OSM Buildings

3D Gebäudevisualisierung mit JavaScript

http://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 dargestellt werden.
Durch die dynamische Komponente werden Kartendarstellungen für den Benutzer aussagekräftiger und laden ihn zu einer Entdeckungstour ein.

Quelle: CartoDB[1]

Die einfache Idee ist, Grundrisse um eine Höheninformation zu ergänzen und diese mit einer Perspektive zu versehen. Als Besonderheit werden die Objekte auch bei Bewegung korrekt dargestellt. Im Web-Browser wird hierzu kein 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[2] wird gezeigt, dass die Bibliothek sehr einfach in vorhandene Projekte zu integrieren ist.

```
<!DOCTYPE html>
<html>
  <head>
  <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.5.1/leaflet.css">
  <script src="http://cdn.leafletjs.com/leaflet-0.5.1/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[3], 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.

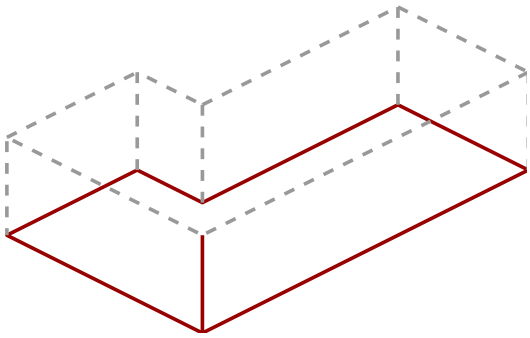
Frontend

Für die Darstellung im Frontend bedient sich OSM Buildings neuer HTML5 Technologien wie Canvas und JavaScript. Zum jetzigen Zeitpunkt wird auf WebGL verzichtet, da die Hardwareanforderungen relativ hoch sind und die Verbreitung auf vielen Mobilgeräten noch nicht gegeben ist.

Die Darstellung kann daher nicht auf komfortable 3D Funktionen zurück greifen und bedient sich einiger geometrischer Verfahren um den räumlichen Effekt zu erzeugen.

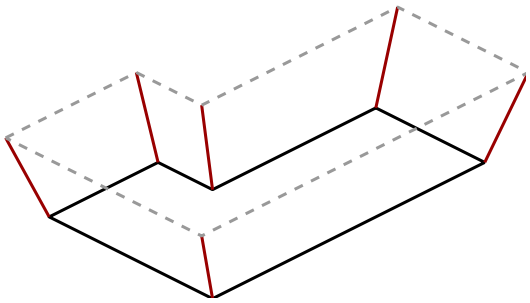
Als Hauptengpässe haben sich erwartungsgemäß Datenmengen, jegliche Operationen in JavaScript als auch Füllraten (Größe und Menge von Polygonen) in Canvas herausgestellt. Daher wird auf externe Bibliotheken verzichtet und der Programmcode so schlank und optimal wie möglich gestaltet.

Datenmenge reduzieren



Je nach Zoomstufe wird die Komplexität der Polygone automatisch reduziert.[4] 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

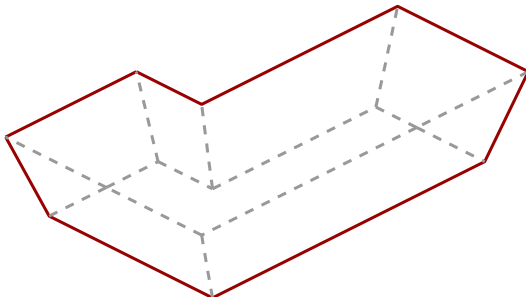


Für volle 3D Darstellung müssten üblicherweise Faktoren wie Drehung und Neigung berücksichtigt werden. Daraus ergeben sich zur Darstellung sehr rechenintensive Matrixoperationen, die im Falle von OSM Buildings erheblich vereinfacht werden können.

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

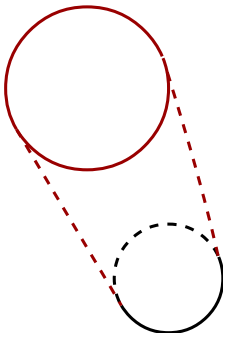
```
X = (x - cameraX) * cameraZ / (cameraZ - z) + cameraX
Y = (y - cameraY) * cameraZ / (cameraZ - z) + cameraY
```

Flächen kombinieren



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

Zylinder



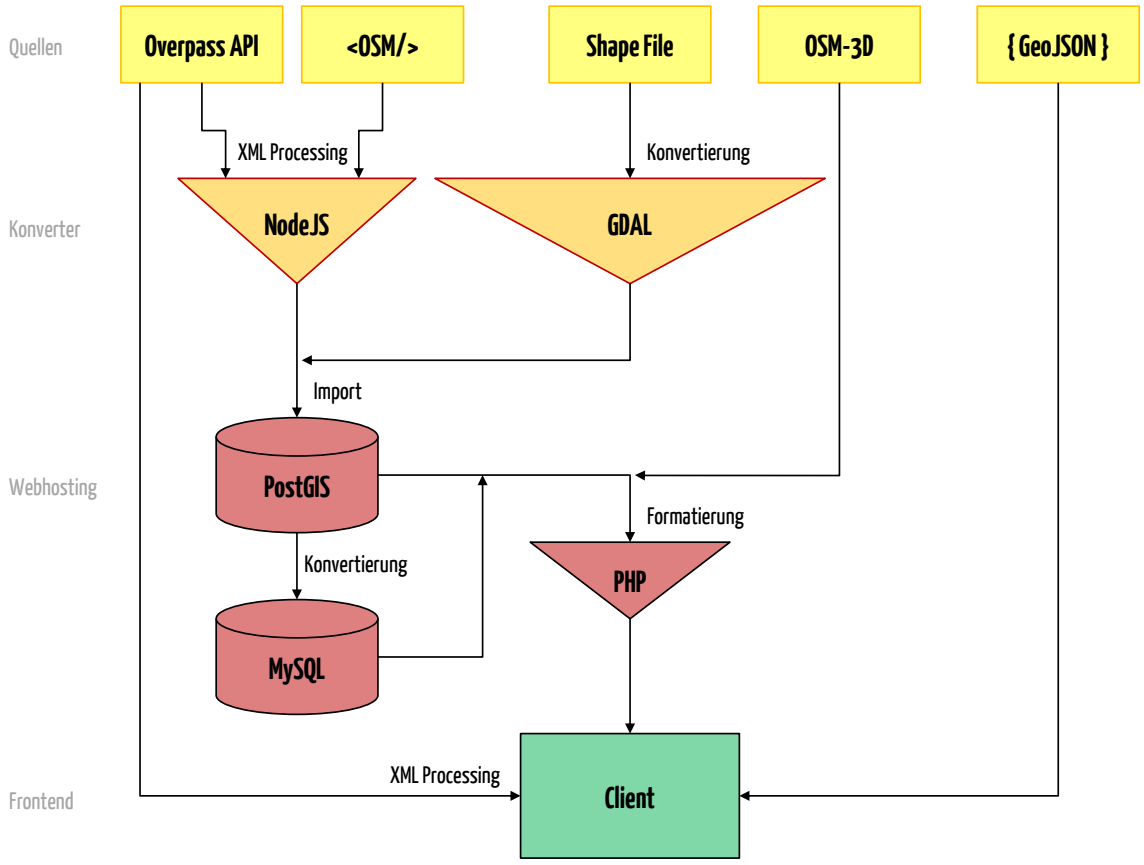
In 3D Systemen stellen abgerundete Formen oft eine besondere Herausforderung dar. Kanten und Schattierung sollen so „weich“ wie möglich dargestellt werden, was eine sehr hohe Zahl an Flächen erfordert. OSM Buildings benötigt für Zylinder genau 2 Flächen: die Dachfläche als Kreis plus die Mantelfläche.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

Daten

Die Qualität der Darstellung ist von den verfügbaren Daten abhängig.

In OpenStreetMaps sind weltweit 70 Millionen Objekte als Gebäude markiert, lediglich ein Prozent der Objekte verfügt über eine Höhenangabe. [5]

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*), Farbe und Dachform. Falls Höhenangaben nicht verfügbar sind, können auch Angaben über Stockwerke (OSM: *levels*, angenommene Standardhöhe 3m) herangezogen werden.[6]



Ablauf Import und Datenfluß

OpenStreetMaps XML Daten

```
<?xml version="1.0" encoding="UTF-8"?>
<osm>
  <node id="1952371852" lat="53.0888119" lon="8.8110002"/>
  <node id="60720575" lat="53.0889927" lon="8.8107713"/>
  <node id="60720576" lat="53.0894791" lon="8.8118365"/>
  …
  <node id="1952371845" lat="53.0887802" lon="8.8109309"/>
  <way id="8103911" timestamp="2013-01-01T16:52:07Z">
    <nd ref="1952371852"/>
    <nd ref="60720575"/>
    <nd ref="60720576"/>
    …
    <nd ref="1952371845"/>
    <nd ref="1952371852"/>
    <tag k="amenity" v="fair"/>
    <tag k="building" v="yes"/>
    <tag k="height" v="15m"/>
    <tag k="name" v="Messehalle 7"/>
  </way>
</osm>
```

Im Beispiel der OpenStreetMaps Daten werden Elemente vom Typ *way*, mit den Attributen *building* und *height* herangezogen. Im derzeitigen Stand von OSM Buildings werden diese Daten durch einen Serverdienst (PHP / MySQL) für schnelle Verarbeitung aufbereitet.

```
// url to backend data service
new OSMBuildings({ url: 'server/?w={w}&n={n}&e={e}&s={s}&z={z}' })


// result data format (JSON)
{
  meta: {
    n: 52.50400, w: 13.33000, s: 52.50100, e: 13.34000,
    x: 9009831, y: 5503399, z: 16
  },
  data: [
    [10,[302,2,303,7,295,30,290,27,292,24,288,22,291,19,296,22,302,13,296,10,302,2]],
    [10,[333,53,313,39,326,18,348,32,333,53]],
    [10,[313,39,333,53,332,55,326,64,323,68,302,55,304,51,317,60,322,52,309,44,313,39]],
    [25,[313,84,302,100,293,94,301,83,304,78,313,84]],
    [10,[304,78,301,83,281,69,273,80,268,76,278,62,284,65,304,78]],
    [10,[243,81,250,85,243,96,245,97,243,98,239,106,228,99,230,95,234,90,236,91,243,81]],
    [33,[251,20,259,26,252,39,265,46,260,55,246,48,240,58,231,52,251,20]],
    [8,[259,26,251,20,257,12,265,17,261,23,268,28,272,22,276,24,273,28,278,31,274,35,259,26]],
    [10,[116,30,100,35,95,22,99,21,106,19,111,17,116,30]],
    [10,[204,-15,206,-9,210,3,208,6,202,8,199,7,192,-11,204,-15]],
    [10,[210,3,217,7,213,13,206,9,208,6,210,3]],
    [14,[209,41,187,27,191,23,202,31,205,26,199,23,202,17,209,21,218,26,209,41]],
    [10,[158,1,175,-5,180,10,169,13,173,22,184,18,187,27,183,30,178,26,168,30,158,1]],
    [10,[175,-5,192,-11,199,7,199,11,194,13,191,7,187,8,188,15,183,16,180,10,175,-5]],
    [10,[158,1,168,30,166,37,167,39,158,42,126,54,124,54,116,30,111,17,111,16,157,0,158,1]],
    [16,[98,84,78,71,74,62,72,55,72,47,75,40,79,,23,95,22,100,35,105,49,117,56,110,65,98,84]],
    [21,[46,116,42,115,35,114,13,111,14,99,22,100,28,101,48,103,46,116]],
    [22,[12,11,12,18,-10,24,-9,14,-7,-15,-27,-72,-15,-71,0,-59,6,-40,14,-16,14,-14,12,11]]
  ]
}
```

Vorteile: freie Verfügbarkeit, einfaches Format, gute Abdeckung in Deutschland und Europa

Nachteile: sehr große Datenmengen, wenige Downloadquellen, Abdeckung unregelmäßig

Neben den statischen XML Dateien kann alternativ auch ein gewünschter Ausschnitt der Daten mittels Overpass API[7] bezogen werden. Der Verarbeitungsaufwand ist allerdings ähnlich. Künftig wird auch ein direkter Zugriff vom Frontend aus auf diesen Dienst zur Verfügung stehen.

Vorteile: individuelle und aktuelle Daten, keine Datenhaltung notwendig
Nachteile: eventuell Caching notwendig da Abfragen langsam sind

Eine weitere Möglichkeit in Vorbereitung ist der Datenservice von OpenStreetMaps 3D[8] der Universität Heidelberg.

Vorteile: sehr aktuelle Daten, keine Datenhaltung und Verarbeitung, notwendig, schneller Zugriff
Nachteile: eventuell begrenzte Verfügbarkeit

Shape Files

Abhängig vom Datenformat ist es über einige Konvertierungsschritte möglich Shapefiles zu importieren, z.B. Esri Multipatch.

Vorteile: sehr detaillierte Modelle, ergänzt Regionen mit schlechter OSM Abdeckung
Nachteile: aufwändige Konvertierung, nicht alle Attribute vorhanden

GeoJSON

Eine der einfachsten Möglichkeiten stellt der Direktzugriff auf GeoJSON[9] [QUELLE] Daten dar. In der Struktur werden entweder *Altitude*-Information pro Koordinate oder eine Property *height* benötigt.

```
{
  "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)"
    }
  }]
};
```

Eine solche Datei kann direkt auf dem Server abgelegt werden oder sie wird mit Diensten wie z.B. CartoDB sehr komfortabel dynamisch erzeugt.

```
// GeoJSON data
new OSMBuildings().addTo(map).geoJSON({
  "type": "FeatureCollection" ...
})

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

Eine neue, kompakte Variante dieses Formates ist TopoJSON[10]. OSM Buildings wird dieses Format kurzfristig unterstützen.

Vorteile: sehr individuelle Daten möglich, weitverbreitetes Format

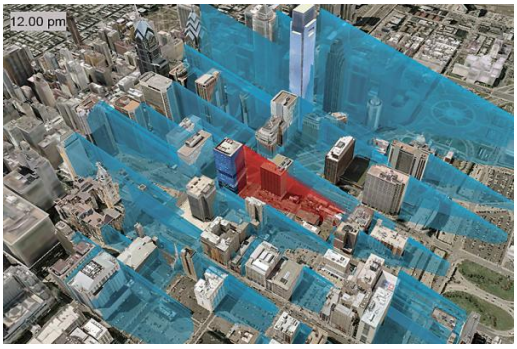
Nachteile: es wird immer der komplette Datensatz geladen

Gegenüberstellung

OSM Buildings positioniert sich außerhalb fotorealistischer 3D Darstellung einerseits und wissenschaftlich orientierten Systemen andererseits.

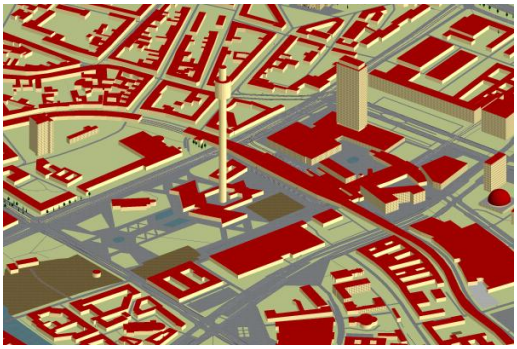
Als effektvolle Alternativen seien die WebGL basierten Lösungen von Google, Nokia und Apple genannt. Diese erfordern teilweise spezielle Apps und produzieren pro Stadt Datenmengen im Bereich mehrerer Gigabytes. Die sehr attraktiven Darstellungen bestehen dabei allerdings aus „dummen“ Polygonen, bei denen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

Quelle: Nokia Maps WebGL



Der wissenschaftliche Ansatz wird 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[11] mit einem höheren Detailgrad als OSM Buildings - jedoch ohne dynamische Bewegung, OSM-3D mit gewisser Ähnlichkeit - aber auf Java basierend und MiniWikiAtlas[12] - mit großer Ähnlichkeit und auf WebGL basierend.

Quelle: OSM2World

In einigen Situationen bietet OSM Buildings die schnellere Lösung und funktioniert auf vielen modernen Mobilgeräten. Es bietet Details, die in den anderen Systemen noch nicht vorhanden sind. Als Beispiel seien tages- und jahreszeitabhängige Schattenwürfe genannt.

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.

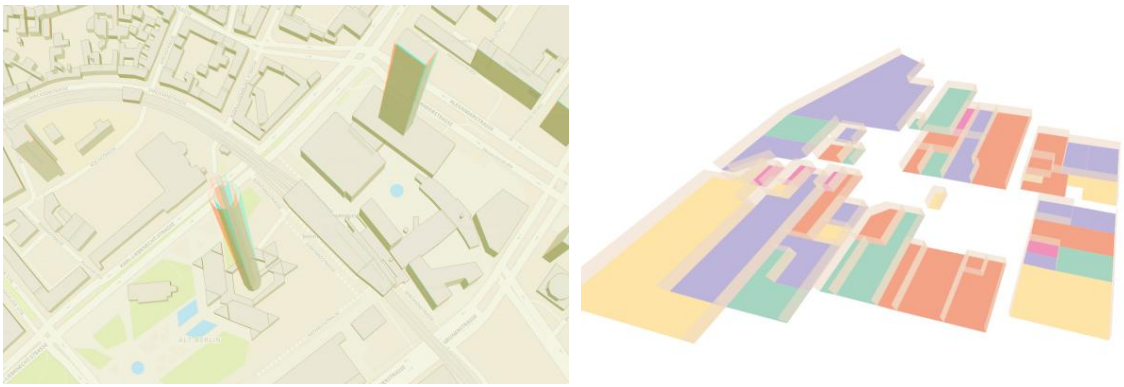
Ausblick

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

Um OSM Buildings für Anwender und Betreiber noch attraktiver zu gestalten, ist geplant weitere Kartensysteme zu unterstützen, z.B. Google Maps, Bing Maps und Esri. Die Darstellung der 3D Modelle wird schrittweise um Dachformen, Kuppeln und zylindrische Objekte erweitert. Langfristig wird WebGL als alternative Darstellungsform in das Projekt integriert.

Neue Datenquellen werden deutlich einfacher zu handhaben sein. Idealerweise soll keine eigene Datenhaltung mehr notwendig sein.

Es sei noch auf experimentelle Funktionen von OSM Buildings hingewiesen. Einerseits die anaglyphe 3D Darstellung für red-cyan Stereobrillen[13] als auch das hochaktuelle Thema Indoor Mapping - hier als echtes 3D Modell[14].



Quelle: OSM Buildings

[1] CartoDB – http://cartodb.com

[2] LeafletJS – http://leafletjs.com

[3] MapBox – http://mapbox.com

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

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

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

[7] Overpass API – http://wiki.openstreetmap.org/wiki/Overpass_API/Language_Guide

[8] OSM-3D – http://www.osm-3d.org

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

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

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

[12] MiniWikiAtlas – http://toolserver.org/~dschwen/wma/iframe_dev.html?wma=41.881944_-87.627778_700_500_en_15_en&globe=Earth&lang=en&page=Chicago

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

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