OSM Buildings

OpenStreetMaps Gebäudevisualisierung mit JavaScript

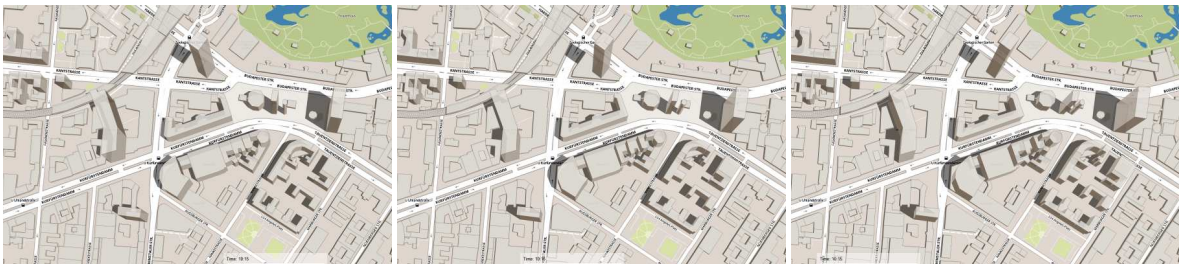
http://osmbuildings.org

OSMBuildings.org - Jan Marsch - Greifswalder Str. 159 - 10409 Berlin - mail@osmbuildings.org

Einführung

Das Projekt OSM Buildings macht abstrakte Gebäudegeometrien auf Web-Karten sichtbar.
Mit geringem Aufwand und ohne zusätzliche Software können z.B. OpenStreetMaps Daten auf verschiedenen interaktiven Karten, verschiedenen Browsern und Mobilgeräten problemlos dargestellt werden.
Durch die dynamische Komponente werden Kartendarstellungen für den Benutzer aussagekräftiger und laden ihn zu einer Entdeckungstour ein.

Die einfache Idee ist, Grundrisse um eine Höheninformation zu ergänzen und diese mit einer Perspektive zu versehen. Die Objekte werden auch in der Bewegung korrekt dargestellt. Im Web-Browser werden hierzu keine Besonderheiten wie WebGL, Java oder Flash benötigt. Es handelt sich um eine rein HTML5 & JavaScript basierte Lösung.



Bewegungssequenz in OSM Buildings

Integration

Am Beispiel der Open Source Kartenkomponente LeafletJS[1] wird gezeigt, wie einfach die Bibliothek in vorhandene Projekte zu integrieren ist.

```
<!DOCTYPE html>
<html>
  <head>
  <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7/leaflet.css">
  <script src="http://cdn.leafletjs.com/leaflet-0.7/leaflet.js"></script>
  <script src="OSMBuildings-Leaflet.js"></script>
</head>

<body>
  <div id="map"></div>

  <script>
  var map = new L.Map('map').setView([52.50440, 13.33522], 17);
  new L.TileLayer('http://{s}.tiles.mapbox.com/v3/your-key/{z}/{x}/{y}.png', maxZoom: 17 })
    .addTo(map);

  new OSMBuildings().addTo(map);
  </script>
</body>
</html>
```

Karte allgemein

Im HEAD Bereich einer HTML Seite werden die benötigten Style- und JavaScript Dateien eingebunden.

Im BODY Bereich wird ein DIV Container für die Kartendarstellung erzeugt und dann anschließend im SCRIPT Bereich referenziert.

LeafletJS wird mit *new L.Map({ parameters })* initialisiert und auf eine Ausgangsposition (Latitude / Longitude / Zoomstufe) festgelegt. Damit auch Karteninformationen in Form von Map Tiles sichtbar sind, wird noch eine Ebene mit *L.TileLayer({ parameters })* hinzugefügt. Hier sind verschiedene Anbieter möglich, z.B. MapBox[2], Google Maps, Nokia Maps etc.

OSM Buildings

Auf diese Basis setzt OSM Buildings eine weitere Darstellungsebene auf und reagiert auf Veränderungen der Hauptkarte wie z.B. Bewegen, Vergrößern, Verkleinern. Da es sich lediglich um eine zusätzliche Anzeigeebene handelt, sind andere GIS Funktionen wie Geolokalisierung, Suche, Routing etc. problemlos möglich.

Diese Ebene wird per *new OSMBuildings ({ parameters })* initialisiert. Je nach gewünschter Datenquelle kann hier gleich eine URL zu einem Serverdienst mit übergeben werden.

Analog zum TileLayer wird die Ebene mit *.addTo(map)* hinzugefügt. Wände, Schattierungen und Dächer können nach Benutzerwunsch mit *.setStyle ({ parameters })* eingefärbt werden.

Frontend

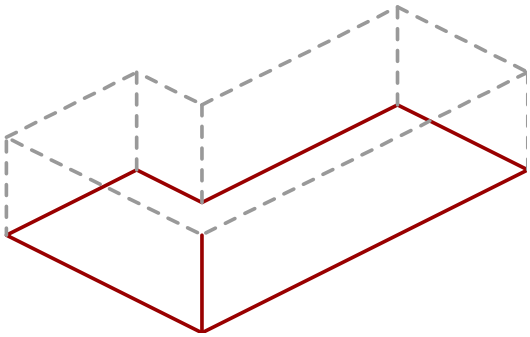
Für die Darstellung im Frontend bedient sich OSM Buildings der HTML5 Technologie Canvas mit JavaScript.

Derzeit wird noch auf den Einsatz von Grafikbeschleunigung durch WebGL verzichtet, da die Hardwareanforderungen relativ hoch sind und die Verbreitung auf vielen Mobilgeräten noch nicht gegeben ist.

Die Darstellung bedient sich daher geschickt einiger Verfahren um die räumliche Wirkung möglichst performant zu erzeugen.

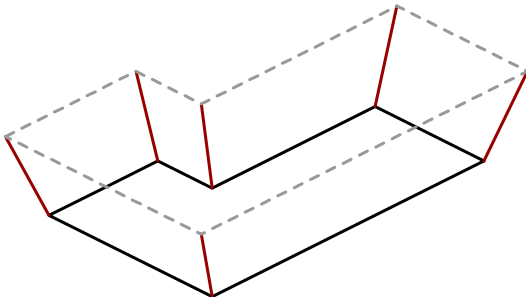
Als Hauptengpässe haben sich erwartungsgemäß Datenmengen, jegliche Operationen in JavaScript als auch Füllraten (Größe und Menge von Polygonen) in Canvas herausgestellt. Daher wird auf externe Bibliotheken verzichtet und der Programmcode so schlank wie möglich gestaltet.

Datenmenge reduzieren



Je nach Zoomstufe wird die Komplexität der Polygone automatisch reduziert.[3] Es erfolgt außerdem keine Verarbeitung kompletter 3D Modelle, da sich mit Grundrisspolygon und Höhe die komplette Dachkante reproduzieren lässt. Durch die gleichbleibende Höhe können weitere Rechenschritte eingespart werden.

3D Projektion

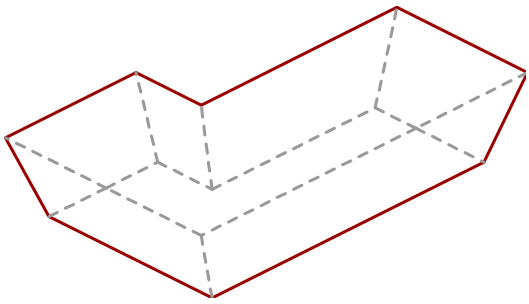


Für volle 3D Darstellung müssten üblicherweise Faktoren wie Drehung und Neigung berücksichtigt werden. Daraus ergeben sich zur Darstellung sehr rechenintensive Matrixoperationen, die im Falle von OSM Buildings erheblich vereinfacht werden können.

Um einem 3D Vektor (x, y, z) bei entsprechender Kameraposition zu projizieren, ist lediglich folgende Berechnung notwendig:

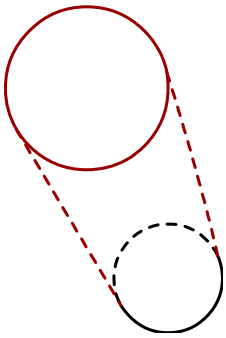
```
X = (x - cameraX) * cameraZ / (cameraZ - z) + cameraX
Y = (y - cameraY) * cameraZ / (cameraZ - z) + cameraY
```

Flächen kombinieren



Zur Reduzierung von Fülloperationen in Canvas werden für einige Zeichenschritte Polygonflächen zusammengefasst. Ursprünglich würden hier 6 Wandflächen plus 1 Dachfläche gezeichnet werden, diese können je nach Situation zu einer einzigen Fläche zusammen gefasst werden.

Zylinder



In 3D Umgebungen stellen abgerundete Formen oft eine besondere Herausforderung dar. Kanten und Schattierung sollen so „weich“ wie möglich dargestellt werden, was eine sehr hohe Zahl an Flächen erfordert. OSM Buildings benötigt für diesen Zylinder genau 2 Flächen: die Dachfläche als Kreis plus die Mantelfläche. In 2D Geometrie gedacht, ergeben sich die Flächen aus dem Grundflächenkreis, dem Dachkreis plus den Kreistangenten.

Daten

Die Qualität der Darstellung ist unmittelbar von den verfügbaren Daten abhängig.

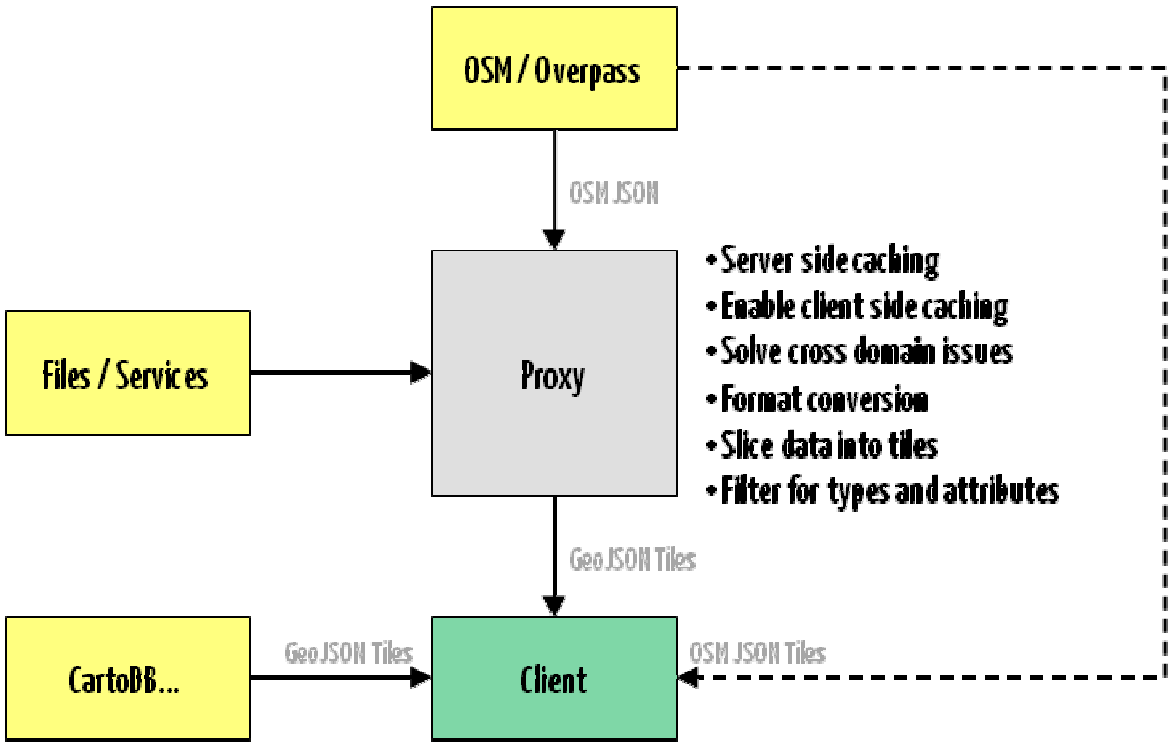
In OpenStreetMaps sind weltweit 70 Millionen Objekte als Gebäude markiert, lediglich ein Prozent der Objekte verfügt über eine Höhenangabe. [4]

Es werden mindestens Grundrißdaten als Multipolygon mit Latitude / Longitude Koordinaten und eine Höheninformation benötigt. Wünschenswert sind zusätzlich Angaben zur Höhe über Grund (OSM: *min_height*), Gebäudezweck (OSM: *amenity*), Material, Farbe und Dachform. Falls Höhenangaben nicht verfügbar sind, können auch Angaben über Stockwerke (OSM: *levels*, angenommene Standardhöhe 3m) herangezogen werden.[5]

OpenStreetMaps

OSM Daten bieten freie Verfügbarkeit, ein einfach strukturiertes Format und eine gute Abdeckung in Deutschland und Europa. Die Informationen sind regelmäßig sehr aktuell und müssen dank Overpass API nicht direkt vorgehalten werden.

Nachteile sind die weltweit sehr unregelmäßige Abdeckung, die schiere Menge an Daten und die eine Infrastruktur sehr von Spenden und persönlichem Engagement abhängig ist.



Import und Datenfluss

GeoJSON

Entgegen früheren Versionen verarbeitet OSM Buildings keine XML Formate mehr. Es wird nun ausschließlich GeoJSON[6] unterstützt. Alle anderen Formate werden wenn möglich zur Laufzeit, durch eine Serverkomponente umgewandelt. Die Daten werden dabei in Vektor-Tiles unterteilt, gefiltert, komprimiert und für Caching aufbereitet.

```
{
  "type": "FeatureCollection",
  "features": [{
    "type": "Feature",
    "geometry": {
      "type": "Polygon",
      "coordinates": [
        [
          [13.42706, 52.49535], [13.42675, 52.49503],
          [13.42694, 52.49496], [13.42678, 52.49480],
          [13.42657, 52.49486], [13.42650, 52.49478],
          [13.42686, 52.49466], [13.42714, 52.49494],
          [13.42692, 52.49501], [13.42717, 52.49525],
          [13.42741, 52.49516], [13.42745, 52.49520],
          [13.42745, 52.49520], [13.42706, 52.49535]
        ]
      ]
    },
    "properties": {
      "height": 30,
      "color": "rgb(180,240,180)"
    }
  }]
};
```

Beispieldatensatz GeoJSON

Eine kompakte Variante dieses Formates ist TopoJSON[7]. OSM Buildings wird dieses Format voraussichtlich unterstützen.

```
// link to CartoDB
new OSMBuildings().addTo(map).geoJSON(
  'http://cartodb.com/api/v2/sql?q=SELECT * FROM buildings&format=geojson'
))
```

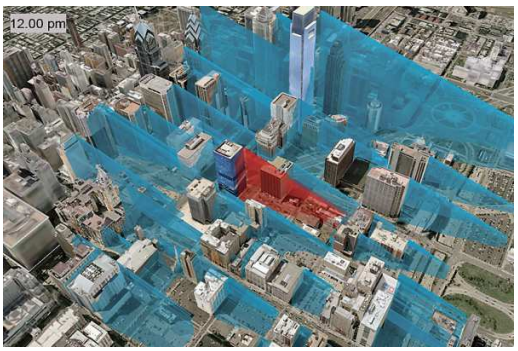
Beispiel externe Datenquelle mit GeoJSON

Gegenüberstellung

OSM Buildings erhebt nicht den Anspruch fotorealistischer Darstellung. Der Ansatz orientiert sich am Endanwender und wird daher wissenschaftlichen Anforderungen kaum gerecht.

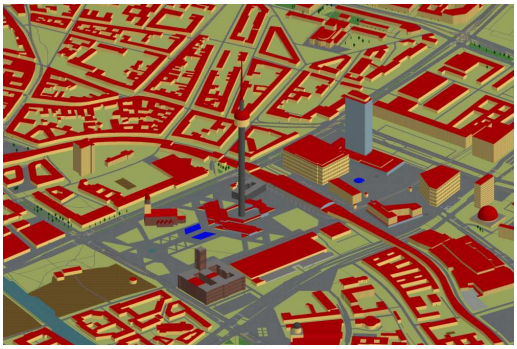
Als effektvolle Alternativen seien die WebGL basierten Lösungen von Google, Here und Apple genannt. Diese erfordern teilweise spezielle Apps und produzieren pro Stadt Datenmengen im Bereich mehrerer Gigabytes. Die sehr attraktiven Darstellungen bestehen dabei allerdings aus „dummen“ Polygonen, bei denen nicht zwischen Haus, Baum oder Straße unterschieden werden kann.

Quelle: Here WebGL[8]



Das andere Extrem, der wissenschaftliche Ansatz wird z.B. mit den Stadtmodellen von Esri VirtualCity verfolgt. Die Darstellung ist dabei zumindest realitätsnah, was sich ebenso im Datenaufkommen nieder schlägt. Sämtliche Objekte sind klar strukturiert und zur Laufzeit auch manipulierbar. Damit lassen sich komplexe physikalische Simulationen durchführen.

Quelle: Esri VirtualCity



Ähnliche Projekte mit Bezug auf OpenStreetMaps sind z.B. OSM2World mit einem höheren Detailgrad als OSM Buildings - jedoch ohne dynamische Bewegung, OSM-3D mit gewisser Ähnlichkeit - aber auf Java basierend.

Quelle: OSM2World[9]

Stark inspiriert von OSM Buildings tritt F4 Maps auf. Eine sich sehr schnell entwickelnde Anwendung mit vielen interessanten Details. Das Entwicklerteam ist leider weitgehend unbekannt, der Quellcode nicht offen verfügbar.

Quelle: F4 Maps[10]

In den meisten Situationen bietet OSM Buildings die schnellere und flexiblere Lösung. Es funktioniert ohne Einschränkung auf sehr vielen modernen Mobilgeräten.

Kooperationen bestehen bereits zu verschiedenen Projekten bzw. sind möglich im Bereich der Datengewinnung und des -austausches. Die gemeinsame Entwicklung von Standards und neuen Features wie z.B. Dachformen wäre denkbar.

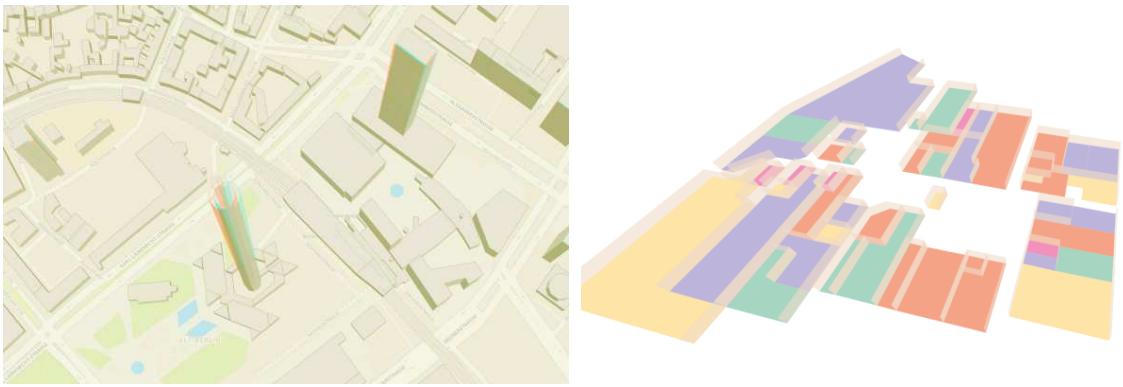
Ausblick

Der Nutzen für den Anwender steht klar im Mittelpunkt des Projektes. Bei zukünftigen Erweiterungen hat daher immer eine einfache und schnelle Bedienung Priorität.

Es ist geplant, die Darstellung der 3D Modelle deutlich realistischer zu gestalten. Allerdings soll hierbei nicht einfach nur ein weiterer WebGL-Renderer entstehen.

Die Qualität der verfügbaren Daten stellt dabei die größte Herausforderung dar. In Verbindung mit OSM Buildings entwickeln sich derzeit jedoch Projekte, die eine deutliche Verbesserung der Datenqualität herbeiführen können.

Es sei noch auf experimentelle Funktionen von OSM Buildings hingewiesen. Einerseits die anaglyphe 3D Darstellung für red-cyan Stereobrillen[11] als auch das Thema Indoor Mapping - hier als frei drehbares 3D Modell[12].



Quelle: OSM Buildings

[2] LeafletJS – http://leafletjs.com

[2] MapBox – http://mapbox.com

[3] Douglas-Peucker Algorithmus – http://de.wikipedia.org/wiki/Douglas-Peucker-Algorithmus

[4] OSM taginfo – http://taginfo.openstreetmap.org/keys/building

[5] Simple Building Models – http://wiki.openstreetmap.org/wiki/Simple_3D_Buildings

[6] GeoJSON – http://www.geojson.org/geojson-spec.html

[7] TopoJSON – https://github.com/mbostock/topojson/wiki

[8] Here WebGL – http://here.com/webgl

[9] OSM2World – http://maps.osm2world.org

[10] F4 Maps – http://map.f4-group.com

[11] OSM Buildings Anaglyph 3D – http://osmbuildings.org/anaglyph3d

[12] OSM Buildings Indoor 3D – http://osmbuildings.org/indoor