WebGL + Canvas + WebAudio

50 Years Swiss Music Charts

Too Much Heaven

Abgesehen von Logo und Lightboxes wurde das gesamte User Interface mittels WebGL umgesetzt.

Eine Herausforderung war die ansehnliche Anzahl an 3D Objekten. **4192 Songs mit 36'215 Platzierungen** sind in den abgebildeten 50 Jahren Hitparade enthalten.

Die Sterne entsprechen den Platzierungen und wurden als **Partikel-Cloud** umgesetzt, deren Rendering **eigens angefertigte Shader** übernehmen, die auf Performance optimiert sind. Auch die Linien sind sehr rudimentäre WebGL Objekte, die zwar in ihrer Anwendung etwas eingeschränkt sind, dafür eine gute Geschwindigkeit zeigen. Hier nehmen wir der 3D-Engine ausserdem Arbeit ab, indem wir unser Wissen über die angezeigten Objekte benutzen: wir kennen schliesslich den abgebildeten Zeitraum (also etwa das Jahr 1973) und können sehr performant entscheiden, welche Objekte sicher nicht sichtbar sind (etwa Songs aus dem Jahr 1984). So muss die Engine nicht erst die Kamera-Projektion ausführen (was immer mehrfache Matrizen-Multiplikationen pro Objekt bedeutet), um die Visibilität zu berechnen.

Das gleiche Prinzip wenden wir auch bei den mit Abstand aufwändigsten Objekten, den Texten, an. Texte gibt es bei WebGL nicht, es handelt sich hier um **transparente Texturen**, die dynamisch via Canvas-Operationen erzeugt werden. Tausende von Songs bedeuten auch tausende von Texturen und Abertausende von Zeichenoperationen. Mittels konventionellem Szenenaufbau geht das nicht, das würde fast jeden Computer in die Knie zwingen. Anstatt dessen kreieren wir die Texturen "on the fly" und verwerfen sie wieder, wenn das Objekt aus dem sichtbaren Bereich verschwindet. Anstatt die Canvas-Elemente, die es für das Erstellen der Textur benötigt, jedes Mal zu erzeugen und wieder zu löschen, greifen wir hier auf einen Pool zurück, der die Objekte recyclet. Dieser Pool gibt zur Performance-Optimierung wenn verfügbar ein Canvas zurück, das bereits die richtige Grösse hat (die Grössen sind relativ einheitlich, da jede Dimension einer 3D-Texture immer eine Potenz von 2 sein muss, also etwa 256x128.

Nur diese Massnahmen erlauben es, dass auch Smartphones fähig sind, die Website abzuspielen.

Hit Me Baby One More Time

Die grosse Menge von Objekten ist auch für eine weitere Herausforderung verantwortlich: um das Objekt unter der Maus zu finden, sind sehr komplexe Berechnungen nötig. Ein Strahl (Raycast) wird so in die Szenerie gelegt, dass er genau unter die Maus zu liegen kommt. Dieser Strahl wird nun mit sämtlichen Objekten der Szenerie verglichen: wenn es zu Überschneidungen kommt, sind das Kandidaten – eine Sortierung nach Entfernung liefert dann das gesuchte Objekt.

Auf einen solchen Raycaster konnte aber nicht zugegriffen werden, weil einerseits die Performance nicht stimmte, aber auch weil ein Raycaster keine Punkte schneiden kann – und das sind unsere Sterne logisch im 3D-Raum: Punkte ohne Ausdehnung. Deswegen wurden sowohl für die Sterne als auch die Labels jeweils eigene optimierte “Hit Tester” geschrieben, die sehr effizient das passende Objekt (bzw. Den referenzierten Song) finden. Bei den Labels muss darüber hinaus noch beachtet werden, dass die Grösse der Fläche nicht mit der Grösse des Labels übereinstimmt, weil die Dimensionen wie bereits erwähnt auf 2^n aufgerundet werden mussten. Wir projektieren deshalb eine Subgeometrie, um den Test durchzuführen.

The Sound Of Silence

Eigentlich werden ganz simpel Mp3 abgespielt, sicher keine Hexerei, jedoch wollen mit zwei Limitationen verbunden, die wir umgehen wollen: auf iOS wird so kein Ton automatisch abgespielt und wir können keine schönen Übergänge gewährleisten. Die WebAudio API stellt sehr viel mächtigere Instrumente zur Verfügung. Via Audio Knoten können wir komplexe Audio-Routing-Graphen bilden. Unser Graph ist eher einfach strickt mit einem Gain-Knoten, um sauber ein- und auszublenden. Etwas aufwändiger war hingen die Ladelogik. Für den seltenen Fall, dass keine WebAudio API verfügbar ist, WebGL hingegen schon, greifen wir auf simple Audio-Elemente zurück und verzichten auf schöne Übergänge. Die betrifft eigentlich nur IE11.

Technische Verantwortung:
Severin Klaus, Head of Frontend Development, Partner bei Hinderling Volkart
sk@hinderlingvolkart.com / +41 78 896 63 65