Dokumentation Visualisierung Sci-Vis

Benedikt S. Vogler

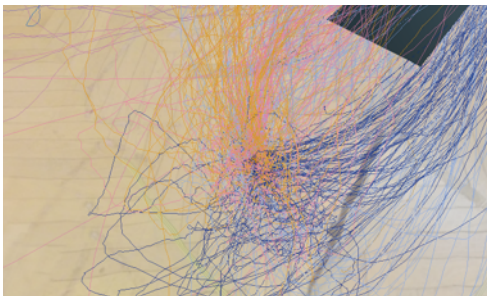
GitHub Repo: https://github.com/BSVogler/MovementPathVisualizer

Die Daten und meine Idee

In meinem Studienprojekt an der Professur für Virtual Reality haben wir eine Nutzerstudie durch geführt. Hierbei zeichneten wir die Bewegungspfade von 20 Studienteilnehmern und Teilnehmerinnen auf. Unser Ziel war es den Einfluss der Freiheitsgrade auf die Leistung zu untersuchen. Deswegen gab es unterschiedliche Aufgaben. Je nach Aufgabe waren die Freiheitsgrade eingeschränkt. So gibt es eine Aufgabe, wo nur rotiert wurde (T0 R3), nur auf der Ebene eine Translation durchgeführt wurde (T2 R0), eine mit Translation im Raum (T3 R0) und eine Aufgabe mit Translation und Rotation zusammen (T3 R3). Die Personen hielten einen Cursorobjekt in der Hand, dessen Position über optisches Tracking aufgezeichnet wurde.
Die Bewegungspfade sollen nun in meiner Visualisierung gezeigt werden, um so eventuell neue Auffälligkeiten in den Daten zu finden oder Probleme für weiterführende Arbeiten aufzuzeigen.

Art der Visualisierung

Bei den Daten handelt es sich um drei dimensionale Positionsdaten mit zeitlicher Ordnung. Einen Teil der Daten, wie z.B. Geschwindigkeitsveränderungen könnte man über Diagramme untersuchen, ich hielt die Rekonstruktion im Raum aber für sinnvoller, um so den Fokus auf die Bewegungspfade zu legen („justified 3D“). Um den Kontext zu kommunizieren, ist der Hintergrund eine Photosphere-Bildaufnahme, von dem Ort an dem die Studie durchgeführt wurde. Ziel der Visualisierung ist es Muster oder Auffälligkeiten in den zurückgelegten Pfaden zu erkennen. Die Visualisierung ist ähnlich zum „Streamlines“-Konzept, nur dass hier eine einzige komplexe Linie visualisiert wird.
Es gibt eine Vielzahl von Datensätzen (125 Datensätze). Diese alle gleichzeitig zu visualisieren macht hier wenig Sinn, da die „Hotspots“ bereits bekannt sind. Vielmehr können die Datensätze einzeln über Menüs ausgewählt und übereinander gelegt werden („superimpose“), in dem man sie nacheinander hinzufügt oder wieder entfernt. Unterschiede zwischen den Nutzern können so aufgezeigt werden.



Technologie

Meine Wahl für ein Web-Front-End ist nicht durch die Art der Daten begründet, sondern vielmehr aus persönlichen Interesse. Native Lösungen würden sich genau so gut eigenen. Da ich bis jetzt keine Erfahrungen mit hardwarebeschleunigter Grafik im Browser (also WebGL) gesammelt habe, wollte ich es hier erstmalig benutzen. Als Framework wählte ich A-FRAME, ein sehr junges Framework (Erstellt August 2015) des Mozilla VR Teams der Mozilla Corporation. A-FRAME ist eigentlich für die Konstruktion von Web-VR-Anwendungen konzipiert. Da es auf THREE.js, einem generischeren WebGL Framework, aufbaut ist eine gute Basis für meine Arbeit. Die VR-Unterstützung ist ein willkommenes Feature für die Visualisierung, was zur Betrachtung der visualisierten Daten genutzt werden kann.

Die Daten und ihre Bereinigung

Die rohen Daten enthielten eine Transformationsmatrix der aktuellen Cursorposition, die alle paar Millisekunden aufgezeichnet wurden mit dem zugehörigen Zeitstempel. Nach dieser Matrix folgte eine weitere Matrix für die aktuelle Zielposition. Da sich dies in den Aufgaben nicht geändert hat, ist dies redundant gespeichert. Durch verrauschte Werte in den Matrizen ist dies mit „Suchen“ und Ersetzen nicht einfach zu entfernen[1], so dass diese Zeilen beim Einlesen übersprungen werden.
Zusätzlich kürzte ich Wörter und löschte Zeichen, um durch die massive Häufigkeit der Wiederholung weiteren Speicher zu sparen. Bereinigt sind die Daten von 300 Mb auf 200 Mb in der Größe reduziert worden. Dies ist weiterhin zu viel, um alles auf ein mal in den Speicher zu laden. Das Laden und Einlesen der Daten erfolgt daher erst auf Anforderung.
Auszug aus den bereinigten Daten:

[1] Hier könnte mit regulären Ausdrücken gearbeitet werden.

```
t:0.3563210964202881
0.998 -0.033 -0.047 0.000 0.033 0.999 0.001 0.000 0.047 -0.002 0.999 0.000 0.000 0.000 0.000 1.000
Target Pos:
0.998 -0.033 -0.047 0.000 0.033 0.999 0.001 0.000 0.047 -0.002 0.999 0.000 0.000 0.000 0.000 1.000

t:0.3729681968688965
0.998 -0.033 -0.047 0.000 0.033 0.999 0.001 0.000 0.047 -0.002 0.999 0.000 0.000 0.000 0.000 1.000
Target Pos:
0.998 -0.033 -0.047 0.000 0.033 0.999 0.001 0.000 0.047 -0.002 0.999 0.000 0.000 0.000 0.000 1.000
```

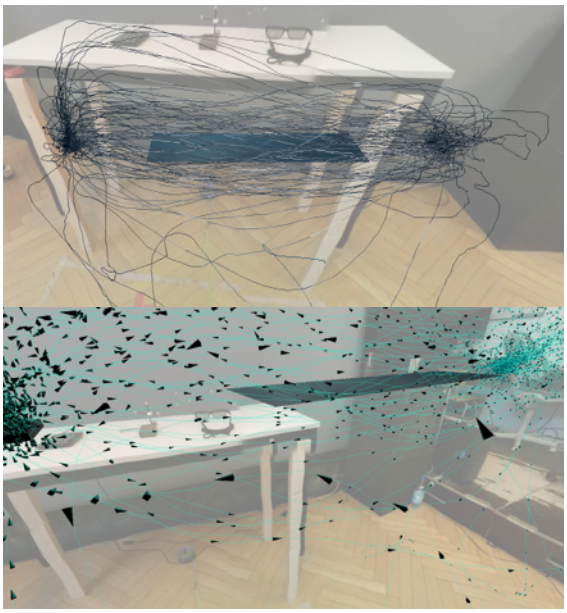
Die Präzision der Matrix ist leider nicht sehr hoch, da die Zellen auf drei Nachkommastellen gekürzt sind. Dies macht sich in einer Diskretisierung sichtbar. Insgesamt gibt es in diesem Format in unseren Daten 1.947.848 Matrizen. Rund die Hälfte davon sind Matrizen der Zielpositionen. Aufgrund meiner Untersuchung der log-Dateien wurde das Datenformal in der laufenden Studie angepasst und Transformations- und Rotationswerte direkt in die Datei geschrieben. Das neue aktuelle Datenformat wird ebenfalls unterstützt.

Auszug aus dem neuen Format:

```
taskSet, task, time, T_x, T_y, T_z, R_x, R_y, R_z, R_w
2,0,0.01300191879272461,0.21911801397800446,-0.2148089933395386,0.16358000040054332,0.0,0.0,0.0,1.0
2,0,0.17356491088867188,0.21911901235580444,-0.2148089933395386,0.1635829806327821,0.0,0.0,0.0,1.0
2,0,0.1895430088043213,0.21911901235580444,-0.21480792045593264,0.1635829806327821,0.0,0.0,0.0,1.0
2,0,0.20786595344543457,0.21911901235580444,-0.21480792045593264,0.1635839343070985,0.0,0.0,0.0,1.0
2,0,0.2254350185394287,0.21911801397800446,-0.21480792045593264,0.1635839343070985,0.0,0.0,0.0,1.0
```

Umsetzung

Für einen Pfad wird nur ein Objekt angelegt, was eine Linie erzeugt und alle enthaltenen Punkte enthält. Die Rotation fehlt in der Visualisierung mittels einer Linie natürlich. Zwei Komponenten der Rotation könnten über „Stream Ribbons“ visualisiert werden. Wenn man die Geschwindigkeitsveränderungen sichtbar machen will, kann man dies über die Farbe tun, was statt einem großen Objekt viele kleine Linienabschnitte benötigt. Die Schwächen des Frameworks zeigen sich, wenn viele Objekte dynamisch angelegt werden. Das Framework arbeitet über das DOM, so dass alle Objekte, die gerendert werden sollen, im DOM abgelegt werden müssen. Der Parser braucht hier länger und die Bildrate fällt stark ab. Die Geschwindigkeit beim Einlesen konnte ich verbessern, in dem ich nur noch ein DOM-Objekt, speziell optimiert für diesen Fall, benutze. Die Render-Geschwindigkeit könnte man durch die Benutzung von einem einzigen Mesh verbessern und die Farbdaten mit in dieses speichern und von einem angepassten Shader bestimmt werden.



Zur Verbesserung der allgemeinen Geschwindigkeit werden Duplikate in den Daten gesucht und entfernt. Der spätere timestamp der Duplikate wird dann benutzt.
Da die Linien eine Richtung angeben, muss diese visualisiert werden. Sonst ist unklar, in welcher Richtung die Bewegung statt gefunden hat. Dies habe ich durch Kegel umgesetzt, die parallel zur Linie rotiert sind und wie Pfeilspitzen die Richtung anzeigen.

Die Steuerung der Szene wird über ein anderes HTML-Dokument getätigt. Die Szene wird, da sie immer den gesamten HTML-Body bedeckt, über ein iFrame eingebunden, so dass trotzdem HTML-Bedienelemente benutzt werden können. Eine Umsetzung des GUIs über VR Elemente sprengt den Rahmen dieses Projektes.
Zusätzlich gibt es noch eine Echtzeit-Animation des Cursor entlang des Pfades.

Weiterführende Arbeiten

An der Stelle des Cursors könnten dann zusätzliche Daten angezeigt werden oder ausgewählte Punkte mit dem „Focus and Context“-Konzept herausgestellt werden. Eine Multiview-Overfläche mit 2D-Diagrammen würde sich hier anbieten. Der Cursor kann mittels Schaltflächen entlang des Pfades navigiert werden.