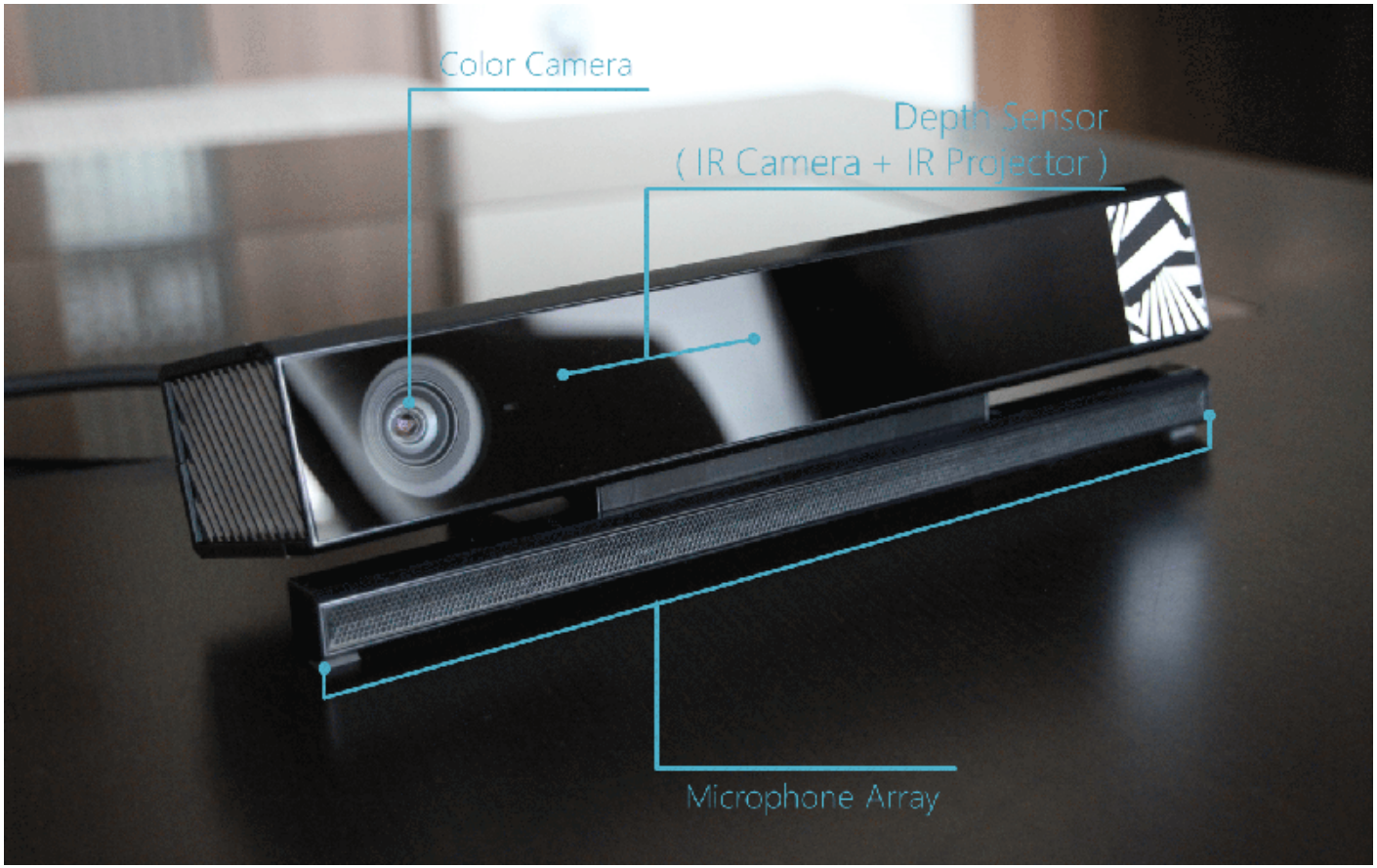
Problemstellung



Problemstellung



Microsoft Kinect v2.0



Kinect 4 Windows v2.0 Plugin

- Erlaubt es eine Kinect auszulesen und zu importieren
- Liest die Kinect über den Treiber aus $\Rightarrow$ Kinect RE und SDK erforderlich
- Importiert die Bilder als Texturen
- Achtung beim kompilieren
- Video

PCViewer

- Client liest Kinect aus und versendet diese auf Anforderung an Server
- Server sammelt Daten ein
- Es gibt 4 Kinects in jeder Ecke des Holodecks ...
- ... und auch für jede einen Rechner
- Der PCViewer kann bereits Point Clouds anzeigen
- Video
- Das 10.000 Cities Projekt benutzt ihn bereits

PCViewer

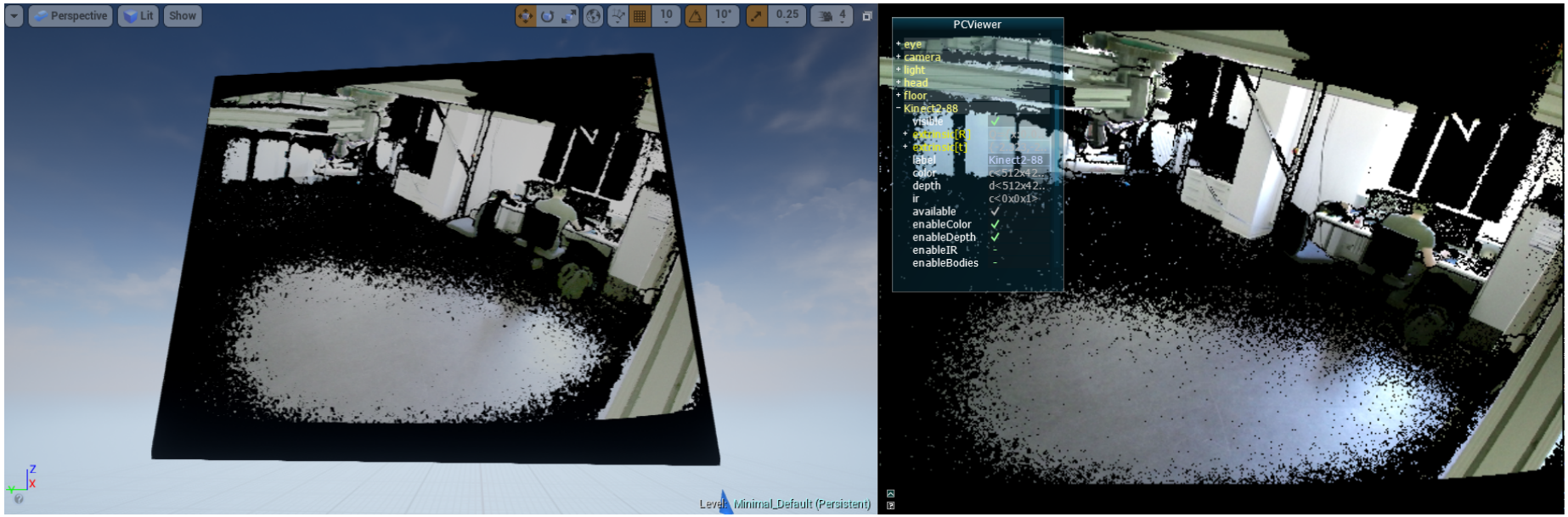
- Client liest Kinect aus und versendet diese auf Anforderung an Server
- Server sammelt Daten ein
- Es gibt 4 Kinects in jeder Ecke des Holodecks ...
- ... und auch für jede einen Rechner
- Der PCViewer kann bereits Point Clouds anzeigen
- Video
- Das 10.000 Cities Projekt benutzt ihn bereits

PCViewer

- Client liest Kinect aus und versendet diese auf Anforderung an Server
- Server sammelt Daten ein
- Es gibt 4 Kinects in jeder Ecke des Holodecks ...
- ... und auch für jede einen Rechner
- Der PCViewer kann bereits Point Clouds anzeigen
- Video
- Das 10.000 Cities Projekt benutzt ihn bereits

Auslesen der Kinect Streams

- Verbindung zur Kinect prüfen
- Auslesefunktion des PCViewer aufrufen
- Unreal Engine arbeitet mit TArrays, PCViewer mit Puffern
- Z.B. Farben in drei uint8 Werten kodiert, Unreal möchte eine FColor



Berechnen der 3D Position

Bildpunkt: $\begin{bmatrix} u \\ v \\ \gamma \end{bmatrix} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$

3D (Ursprung): $\begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \end{bmatrix} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}^{-1} \times \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \times \gamma$

Berechnen der 3D Position

Bildpunkt: $$\begin{bmatrix} u \\ v \\ \gamma \end{bmatrix} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

3D (Ursprung): $$\begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \end{bmatrix} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}^{-1} \times \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \times \gamma$$

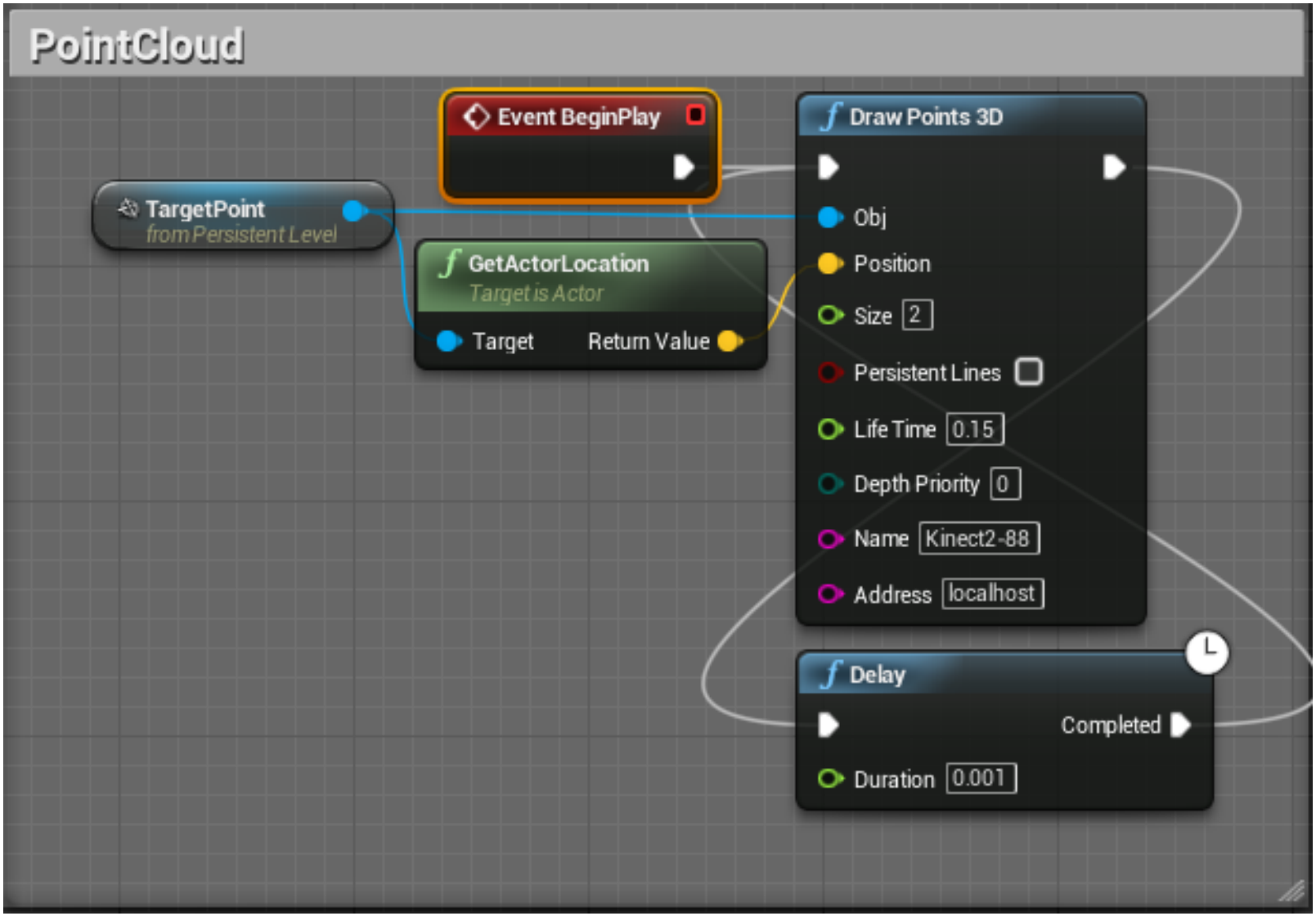
Berechnen der 3D Position

Bildpunkt: $\begin{bmatrix} u \\ v \\ \gamma \end{bmatrix} = I \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$

3D (Ursprung): $\begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \end{bmatrix} = I^{-1} \times \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \times \gamma$

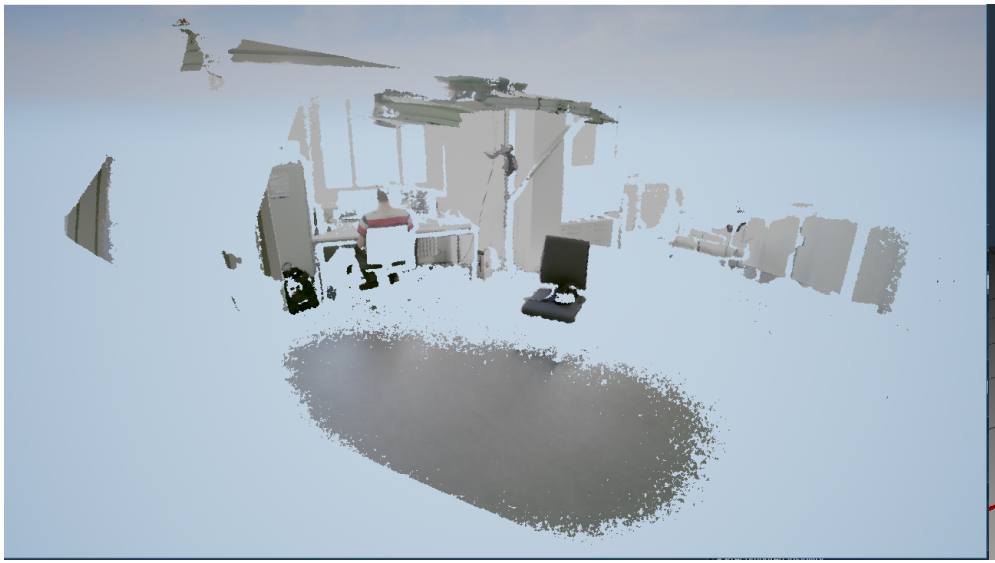
3D Punkt: $\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = E \times \begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \end{bmatrix}$

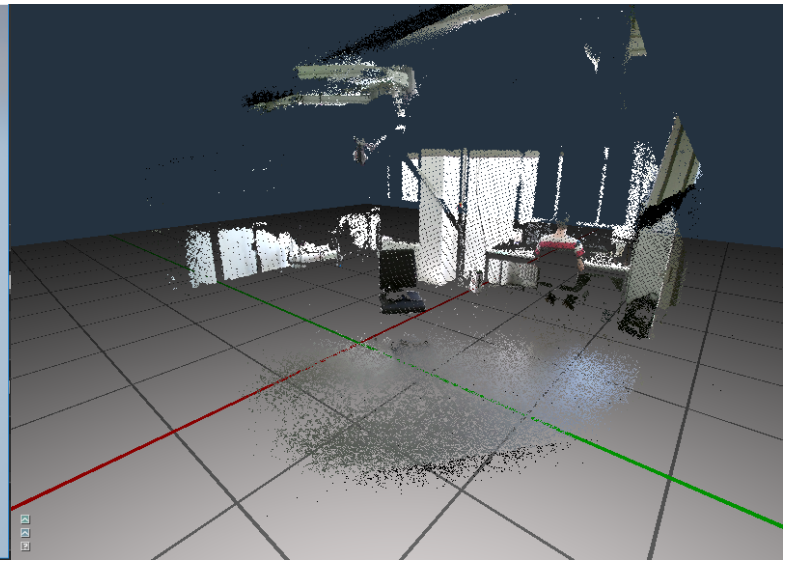
Beispiel



Ergebnis

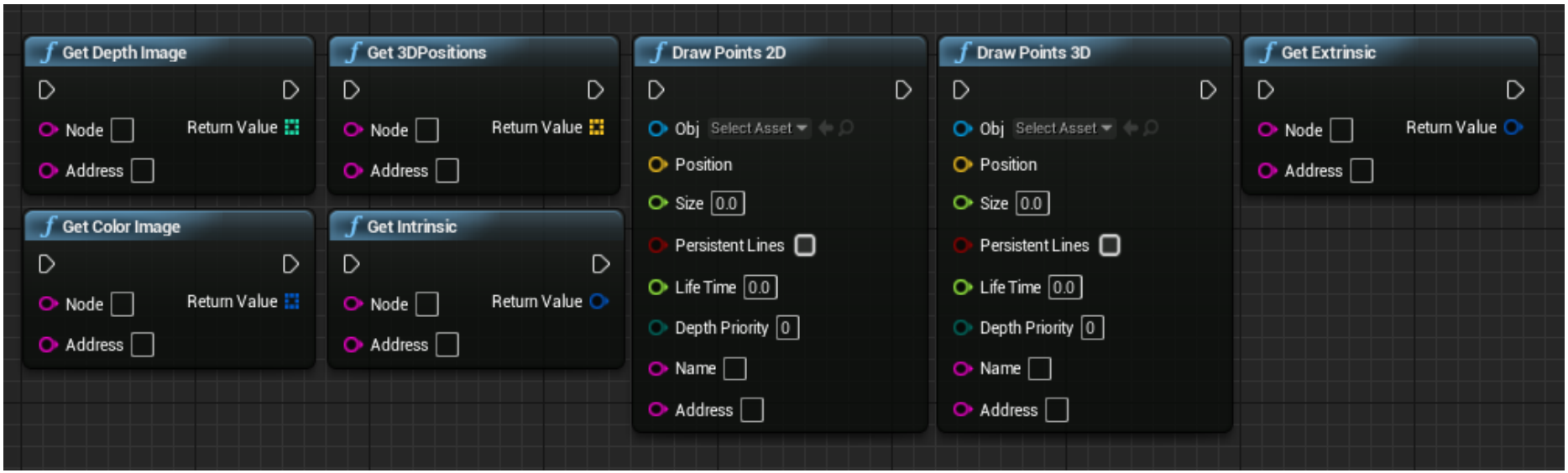
- Nun an jede 3D Position einen farbigen Pixel zeichnen
- Dabei schwarze Pixel und Pixel ohne Tiefe auslassen
- Video





Implementierte Methoden und Fazit

- Nun an jede 3D Position einen farbigen Pixel zeichnen
- Dabei schwarze Pixel und Pixel ohne Tiefe auslassen
- Video



- Überraschend viele Umwandlungen notwendig
- Performancekritische Aufgabe

Danke für Ihre Aufmerksamkeit