Browserbasierte Visualisierung großer 3D-Stadtmodelle durch Erweiterung des Cesium Web Globe

Zhihang YAO, Kanishk CHATURVEDI und Thomas H. KOLBE

Lehrstuhl für Geoinformatik · Technische Universität München · Arcisstraße 21 · 80333 München
E-Mail: zhihang.yao|kanishk.chaturvedi|thomas.kolbe@tum.de

1 Einleitung

Semantische 3D-Stadtmodelle beschreiben die urbane Topographie durch die räumlich-semantische Repräsentation der grundlegenden Objekte der Stadt wie z. B. Gebäude, Brücken, Gewässer, Verkehrswege und Gelände. Das einheitliche semantische Modell des internationalen Standard City Geography Markup Language (CityGML) des Open Geospatial Consortiums (OGC) ermöglicht, dass Stadtmodelle zunehmend auch als Plattform für die Integration städtischer Fachdaten verstanden werden. Immer mehr Städte im In- und Ausland verfügen über CityGML-basierende semantische 3D-Stadtmodelle und setzen sie in diversen Anwendungsgebieten ein, z. B. in der Stadtplanung, für energetische Simulationen, zur Navigation oder im Katastrophenmanagement (vgl. KOLBE 2009, GRÖGER et al. 2012). Die Mehrheit dieser Anwendungen erfordert einen einfachen Zugriff auf die 3D-Modelle und die Simulationsergebnisse. Auch müssen die 3D-Modelle oftmals vielen Benutzern (bis hin zur Öffentlichkeit) bereitgestellt werden. Im Hinblick auf diese Anforderungen ermöglichen webbasierte Geoinformationssysteme (Web-GIS) den breiten Zugang zu den weltweit verteilten geografischen Daten und die Bereitstellung von GIS-Funktionen über das Internet. Die besonderen Anforderungen bestehen dabei gleichermaßen in der performanten Datenübertragung und 3D-Visualisierung der Stadtmodelle sowie der Darstellung und Abfrage der semantischen Informationen.

Dank neuer Technologien wie der Grafikbibliothek WebGL und HTML5 kann die Visualisierung der 3D-Geodaten heutzutage direkt im Webbrowser realisiert werden. HTML5 ist dabei die neueste Generation der textbasierten Auszeichnungssprache HTML zur Anzeige von Multimedia-Inhalten in modernen Webbrowsern. HTML5 bietet eine interoperable Plattform zur Entwicklung komplexer Webanwendungen, die mittels Multi-Threading-Mechanismus auch die parallele Ausführung verschiedener Aufgaben innerhalb des Webbrowsers durchführen kann. WebGL ist eine Erweiterung des HTML5-Canvas-Elements, das heute sehr oft für die Entwicklung von webbasierten GIS-Anwendungen zur 2D- und 3D-Visualisierung der grafischen Daten verwendet wird (MARRIN 2013). Im Vergleich zu den früheren Plug-in-basierten Ansätze (z. B. Google Earth Plug-in) bestehen die Vorteile der WebGL-Technologie darin, dass die für die Anwendung relevanten 3D-Funktionen direkt im Webbrowser auf allen gängigen Betriebssystemen ohne zusätzliche Installation oder Aktivierung der benötigten Plug-ins ausgeführt werden können. Zudem verwendet WebGL die Grafikkarte für hardwarebeschleunigtes 3D-Rendering und ermöglicht daher eine sehr performante Darstellung der 3D-Inhalte im Webbrowser (LUBBERS et al. 2011).

Heutzutage existieren bereits mehrere WebGL- und HTML5-basierte Lösungen von virtuellen Globen, welche zur dreidimensionalen Darstellung von Geodaten im Webbrowser die-

nen können. Zu den bekanntesten browserbasierten virtuellen Globen zählen z. B. OpenWebGlobe (OPENWEBGLOBE 2015), WebGL Earth (WEBGL EARTH 2015) und Cesium Virtual Globe (ANALYTICS GRAPHICS INC. 2015). Als Open Source Projekt wurde Cesium unter der Apache 2.0 Lizenz im Jahr 2012 von der Firma Analytical Graphics, Inc. (AGI) entwickelt, und es wird seitdem durch eine große weltweite Entwickler-Community kontinuierlich mitentwickelt. Praktisch gesehen handelt es sich um eine JavaScript-Bibliothek zur Erstellung eines 3D-Webviewers für die Visualisierung von Geodaten, die aus verschiedenen Quellen stammen und in verschiedenen Datenformaten gegeben sein können. Wie die anderen virtuellen Globen auch, stellt das Cesium-Framework verschiedene Grundkarten (Luftbilder, Karten) zur Verfügung. Diese können durch eigene Grundkarten ergänzt werden, wobei auch OGC Web Map Services eingebunden werden können. Cesium unterstützt die Visualisierung vordefinierter sowie hoch detaillierter eigener Geländemodelle und die Darstellung von statischen und dynamischen Geodaten in 2D- und 3D-Ansicht. Cesium bietet damit ein umfassendes Software-Grundgerüst für den Aufbau und Betrieb einer 3D-Web-GIS-Anwendung zur grafischen Darstellung der 3D-Stadtmodelle. Cesium unterstützt aber weder die direkte Visualisierung von CityGML-Modellen im Speziellen noch von beliebig großen 3D-Stadtmodellen im Allgemeinen. Aufgrund der Tatsache, dass CityGML-basierende semantische 3D-Stadtmodelle in der Regel ein sehr großes Datenvolumen besitzen, ergibt sich die besondere Herausforderung, dass ein Verfahren zur effizienten Übertragung und Visualisierung der Geodaten entwickelt werden und in Cesium implementiert werden muss, das große 3D-Stadtmodelle in geeigneter Form zu den Nutzern bringt und performant im Webbrowser visualisiert werden kann.

Ziel der in diesem Beitrag vorgestellten Arbeiten ist es, eine Lösung zu entwickeln, die einerseits die performante Visualisierung großer semantischer 3D-Stadtmodelle auf Basis von HTML5 und WebGL Technologien ermöglichen soll, wobei auch die Anreicherung der Stadtmodell-Objekte um thematische Informationen von unterschiedlichen Sachdatenanbieter ermöglicht werden soll. Andererseits soll der Ansatz die Einbindung von zusätzlichen heterogenen Datenquellen wie z. B. Satelliten- und Luftbilder, digitale Geländemodelle und thematische Karten erlauben, die im 3D-Web-GIS für eine realitätsnahe Darstellung der semantischen 3D-Stadtmodelle notwendig sind. Dieser Ansatz lässt sich durch die Erweiterung der bestehenden Klassen der Cesium-Bibliothek um zusätzliche Funktionen zur performanten Visualisierung großer 3D-Stadtmodelle realisieren. Dabei wurde auch eine neue Systemarchitektur für 3D-Web-GIS-Anwendungen konzipiert.

2 Architektur der 3D-Web-GIS-Anwendung

Die von uns entwickelte 3D-Web-GIS-Anwendung ist in einer Client-Server-Architektur aufgebaut, die einen funktional umfassenden 3D-Webclient mit einer intuitiv erlernbaren GUI bereitstellt. Benutzer können Projekte anlegen, in die mehrere und verschiedenartige Datenserver in Form von Layern eingebunden werden können. Die Verwendung standardisierter Schnittstellen gewährleistet dabei ein hohes Maß an Interoperabilität. Endnutzer wie beispielsweise Architekten oder Stadtplaner können ihre eigenen 3D-Modelle in standardisierten 3D-Grafikformaten wie KML oder glTF hinzuladen. Auch können sie eigene Kartenhintergründe mittels OGC Web Map Services einbinden.

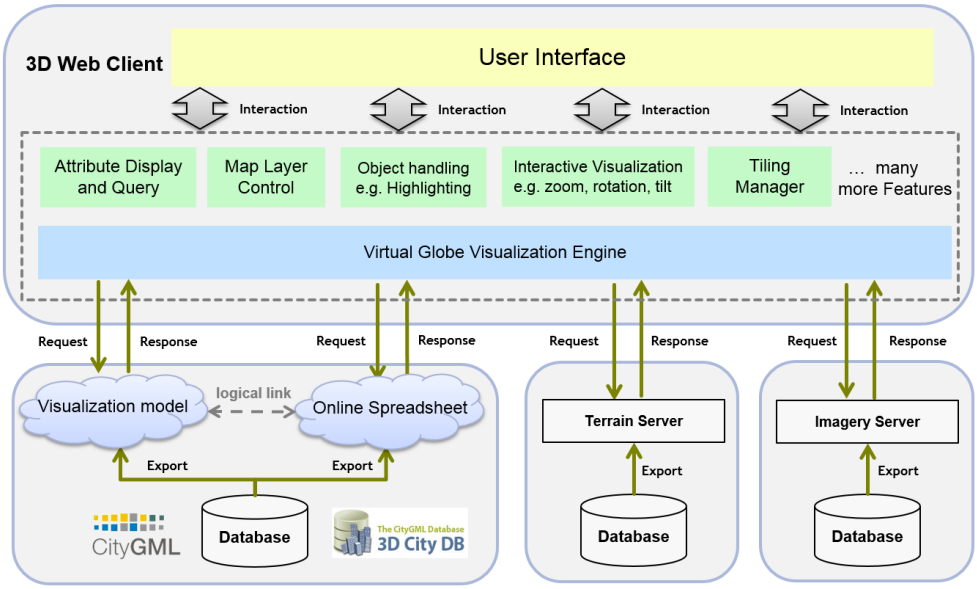

Abb. 1: Systemarchitektur der 3D-Web-GIS-Anwendung.

Abbildung 1 zeigt die Architektur der 3D-Web-GIS-Anwendung, wobei zwischen dem 3D-Webclient und den Datenservern für Geländedaten, Bilddaten und den 3D-Stadtmodelldaten unterschieden wird.

Geländemodelle dienen zur 3D-Darstellung der relevanten Geländemerkmale (z. B. Gewässer, Berge und Täler etc.). In der Regel sind hochauflösende Geländemodelle sehr umfangreich; der benötigte Speicherplatz liegt oftmals bei mehreren Gigabytes bis hin zu Terabytes. Um die effiziente Visualisierung des Geländemodells im Webbrowser zu gewährleisten, wird das Geländemodell vorab aus der Geodatenbank exportiert und gemäß des Tile Map Service (TMS) Standards (TILE MAP SERVICE SPECIFICATION 2012) je Zoomstufe zu mehreren kleinen Kacheln aufgeteilt und in das von Cesium unterstützte Rasterformat (.terrain) umgewandelt. So kann Cesium je nach der aktuellen Zoomstufe und Kameraperspektive die benötigten Geländemodellkacheln laden und anschließend in der 3D-Karte grafisch darstellen. Zudem besteht die Möglichkeit, externe Geodaten wie z. B. Satelliten- und Luftbilder über OGC Web Map Services hinzu zu laden, die dann, dem Geländemodell visuell überlagert, in der 3D-Web-GIS-Anwendung dargestellt werden.

Die Repräsentation der semantischen 3D-Stadtmodelle erfolgt in CityGML, um die verschiedenen Objektarten der Stadtmodelle in einen gemeinsamen Datenbestand zu integrieren und die semantische Interoperabilität sicherzustellen. Die Datenhaltung erfolgt in der Open Source 3D-Geodatenbank „3DCityDB“. Dabei handelt sie sich um ein relationales Datenbankschema mit räumlichen Erweiterungen für Oracle Spatial bzw. PostGIS zur effizienten Speicherung und Verwaltung der CityGML-basierenden semantischen 3D-Stadtmodelle (STADLER et al. 2009). In der Praxis besteht eine besondere Herausforderung darin, dass einerseits das relationale Paradigma, das der Datenbank zugrunde liegt, es notwendig macht, dass die räumlichen und thematischen Informationen zu den objektorientiert model-

lierten Stadtobjekten in viele unterschiedliche und über Fremdschlüssel miteinander verknüpften Datenbank-Tabellen aufgeteilt werden. Räumliche und thematische Abfragen erfordern daher die genaue Kenntnis des Datenbankschemas, das für die Anwender – die in der Regel weder Geoinformatiker noch GIS-Spezialisten sind – unverständlich ist. Andererseits können die in der Datenbank gespeicherten Geodatenmengen eine lange Prozessierungszeit und große serverseitige CPU-Auslastungen verursachen, wenn sie direkt vom Datenbank-Server abgerufen und zum Client übertragen werden. Um dieser Herausforderung gerecht zu werden, wurde eine cloudbasierte Lösung entwickelt, die durch die Entkopplung der Endnutzer von der CityGML-Datenbank realisiert wurde (siehe YAO et al. 2014).

Die Bereitstellung der 3D-Stadtmodelldaten erfolgt durch den Vorab-Datenbankexport von zwei Arten von Daten. Auf der einen Seite wird das für die Anwendung relevante 3D-Visualisierungsmodell im KML/glTF-Format aus der 3DCityDB exportiert und in mehrere Kacheln mit einer vom Benutzer spezifizierbaren Kantenlänge aufgeteilt. Alle Daten des 3D-Visualisierungsmodells werden anschließend auf einen einfachen Webserver oder in einen Cloud-Dienst (z. B. Dropbox) hochgeladen, sodass auf das Visualisierungsmodell direkt über das Internet performant zugegriffen werden kann. Auf der anderen Seite werden die Sachdaten der Stadtmodell-Objekte exportiert und pro Objektklasse auf eine einfache Tabellenstruktur abgebildet. Diese tabellarischen Sachdaten werden in einen entsprechenden Cloud-Dienst (derzeit „Google Docs“) hochgeladen. Die beiden Datensätze – das 3D-Visualisierungsmodell und die Sachdaten – werden über einen eindeutigen Schlüssel (die GML-IDs der Objekte) miteinander logisch verknüpft, sodass thematische Anfragen zu angeklickten 3D-Stadtmodell-Objekten clientseitig sehr einfach realisiert werden können. Nähere Details können (HERRERUELA et al. 2012, YAO et al. 2014) entnommen werden.

Der 3D-Webclient realisiert die Benutzerschnittstelle und ist ein wesentlicher Teil der 3D-Web-GIS-Anwendung. Der Cesium-Viewer bildet dabei das Grundgerüst des 3D-Webclients. Er stellt eine intuitiv bedienbare 3D-Visualisierungskomponente zur grafischen Darstellung der und Interaktion mit den Geodaten bereit. Es werden die gängigen Navigationsmöglichkeiten unterstützt wie z. B. Panning, Zoom und Rotation, die in der 3D-GIS-Karte notwendig sind. Weiterhin bringt der Cesium-Viewer standardmäßig viele Grundkartenvarianten (z. B. Esri World Street Map, Bing Maps und OpenStreetMap etc.) und ein globales Geländemodell (STK World Terrain Meshes mit 30 m Auflösung) mit. Eine Besonderheit des Cesium-Viewers ist die Unterstützung der „on-the-fly“ Umschaltung zwischen 2D- und 3D-Ansichten, sodass die 2D- und 3D-Ansichten in einer Plattform bzw. Programmierschnittstelle (API) gemeinsam zur Verfügung stehen. Für die Visualisierung der 3D-Objekte unterstützt der Cesium-Viewer das Laden von unterschiedlichen Datenformaten z. B. GeoJSON, KML, CZML (zu Letzterem siehe ANALYTICS GRAPHICS INC. 2015). Der Cesium-Viewer bietet auch die Möglichkeit, texturierte Modelle im glTF-Format einzubinden, welches von der Khronos Group als neue Spezifikation (Version 1.0) entwickelt wurde, um computergrafische 3D-Modelle effizient in WebGL-basierten Anwendungen zu visualisieren (KHRONOS 2015).

Zur performanten Visualisierung und thematischen Exploration der komplex strukturierten und großvolumigen 3D-Inhalte aus unterschiedlichen Datenquellen (z. B. Stadtmodell-Objekte, Geländemodelle, Satelliten- und Luftbilder) musste der Cesium-Viewer um weitere Module bzw. Funktionalitäten erweitert werden.

Dazu werden zunächst die Funktionsblöcke, z. B. das Laden und Entladen sowie die Funktion, die Daten der einzelnen Layer in der 3D-Ansicht ein- oder auszublenden, von den formatspezifischen Implementationen der verschiedenen Datenquellen abstrahiert, sodass die unterschiedlichen 3D-Inhalte der Daten-Layer auf einheitliche Weise vom Webclient behandelt werden können. Es wird somit eine Abstraktionsebene zwischen den unterschiedlichen Datenformaten und der grafischen Benutzerschnittstelle (GUI) des Webclients geschaffen, die dem Benutzer eine einheitliche Interaktion mit den Daten ermöglicht. Weiterhin unterstützt der Cesium-Viewer nicht ohne weiteres das „Highlighting“ beim Anklicken oder Berühren von 3D-Objekten mit dem Mauszeiger. Der Cesium-Viewer kann zwar für das Anzeigen der Sachinformationen (z. B. GML-ID, Description) des selektierten Stadtmodell-Objekts eine Informationstabelle darstellen; es ist aber nicht möglich, die Stadtmodell-Objekte um weitere thematische Informationen nachträglich anzureichern oder thematische Abfragen durchzuführen. Zum Zeitpunkt der Erstellung dieses Beitrags wird das Streaming und die Visualisierung großer 3D-Stadtmodelle vom offiziellen Release des Cesium Viewers nicht unterstützt. In der Cesium-Community werden aktuell Ansätze wie das „3D-Tiling“ vorangetrieben, um große Datenmengen performant visualisieren zu können. In diesen Ansätzen wird jedoch das KML-Format, welches bereits seit vielen Jahren in vielen Kommunen und öffentlichen Verwaltungen genutzt wird, nicht unterstützt. Eine zusätzliche Visualisierungsstrategie wird deswegen benötigt, um diese fehlende Funktionalität abzudecken. Vor diesem Hintergrund wurde der 3D-Webclient um entsprechende Erweiterungen des Cesium-Viewers ergänzt. Softwaretechnisch gesehen ist der 3D-Webclient eine Spezialisierung des Cesium-Viewers. Der technische Aufbau wird im folgenden Text sowie der Abbildung unten erläutert.

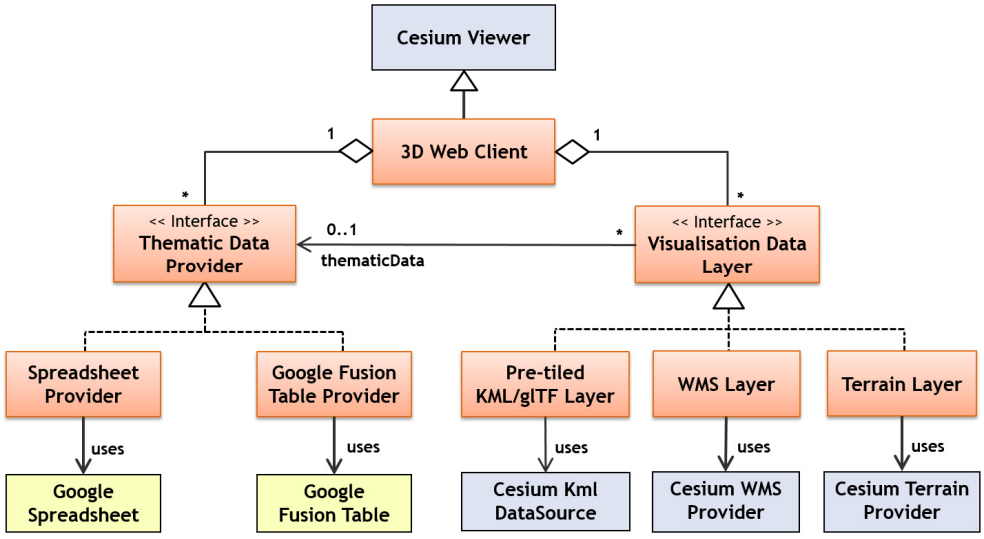

Abb. 2: Konzeptuelle Softwarearchitektur des 3D-Webclients in UML-Notation.

Die Softwarearchitektur des 3D-Webclients basiert auf der Anwendung des Strategie-Entwurfmusters (strategy pattern), das bei der Softwareentwicklung in der Regel dazu verwendet wird, dass eine spezifische Funktionalität eines Programms flexibel und schnell erwei-

tert werden kann (GAMMA et al. 1995). Zunächst wird eine Klasse `3DWebclient` definiert, die aus der Cesium-Klasse `CesiumViewer` abgeleitet ist, um seine Funktionalitäten zu erweitern, z. B. um die farbliche Hervorhebung der selektierten oder berührten Stadtmodell-Objekte. Weiterhin wird eine Schnittstelle (Interface) `VisualisationDataLayer` für diejenigen Methoden definiert, die für das Management wie dem Laden und Entladen, Ein- und Ausschalten der konkreten Visualisierungsdaten-Layer notwendig sind. Die Implementierung solcher Methoden wird gemäß der definierten Schnittstelle in den konkreten Daten-Layer-Klassen vorgesehen. Drei Daten-Layer-Klassen sind bereits implementiert. Dies sind z. B. `KmlGlTFLayer`, `WMSLayer`, und `TerrainLayer`, die jeweils die Cesium Klassen `KmlDataSource`, `WebMapServiceProvider`, und `TerrainProvider` verwenden. Die Klasse `3DWebclient` enthält eine Liste der Daten-Layer und kann die darin referenzierten Methoden aufrufen. Mit dieser Entwurfsstrategie wird hinsichtlich der Einbindung von Geodaten unterschiedlicher Quellen ein hohes Maß an Flexibilität und Dynamik gewonnen. Analog wurde eine zusätzliche Schnittstelle `ThematicDataProvider` definiert, die mit der Schnittstelle `VisualisationDataLayer` assoziiert ist. Zwei Klassen `GoogleSpreadsheetProvider` und `GoogleFustionTableProvider` stellen bereits konkrete Implementierungen dieser Schnittstelle dar und verwenden die durch Google APIs unterstützten Klassen `GoogleSpreadsheet` und `GoogleFusionTable`, die zur webbasierten Speicherung und Abfrage der Sachinformationen dienen. Die entwickelte Funktionalität des 3D-Webclients geht somit von der Layerbasierten flexiblen Einbindung der Geodaten unterschiedlicher Quellen über die interaktive Exploration der Stadtmodell-Objekte bis hin zur Anreicherung der Geodaten um thematische Informationen.

3 Streaming zur Visualisierung großer 3D-Stadtmodelle

Zur Visualisierung großer 3D-Stadtmodelle wird eine Kachelungsstrategie realisiert, die auf der Basis des 3DCityDB Open-Source-Programmpakets und des 3D-Webclients umgesetzt wird. Zunächst werden die in der 3DCityDB-Datenbankinstanz gespeicherten 3D-Stadtmodelle anhand des vom Nutzer ausgewählten geographischen Gebiets („Bounding Box“) in rechteckige Kacheln unterteilt, die in den Formaten KML (und bei texturierten Modellen in COLLADA/glTF) exportiert und anschließend in automatisch angelegte Unterordner abgelegt werden. Alle Kacheln werden dann auf einen Webserver hochgeladen, sodass die Kacheln über die HTTP-Schnittstelle vom 3D-Webclient zugegriffen werden können. Damit die Kacheln anhand der aktuellen 3D-Kameraperspektive im 3D-Webclient dynamisch geladen und entladen werden können, benötigt der 3D-Webclient Metainformationen, die die Dateistruktur und Eigenschaften der exportierten Kacheln beschreiben. Diese Metainformation lässt sich in einem JSON-basierten Format parallel zum Export der Kacheln ausgeben und gemeinsam mit den Kacheln auf den Webserver hochladen. Die folgende Abbildung stellt die in der JSON-Datei spezifizierten Parameter und einen Ausschnitt der exportierten Kachel-Dateien exemplarisch dar.

```
{
    "layername": „London_Building_Geometry,
    "fileextension": ".kmz",
    "displayform": „Geometry",
    "minLodPixels": 140,
    "maxLodPixels": 500,
    "colnum": 16,
    "rownum": 16,
    "bbox": {
        "xmin": -0.030311040,
        "xmax": 0.0063835629,
        "ymin": 51.507540717,
        "ymax": 51.529197282
    }
};
```

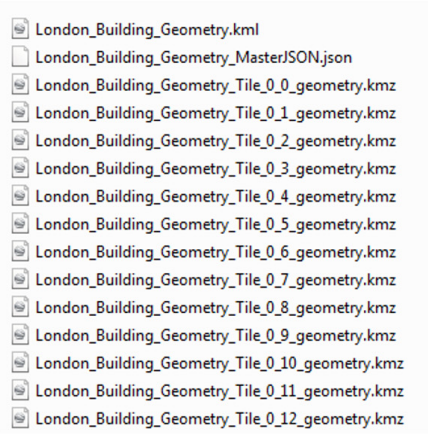

Abb. 3: Beispiel der JSON-Parameter und Dateistruktur der exportierten Kacheln.

Der Parameter `layername` gibt den Basisdateinamen aller Kacheln an, die in zwei möglichen Datenformaten – KML oder KMZ – exportiert werden können. Anhand des Parameters `fileextension` kann der Webclient ermitteln, in welchem Format die Kacheln serverseitig vorliegen. Der Parameter `displayform` dient zur Ermittlung der Visualisierungsstufen bzw. Darstellungsvarianten, die im 3D-Webclient visualisiert werden können. Vier unterschiedliche Darstellungsvarianten - *Footprint*, *Extruded*, *Geometry* und *Collada/glTF* – werden unterstützt, welche mittels des 3DCityDB-KML/COLLADA-Exporters erzeugt werden können (KOLBE et al. 2015). Die Darstellungsvariante „*Footprint*“ ist geeignet für die einfache LOD0-Repräsentation von Grundrissen der Stadtmodell-Objekte, die dem Geländemodell im virtuellen Globus überlagert dargestellt werden. Ferner gibt es eine weitere Darstellungsvariante „*Extruded*“, die das Extrudieren der 2D-Grundrisse anhand der Höhe der Stadtmodell-Objekte ermöglicht und deshalb sehr nützlich zur effizienten LOD1-Darstellung der generalisierten komplex strukturierten 3D-Objekte ist. Zur vollständigen Darstellung der Geometrien der Stadtmodell-Objekte (LOD > 1), die unterschiedliche Geometrietypen z. B. Punkte, Linien und Polygone beinhalten können, muss die Darstellungsvariante „*Geometry*“ gewählt werden. Für texturierte 3D-Modelle steht die Darstellungsvariante „*Collada/glTF*“ zur Verfügung. Die Parameter `minLodPixels` und `maxLodPixels` spezifizieren ein vom Nutzer festgelegtes Größenintervall in Pixeln, damit nur diejenigen Kacheln in den 3D-Webclient geladen werden, die im aktuellen Betrachtungsbereich sichtbar sind und deren Diagonale im Bildschirmausschnitt innerhalb des Größenintervalls liegen. Mithilfe verschiedener Konfigurationen kann eine maßstabsabhängige LOD-Auswahl realisiert werden, wobei Kacheln mit weniger Details (z. B. „*Extruded*“) nur in kleinen Maßstäben dargestellt werden, und Kacheln mit vielen Details (z. B. „*Geometry*“) nur geladen werden, wenn die Kacheln sehr nah zum Betrachter sind. Damit können die Lade- und die Renderingzeiten clientseitig enorm verringert werden.

Die Parameter `colnum` und `rownum` spezifizieren die Anzahl der Kacheln in Längen- und Breitengradrichtung. Der Parameter `bbox` stellt die globale Bounding Box aller exportierten Kacheln dar, wobei mit dem WGS84 dasselbe Koordinatenreferenzsystem wie der Cesium WebGlobe verwendet wird. Mit allen diesen Parametern ist es möglich, während

der Navigation die Kacheln mittels eines Kachelmanagers im 3D-Webclient dynamisch laden und entladen zu können.

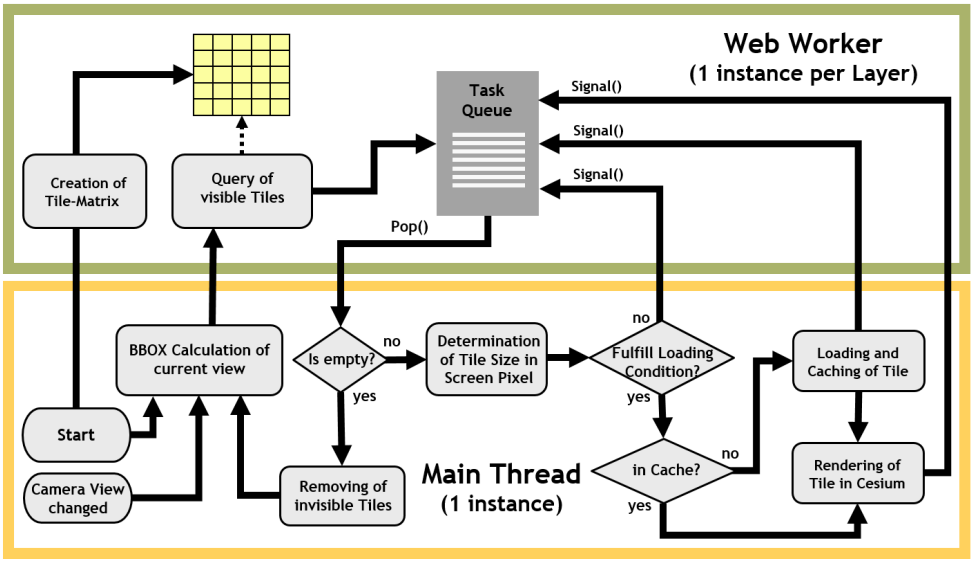

Abb. 4: Ablauf für das dynamische Laden und Entladen der Kacheln im 3D-Webclient.

Der Ablauf des dynamischen Ladens und Entladens der Kacheln wird in Abb. 4 gezeigt und läuft folgendermaßen ab: Zunächst wird im Webclient eine Kachel-Matrix aufgebaut. Der Ursprung der Kachel-Matrix liegt in der Südwestecke der geographischen Bounding Box, die in der JSON-Datei spezifiziert ist. Darüber kann die Kachel-Matrix sehr leicht durch die Zuordnung des Ursprungs der Kachelmatrix zum übergeordneten Koordinatensystem der globalen Bounding Box georeferenziert werden. Während der Navigation kann der 3D-Webclient anhand der Kameraperspektive ein auf der geographischen Ebene abgebildetes Polygon erzeugen, aus dem eine Bounding Box berechnet werden kann, die den aktuellen Betrachtungsbereich des Bildschirms abdeckt (siehe Abb. 5). Nun müssen die Kacheln bestimmt werden, die eine Überlappung mit dieser Bounding Box haben. Mit den in der JSON-Datei spezifizierten Parameterwerten, bzw. der Anzahl aller Kacheln in Längen- und Breitengradrichtung und der globalen Bounding Box können die Nummern der zu ladenden Kacheln in Längen- und Breitengradrichtung und die Bounding Box jeder Kachel berechnet werden, weil die Adressierung der Kacheln davon ausgeht, dass die Kacheln mit gleichen Kantenlängen im WGS84 geographischen Koordinatensystem unterteilt wurden. Die URL der individuellen Kachel kann relativ einfach gemäß folgender Regel aufgebaut werden:

URL der Kachel = URL des Wurzelordners + [layername] + „_Tile_" + [x] + „_" + [y] + „_" + [displayform] + „." + [fileextension]

Da die Dateinamen aller Kacheln beim Export so gewählt werden, dass die Nummer der Kachel im Dateinamen enthalten ist, können die im Webserver gespeicherten Kacheln direkt über eine eindeutige URL angesprochen werden. Der Zugriffe auf die Kacheln erfordert dadurch vom Server keine weitere Logik und kann sehr schnell durchgeführt werden.

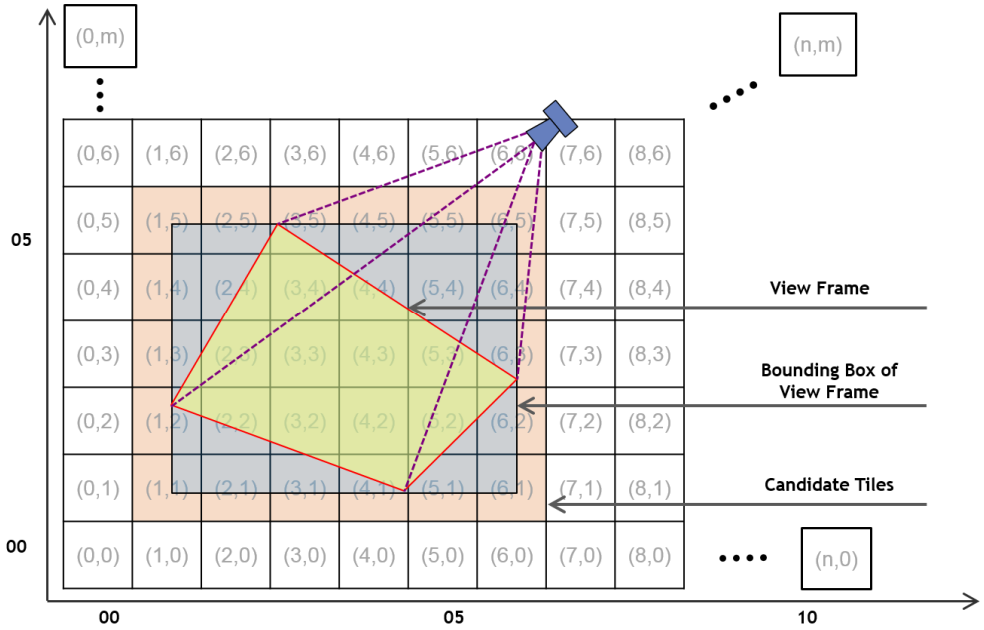

Abb. 5: Ermittlung der zu ladenden Kacheln anhand der Kameraperspektive.

Jede potenziell zu ladende Kachel wird nun als ein Javascript-Objekt repräsentiert, welches seine URL und Bounding Box beinhaltet. Diese Kachel-Objekte werden in einer Warteschlange (Queue) zwischengespeichert. Anschließend entnimmt der 3D-Webclient nacheinander ein Kachel-Objekt aus der Queue und ermittelt die Bildschirmkoordinaten der Kachel in Pixeln, um ein viereckiges Polygon auf dem Bildschirm abzubilden. Der 3D-Webclient überprüft, ob die kleinste Diagonale des Polygons innerhalb des in der JSON-Datei spezifizierten Größenintervalls liegt. Liegt die Pixelanzahl der Diagonale nicht im Größenintervall, dann wird das Kachel-Objekt ignoriert und das nächste Kachel-Objekt wird aus der Queue entnommen. Wenn die Pixelanzahl der Diagonale im Größenintervall liegt, wird für die 3D-Inhalte der Kachel überprüft, ob sie bereits im Cache vorhanden sind oder nicht. Sind die 3D-Inhalte bereits im Cache vorhanden, so werden die 3D-Inhalte der Kachel direkt aus dem Cache (im Hauptspeicher) an den Cesium-Viewer zur Visualisierung übergeben. Anderenfalls sendet der Webclient anhand der URL eine Anfrage an den Server, um die Kachel zu laden, ihre Inhalte zu visualisieren und anschließend im Cache zwischen zu speichern. Durch den Caching-Mechanismus wird die Anzahl der Ladevorgänge signifikant und spürbar reduziert. Aufgrund der guten Hauptspeicherausstattung heutiger Rechner und der Verwendung von 64-bit-Webbrowser-Anwendungen stellen auch mehrere Gigabytes an Kacheldaten im Hauptspeicher des Webbrowsers kein Problem dar. Wenn alle Objekte der Queue abgearbeitet sind, geht der Thread in den Wartezustand über.

Nach jeder Veränderung des Betrachtungsbereichs durch den Benutzer (z. B. durch Zooming oder Panning) wird die Queue neu erstellt und der Worker-Thread aufgeweckt. Die Kacheln, die sich nicht mehr im aktuellen Betrachtungsbereich befinden, werden im Cesium-Viewer ausgeblendet. Mit diesem Workflow werden die Kacheln während der Navigation

sukzessive geladen und entladen. Wenn die maximale Anzahl gecacheter Kacheln überschritten wird, werden Kacheln aus dem Cache entfernt, wobei immer zuerst diejenige Kachel entfernt wird, die am längsten zurückliegend angezeigt wurde.

Für jeden Daten-Layer wird im 3D-Webclient ein eigener Worker-Thread erzeugt. Auf diese Weise werden mehrere CPU-Kerne gleichzeitig genutzt und das Daten- und Visualisierungsmanagement jedes Layers erfolgt unabhängig voneinander.

4 Anwendungsbeispiel: 3D-Stadtmodell von New York City

Im Jahr 2015 wurde im Rahmen eines Studienprojekts am Lehrstuhl für Geoinformatik an der TU München ein semantisches 3D-Stadtmodell der Stadt New York City (NYC) erstellt. Die Eingangsdaten zur Erzeugung des semantischen 3D-Stadtmodells entstammen dem NYC Open Data Portal, wobei insgesamt 26 Datensätze aus fünf verschiedenen Institutionen der Stadtverwaltung zu einem CityGML-Datensatz integriert wurden. Aus 2D- und 2,5D-Geodaten wurden unter Verwendung des ETL-Werkzeugs FME in Kombination mit photogrammetrischen Analysen die unterschiedlichen 3D-Stadtmodellobjekte vollautomatisch abgeleitet. Der CityGML-Datensatz umfasst alle Gebäude, Grundstücke, Straßen, Parks, Gewässer und das digitale Geländemodell von NYC in LOD1. Details zum Ablauf der Ableitung des semantischen 3D-Stadtmodells werden in (KOLBE et al. 2015, BURGER 2015, CANTZLER 2015) gegeben. Nach unserem Kenntnisstand ist das resultierende semantische 3D-Stadtmodell zum Zeitpunkt der Erstellung dieses Beitrags das erste öffentlich zugängliche semantische 3D-Stadtmodell einer Großstadt in den USA, das zudem noch auf amtlichen Geodaten basiert. Die CityGML-Datensätze stehen zum freien Download als Open Data auf der Homepage[1] des Lehrstuhls für Geoinformatik zur Verfügung.

In diesem Anwendungsbeispiel wird die 3D-Visualisierung von unterschiedlichen Objektarten des semantischen 3D-Stadtmodells von NYC im 3D-Webclient demonstriert. Dazu zählen z. B. die Gebäude (1.082.015 Objekte mit Adress-Informationen und 55 verschiedenen thematischen Attributen pro Objekt), Straßen (149.292 Objekte mit 24 thematischen Attributen pro Objekt) und Parzellen (866.853 Objekte mit 92 thematischen Attributen pro Objekt). Luftbilder und digitales Geländemodell können auch im 3D-Webclient gemeinsam dargestellt werden. Die geladene Daten-Layer werden in einer Auswahlliste (siehe Punkt 1 in Abb. 6) aufgelistet und können in der 3D-Kartenansicht vom Nutzer ein- und ausgeblendet werden. Zusätzlich können die Sachinformationen von selektierten Stadtmodell-Objekten abgefragt werden. Diese werden in der Tabellenansicht (Punkt 2) angezeigt. Die Sachinformationen werden im Clouddienst „Google Fusion Tables“ gespeichert, der eine große Kapazität zur Speicherung und Verwaltung der Sachdaten zur Verfügung stellt. In einer Statusanzeige (Punkt 3) wird die Anzahl der geladenen und der im Cache zwischengespeicherten Kacheln angezeigt. Diese kann jederzeit zur Kontrolle der CPU- und Arbeitsspeicher-Auslastung des 3D-Webclients dienen.

Der 3D-Webclient mit der Visualisierung des 3D-Stadtmodells von NYC kann über den unten angegebenen Hyperlink zugegriffen, frei verwendet und weitergegeben werden. Die Konfigurationseinstellungen (inkl. aller Layer-Angaben) werden übrigens komplett in der URL des Webclients codiert (siehe dazu auch KOLBE et al. 2003).

[1] https://www.gis.bgu.tum.de/projekte/new-york-city-3d/

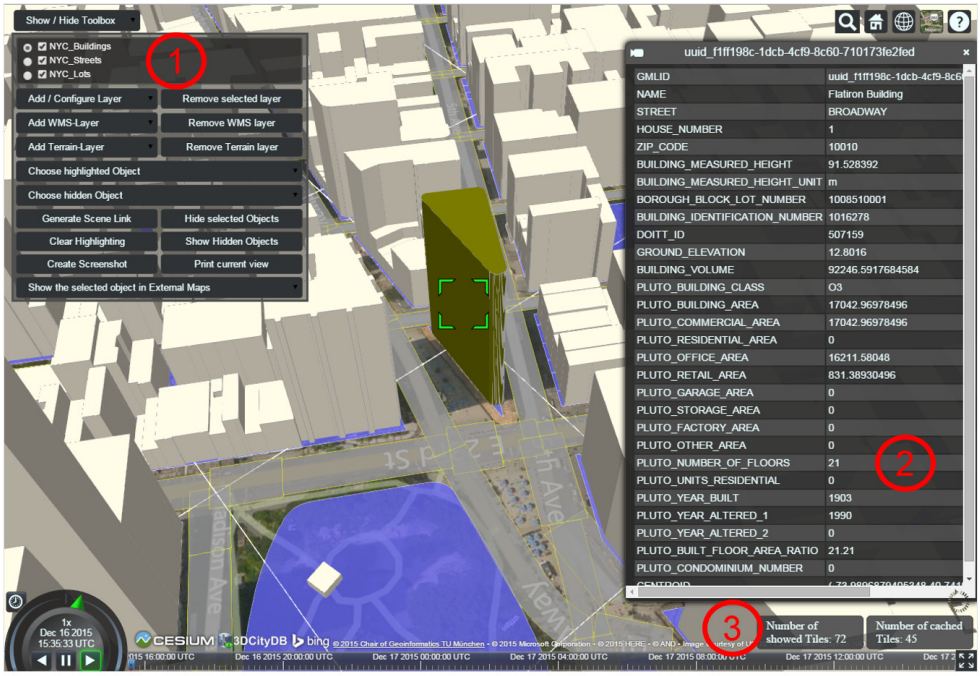

Abb. 6: Visualisierung des gesamten semantischen 3D-Stadtmodells der Großstadt New York City im 3D-Webclient.

5 Zusammenfassung und Ausblick

CityGML-basierende semantische 3D-Stadtmodelle nehmen heutzutage in zahlreichen Anwendungsbereichen einen immer wichtigeren Platz ein. Sie ermöglichen, die umfangreichen thematischen und räumlichen Daten unterschiedlicher Quellen in einen gemeinsamen Datenbestand zu integrieren. Damit die CityGML-basierenden Daten den Nutzern öffentlich zugänglich und intuitiv explorierbar gemacht werden können, ist der Einsatz der 3D-Web-GIS-Technologie im Vergleich zur traditionellen Desktop-Lösung notwendig. Wenn man jedoch die CityGML-basierenden Daten mit ihren großen Datenmengen im Webbrowser visualisieren möchte, stieß man bisher rasch an Grenzen der Leistungsfähigkeit der Webbrowser. Die Erfahrungen in vielen Anwendungsszenarien haben gezeigt, dass ein Ansatz entwickelt werden musste, der den Nutzern die Möglichkeit einer performanten und realitätsnahen 3D-Visualisierung der großen Stadtmodelle in modernen Webbrowsern gibt und gleichzeitig den Zugriff auf bzw. die Abfrage und das Bearbeiten von Sachinformationen (die semantischen Informationen) ermöglicht.

Dank der nunmehr weiten Verbreitung und Implementierungsunterstützung der WebGL- und HTML5-Technologien ist die Entwicklung einer 3D-Web-GIS-Anwendung zur Visualisierung von 3D-Stadtmodellen realisierbar. In diesem Beitrag wurde zunächst eine Systemarchitektur nach dem Client-Server-Konzept entwickelt, die die Geodaten unterschiedlicher Quellen und Formaten z. B. Satelliten- und Luftbilder, digitale Geländemodelle und 3D-Stadtmodelle auf einer einheitlichen Plattform integriert und in einem 3D-Webclient

visualisiert. Die Umsetzung des 3D-Webclients erfolgt durch eine spezifische Softwarearchitektur, die eine Erweiterung des Open Source Cesium-Frameworks um die performante Visualisierung von und Interaktion mit großen semantischen 3D-Stadtmodellen darstellt. Dazu wurde u. a. ein sogenannter Kachelmanager für den 3D-Webclient entwickelt, der durch kachelbasiertes Laden und Entladen sowie einem Caching-Mechanismus ein effektives Streaming des semantischen 3D-Stadtmodells mit enormen Datenmengen umsetzt. Dieser 3D-Webclient wurde erfolgreich für die Visualisierung des 3D-Stadtmodells von New York City verwendet und evaluiert. Er funktioniert auch (ohne Änderungen) auf mobilen Endgeräten wie dem Apple iPhone und iPad sowie auch Android-Geräten.

In Zukunft kann der 3D-Webclient konzeptuell und technisch um weitere Funktionen oder Module ergänzt sowie für weitere Anwendungsfälle eingesetzt werden. Hinsichtlich der 3D-Visualisierungskomponente basiert der 3D-Webclient derzeit zwingend auf dem Cesium-Viewer. Denkbar wäre zum Beispiel, dass neben der Abstraktion der räumlichen und thematischen Datenquellen eine weitere Schnittstelle (Interface) für die 3D-Visualisierungskomponente eingebracht wird, um die 3D-Visualisierungskomponente von spezifischen Produkten zu abstrahieren. So soll es künftig möglich sein, dass die 3D-Visualisierungskomponente des 3D-Webclients durch andere WebGL-basierte 3D-Webviewer z. B. OpenWebGLobe, WebGL Earth flexibel ausgetauscht werden kann. Weiterhin könnten die grafische Nutzeroberfläche und die Performanz mancher Funktionsblöcke des 3D-Webclients für mobile Geräte z. B. Smartphones verbessert werden, um den 3D-Webclient einem breiteren Nutzerkreis zur Verfügung zu stellen. Die aktuelle Version des 3D-Webclients ist Open Source und kann bereits unter GitHub[2] heruntergeladen werden. Dort sind auch schon ein paar Live-Demos aufrufbar. Der 3D-Webclient wird künftig ein integraler Bestandteil des 3DCityDB-Programmpakets sein.

Literatur

ANALYTICS GRAPHICS INC. (2015), Cesium – WebGL Virtual Globe and Map Engine. http://cesiumjs.org/index.html (20.11.2015).

BURGER, B. (2015), Ein semantisches 3D-Stadtmodell der Stadt New York auf Basis von Open Data – DGM, Straßen und Zonierung. Master Thesis, Chair of Geoinformatics, Technische Universität München.

CANTZLER, B. (2015), Ein semantisches 3D-Stadtmodell der Stadt New York auf Basis von Open Data – DGM, Gebäude, Flurstücke, Gewässer, Parks und Bäume. Master Thesis, Chair of Geoinformatics, Technische Universität München.

GAMMA, E., HELM, R., JOHNSON, R. & VLISSIDES J. (1995), Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley. ISBN 0-201-63361-2.

GONZALEZ, H., HALEVY, A., JENSEN, C., LANGEN, A., MADHAVAN, J. SHAPLEY, R. & SHEN W. (2010): Data management, integration and collaboration in the cloud. In: Proceedings of the International Symposium on Cloud Computing, 175-180.

GRÖGER, G., KOLBE, T. H., NAGEL, C. & HÄFELE, K.-H. (2012), OGC City Geography Markup Language (CityGML) Encoding Standard, v2.0. OGC Doc. No. 12-019. http://www.opengeospatial.org/standards/citygml (09.01.2014).

[2] https://github.com/3dcitydb/3dcitydb-web-map

HERRERUELA, J., NAGEL, C. & KOLBE, T. H. (2012), Value-added Services for 3D City Models using Cloud Computing. In: LÖWNER, M., HILLEN, F. & WOHLFAHRT, R. (Hrsg.), Geoinformatik 2012 „Mobilität und Umwelt“. Shaker Verlag, Aachen, 327-335.

KHRONOS (2015), glTF – Efficient, Interoperable Transmission of 3D Scenes and Models. https://www.khronos.org/gltf/ (22.11.2015).

KOLBE, T. H. (2009), Representing and Exchanging 3D City Models with CityGML. In: LEE, J. & ZLATANOVA, S. (Eds.), 3D Geo-Information Sciences. Springer, Berlin/Heidelberg, 15-31.

KOLBE, T. H., BURGER, B. & CANTZLER, B. (2015), CityGML goes to Broadway. In: FRITSCH, D. (Ed.), Photogrammetric Week '15. Wichmann, Berlin/Offenbach, 343-356.

KOLBE, T. H., STEINRÜCKEN, J. & PLÜMER L. (2003): Cooperative Public Web Maps. In: Proceedings of the International Cartographic Congress ICC, Durban, South Africa.

KOLBE, T. H., YAO, Z., NAGEL, C., KUNDE, F., WILLKOMM, P., HUDRA, G., MÜFTÜOGLU, A. & HERRERUELA, J. (2015), 3D City Database for CityGML, Version 3.0.0. http://www.3dcitydb.org. (09.05.2015).

LUBBERS, P., ALBERS, B. & SALIM, F. (2011), Pro HTML5 programming. Springer, Berlin/Heidelberg. ISBN: 978-1-4302-3864-5.

MARRIN, C. (2013), WebGL Specification. Khronos WebGL Working Group https://www.khronos.org/registry/webgl/specs/1.0/ (22.11.2015).

OPENWEBGLOBE (2013), OpenWebGlobe | WebGL geospatial data viewer. http://www.openwebglobe.org/ (22.11.2015).

STADLER, A., NAGEL, C., KÖNIG, G. & KOLBE, T. H. (2009), Making Interoperability Persistent: A 3D Geodatabase Based on CityGML. In: LEE, J. & ZLATANOVA, S. (Eds.), 3D Geo-Information Sciences, Springer, Berlin/Heidelberg,175-192.

TILE MAP SERVICE SPECIFICATION (2012), Tile Map Service Specification – OSGeo Wiki. http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification/ (22.11.2015).

W3C RECOMMENDATION (2014), HTML5. http://www.w3.org/TR/html5/ (10.11.2015).

WEBGL EARTH (2015), WebGL Earth – open source 3D digital globe written in JavaScript. http://www.webglearth.org/ (22.11.2015).

YAO, Z., SINDRAM, M., KADEN, R. & KOLBE, T. H. (2014), Cloud-basierter 3D-Webclient zur kollaborativen Planung energetischer Maßnahmen am Beispiel von Berlin und London, In: KOLBE, T. H., BILL, R. & DONAUBAUER, A. (Hrsg.), Geoinformationssysteme 2014. Wichmann, Berlin/Offenbach, 40-52.