Kinect Punktwolken in die Unreal Engine importieren

Marius Wodtke

***Zusammenfassung*—Die Ausarbeitung beginnt mit einer kurzen Zusammenfassung.**

I. Einleitung

Technische Entwicklungen wie *head-mounted displays* haben die Erfahrung der Telepräsenz in den letzten Jahren deutlich verbessert. Aber um sich an einem entfernten beziehungsweise virtuellen Ort wirklich anwesend zu fühlen reicht es nicht die Bewegung des Nutzers korrekt zu übersetzen. Zusätzlich muss die Welt und die Personen in ihr passend dargestellt werden. In einer ausgedachten Umgebung mag dies nur eine Frage der möglichen Grafikqualität sein, will man jedoch einen realen Ort mit echten Menschen abbilden fallen dem Nutzer viel schneller Ungenauigkeiten und Unstimmigkeiten auf. Optimal wäre es die Welt und ihre Personen nicht designen zu müssen, sondern sie originalgetreu zu digitalisieren. Ein wichtiges Werkzeug dazu ist die Kinect Kamera von Microsoft, diese bietet Sensoren um die Welt und ihre Menschen zu scannen und so zu digitalisieren, nun müssen die Daten nur noch sinnvoll interpretiert und nutzbar gemacht werden.

II. Aufgabenstellung

Aufgabe soll es sein, die von der Kinect ausgegebenen Streams in die frei verfügbare Unreal Engine zu importieren und sie so für weitere Aufgaben nutzbar zu machen. Zusätzlich soll eine dieser Aufgaben Fokus dieser Arbeit sein - die Zusammenführung von Farb- und Tiefeninformation zu einer Punktwolke, die die reale Welt in die virtuelle Welt abbilden soll.

III. Microsoft Kinect

Die Kinect von Microsoft bietet einen Tiefensensor, eine RGB-Kamera, einen Infrarotsensor und mehrere Mikrofone. Kombiniert man Tiefensensor und RGB-Kamera kann man eine Punktwolke erzeugen, dazu müssen die Pixel des Kamerabildes entlang der Tiefeninformation in den 3-dimensionalen Raum projeziert werden.
Die dazugehörige Software ermöglicht noch viele andere Interpretationsarten der gelieferten Bilder, zum Beispiel kann sie ein Knochenmodell von Personen vor der Kamera berechnen, diese können dann in Avatare umgesetzt werden, die die Bewegungen der Personen kopieren.

IV. Kinect 4 Windows v2.0 Plugin

Das Kinect 4 Windows v2.0 Plugin ermöglicht es die verschiedenen Input Streams der Kinect in die Unreal Engine zu importieren. Das Plugin setzt dabei bereits eine funktionierende Kinect voraus, es müssen zunächst die *Kinect Runtime* (https://www.microsoft.com/en-us/download/details.aspx?id=44559) und die *Kinect SDK* (https://www.microsoft.com/en-us/download/details.aspx?id=44561) installiert werden.
Anschließend sollte die Kinect getestet werden, dazu kann der *PCViewer* des ISAS verwendet werden. Zunächst wird der Server (*PCViewer*) und anschließend der Client (*Holodeck*) gestartet, nun sollte die Kinect im Menu des Servers erscheinen und es können die *Color-* und *Depth-Streams* aktiviert werden. Sieht man nun eine Punktwolke, die dem Blickwinkel der Kamera entspricht, funktioniert alles wie vorgesehen.

V. Projektmappe erstellen

Zunächst klont man das GitHub Projekt KinectDemoRoom (https://github.com/lion03/KinectDemoRoom). Nach einem Rechtsklick auf die .uprojekt Datei muss zunächst die Version 4.9 der Unreal Engine ausgewählt werden, danach können im selben Menu die Projektdateien erstellt und eigentlich auch kompilliert werden, letzteres schlägt aber fehl und es müssen einige Kompilerfehler von Hand behoben werden.
Zum einen muss die *UK2Node_BaseAsyncTask* in *xy* bekannt gemacht werden, es muss eine Zeile *include* im Dateikopf hinzugefügt werden. Zum anderen muss im *KinectV2Module* für die *PrivateDependencyModuleNames* und *PublicDependencyModuleNames* der Eintrag *"UnrealEd"* hinzugefügt werden, weiteres zum dem Problem ist unter https://github.com/lion03/KinectDemoRoom/issues/1 zu finden.

VI. Die Demo Umgebung

Nachdem die Projektmappe erfolgreich erstellt wurde sollte sich die .uproject Datei öffnen lassen und der Unreal Editor öffnet sich. Die Demo Umgebung besteht aus mehreren kleinen Räumen, die die Grundfunktionalität des Plugins präsentieren. Dazu gehört das Auslesen der Input Streams wie Farbe, Tiefe und Infrarot, außerdem trennt der *Body ID* Stream Personen vom Hintergrund und das *Bone Model* zeigt eine visuelle Repräsentation des berechneten Knochenmodells. Eine Implementierung für die Darstellung einer Punktwolke ist noch nicht vorhanden.

VII. Blueprint

Zu Spielbeginn muss zunächst die Kinect gestartet werden, wichtig auch sie bei Spielende wieder abzuschalten, sonst läuft sie die ganze Zeit. Wenn sie gestartet wurde können *Event Dispatcher* angelegt werden, an die passende Events gebunden werden. Diese liefern bei jedem ankommenden Bild der Kinect ein Ereignis und das Bild als Textur zurück, diese lässt sich gut auf Flächen projezieren, die Projektion auf einzelne Pixel gestaltet sich schwieriger.
Um das Farbbild hierfür nutzbar zu machen muss die Textur in ein Array mit *LinearColor* Werten umgewandelt werden, die Werte des Arrays können dann einzeln ausgelesen und als Pixelfarben gesetzt werden. Eine Funktion zur Umwandlung stellt der Unreal Editor nicht zur Verfügung, sie muss als C++ Funktion hinzugefügt werden, dies wird aber in einem gesonderten Abschnitt behandelt. Ähnliches gilt für die Tiefeninformation, die ebenfalls als Textur kodiert werden.
Sind die Farb- und Tiefeninformationen eines Frames in Arrays verpackt kann man beginnen die Punktwolke zu erzeugen.

VIII. Erweiterungen des Plugins

IX. Projektion in den dreidimensionalen Raum

Während die Kameras der Kinect einen konischen Bereich aus der Welt aufzeichnen, bilden sie diesen natürlich auf ein zweidimensionales Bild ab. Dabei wird jeder dreidimensionale Punkt $[y_1, y_2, y_3]^T$ mit der intrinsischen Kamera-Matrix

$$K = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}$$

in ein einen zweidimensionalen Pixel und eine Tiefe $[u, v, \gamma]^T$ transformiert. Dabei bezeichnen f_u und f_v die Fokuslänge und c_u und c_v den Hauptpunkt. Ein Punkt des Tiefenbildes berechnet sich also aus

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

Für die Projektion in den dreidimensionalen Raum der virtuellen Realität muss mit $[u, v, \gamma]^T \times K^{-1} = [y_1, y_2, y_3]^T$ aus jedem Punkt des Tiefenbildes und dem Inversen der intrinsischen Kamera-Matrix der unsprüngliche Punkt im dreidimensionalen Raum berechnet werden, die Formel lautet also

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \frac{u - c_u}{f_u} \\ \frac{v - c_v}{f_v} \\ 1 \end{bmatrix} \times \gamma.$$

Hinzu kommt noch die extrinsische Kamera-Matrix, diese definiert die Position und Rotation der Kamera in der realen Welt. Sie wird durch eine Kamera simuliert, die in der virtuellen Welt auf die korrekte Position gesetzt und gedreht wird, nun kann jeder der zuvor berechnete Vektoren $[y_1, y_2, y_3]^T$ mit der Rotation multipliziert und auf die Postion der Kamera addiert werden, nun erhält man den korrespondierenden Vektor $[x, y, z]^T$ in der virtuellen Welt. Anschließend wird der Farbwert des jeweiligen Pixels $[u, v]^T$ aus dem Farbbild ausgelesen und es wird an der Stelle $[x, y, z]^T$ ein Pixel in der entsprechenden Farbe erzeugt.

X. Fazit