Potree-Next Benutzerdokumentation

Call 18 | Projekt ID 6863

1 Einleitung

Potree-Next ist ein WebGPU-basierter Viewer für große Punktwolkendatensätze. Diese entstehen z.B. durch 3D-Aufnahmen ganzer Landschaftsstriche und Länder. Beispiele dafür sind

- Kappazunder – Virtuelles Abbild der Stadt (Wien).
  https://digitales.wien.gv.at/projekt/kappazunder/
  Aerial Lidar Scan 2023: 239 GB.
  Mobile Laser Scan + Bilder: ~40TB
- Actueel Hoogtebestand Nederlands (AHN): Ein Programm welches alle paar Jahre die gesamten Niederlande als 3D Datensatz im Zuge von Befliegungen aufnimmt. Die zweite Befliegung (AHN2) bestand aus 640 Milliarden Punkten (2.4TB komprimiert).
- 3D Elevation Program (3DEP): Aerial Laserscans der gesamten USA. Derzeit stehen 2201 Einzeldatensätze mit insgesamt 71 Billionen Punkten auf https://usgs.entwine.io/ zur Betrachtung zur Verfügung, unter anderem mit der WebGL-basierten Version von Potree.

Potree-Next ist ein Rewrite mit dem Ziel von WebGL auf den neuen Web-Standard WebGPU zu wechseln, und damit auch von neueren Graphics-APIs mit neuen Features und besserer Performance zu profitieren.

Source Code: https://github.com/m-schuetz/Potree-Next

2 Setup

2.1 Lokaler Testserver

Node.js installieren

http-server installieren: `npm install -g http-server`

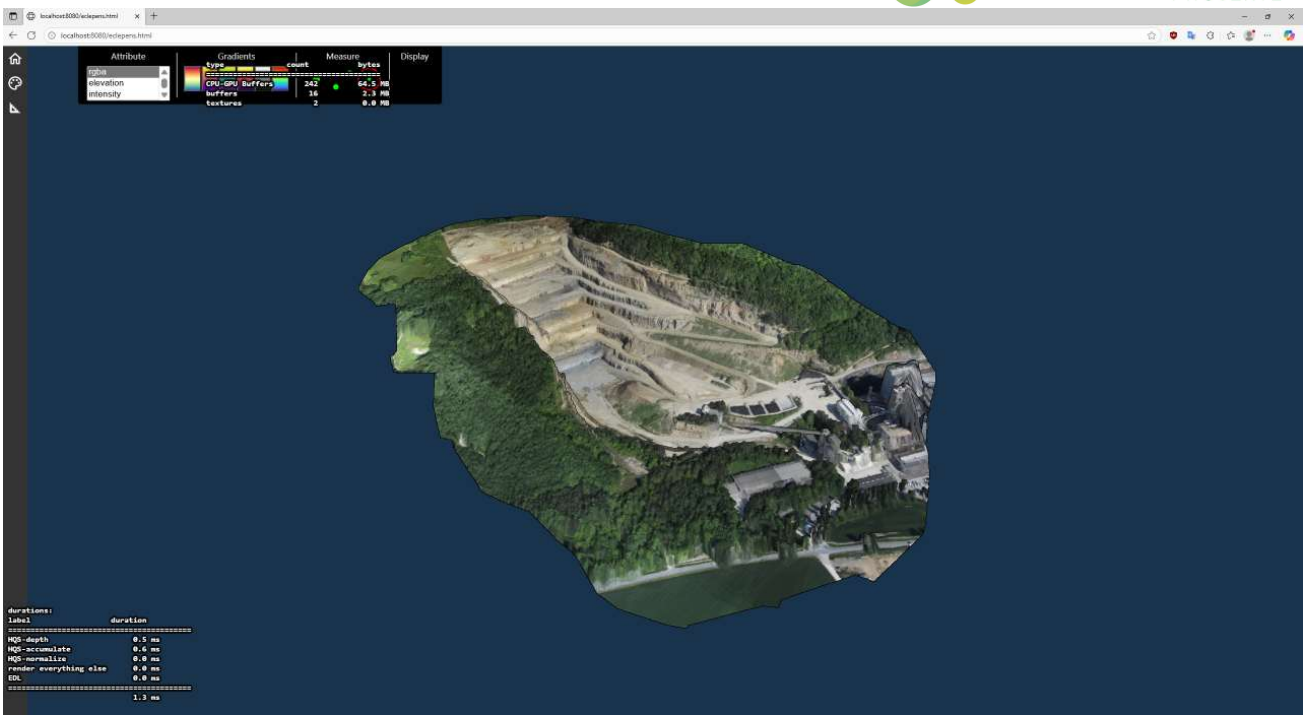
2.2 Potree-Next installieren

Projekt local klonen:

```
git clone https://github.com/m-schuetz/Potree-Next.git
```

Den lokalen Testserver aus dem Potree-Verzeichnis starten: `http-server`

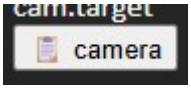
Die Example-Pages sind nun im Web Server unter http://localhost:8080/ erreichbar.



3 Visualisieren eigener Punktwolken

Eigene Punktwolkendatensätze müssen zuerst mit PotreeConverter in eine Octree-Beschleunigungsstruktur konvertiert werden: https://github.com/potree/PotreeConverter.

Anschließend empfehlen wir eines der vorhandenen Beispiele zu kopieren und die Links in der Datei durch einen Pfad zum eigenen Datensatz zu ändern. Dabei ist zu beachten, dass die Daten aus Sicherheitsgründen nur geladen werden können, wenn sie von einem Webserver gehostet werden. Ein Laden direkt von der Festplatte ohne Webserver ist nicht möglich. Mit dem im Kapitel 2 erwähnten „http-server“ wird soein lokaler Webserver eingerichtet, der Dateien aus dem Verzeichnis in dem er läuft über die URL http://localhost:8080 verfügbar macht.



Um einen Default-Viewpoint festzulegen der bei Programmstart angezeigt wird können Nutzer in der Sidebar den „camera“ Button klicken, welcher den aktuellen Blickwinkel in die Zwischenablage kopiert. Anschließend sollten die Argumente zu „controls.set“ im html file durch die Werte in der Zwischenablage ersetzt werden (ctrl + v).

4 Visualisieren eigener 3D-Tiles

3D-Tiles Datensätze können mittels Cesium Ion erstellt werden: https://ion.cesium.com/

Anschließend kann das 3dtiles.html Beispiel adaptiert werden um den entsprechenden 3D-Tiles Datensatz darzustellen.

5 Eigene Punktattribute

Das Beispiel extra_materials_terrasolid_sitn.html zeigt wie eigene Punktattribute, z.B. aus speziellen Anwendungen, verwendet werden können. Dazu muss einerseits ein kleines Shaderprogramm geschrieben werden welches die Bytes in Farben umwandelt, und andererseits muss dieses Shaderstück in Potree registriert werden damit es anschließend in der Sidebar zur Auswahl aufscheint. Ein minimales Programmstück um sich das „group“ Attribut aus Terrascan zu visualisieren schaut z.B. folgendermaßen aus:

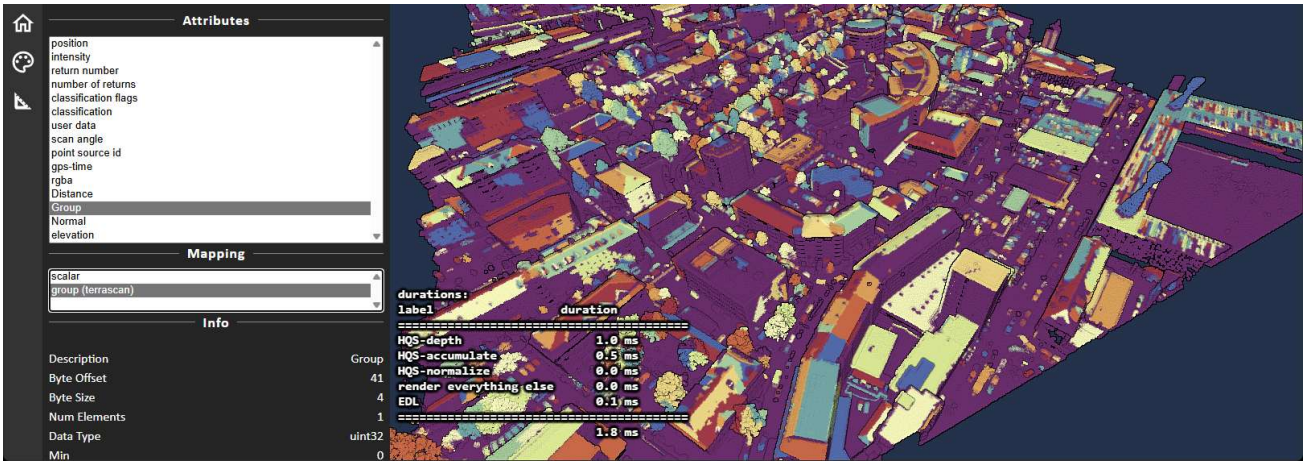
```
const TERRASCAN_GROUP = {
    name: "group (terrascan)",
    condition: (attribute) => (attribute.name === "Group"),
    inputs: [],
    wgsl: `
        fn mapping(
            pointID : u32,
            attrib : AttributeDescriptor,
            node : Node,
            position : vec4f
        ) -> vec4f {
            var offset = node.numPoints * attrib.offset + attrib.byteSize * pointID;
            var value = readU32(offset);

            var w = f32(value) / 1234.0;
            w = f32(value % 10u) / 10.0;
            var uv = vec2f(w, 0.0);

            return textureSampleLevel(gradientTexture, sampler_repeat, uv, 0.0);
        }
    `,
};

material.registerMapping(MAPPINGS.TERRASCAN_GROUP);
```

Das Resultat im Web Browser:



Der code registriert ein neues Attribut mit dem Namen „group (terrascan)“, welches dann im User Interface zur Auswahl aufscheint, wenn der Name des darzustellenden Punktattributes „Group“ ist. Der WGSL Shadercode der die Umwandlung von Bytes zu Farben übernimmt bekommt immer die Attribute pointID, attrib, node und position übergeben. Der AttributeDescriptor attrib beinhaltet Informationen über das Memory Layout der Bytes vom Attribut „Group“, und zusammen mit pointID können wir so die relevanten Bytes für den aktuell Prozessierten Punkt mittels readU32(offset) laden.