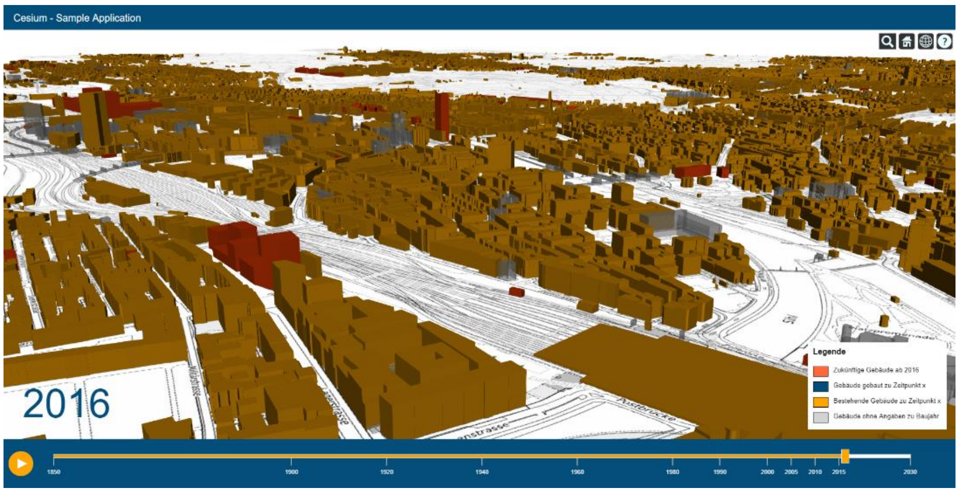
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

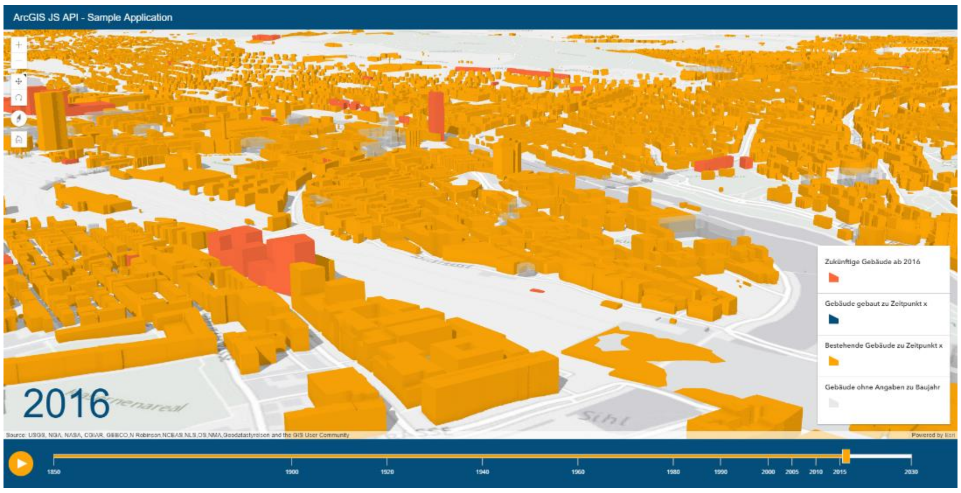
IKG
Institut für Kartografie und Geoinformation
Institute of Cartography and Geoinformation

Cartography Lab Autumn Semester 2017
Institute of Cartography and Geoinformation, ETH Zurich

Cesium vs. ArcGIS API for JavaScript

An assessment of the usability of virtual globe toolkits to create an animated 3D web scene


Cesium - Sample Application
Legende
Zukünftige Gebäude ab 2016
Gebäude gebaut zu Zeitpunkt x
Bestehende Gebäude zu Zeitpunkt x
Gebäude ohne Angaben zu Baujahr
2016
1850
1900
1920
1940
1960
1980
1990
2000
2005
2010
2015
2030


ArcGIS JS API - Sample Application
Zukünftige Gebäude ab 2016
Gebäude gebaut zu Zeitpunkt x
Bestehende Gebäude zu Zeitpunkt x
Gebäude ohne Angaben zu Baujahr
2016
1850
1900
1920
1940
1960
1980
1990
2000
2005
2010
2015
2030

Author: Lisa Stähli, MSc Geomatics Engineering
staehlli@student.ethz.ch

Examiner: Prof. Dr. Lorenz Hurni
Supervision: Dr. Christian Häberling, Raimund Schnürer

Zurich, 30.09.2017

Content

1. Introduction

A well-grounded and systematic comparison between 3D Web Mapping APIs that is also up-to-date is currently missing in the field of Web Cartography. This might be explained by the fast-paced development of JavaScript APIs in general and by the fact that the creation of 3D web maps requires a specific technical knowledge and in general a greater effort in comparison to creating 2D maps. The 3D representations of data on the web increasingly gain importance in Web Cartography and users feel the need to explore the available technologies including JavaScript APIs – both open source such as **Cesium** as well as closed source solutions such as the **ArcGIS API for JavaScript** that requires an Esri license to use all functionalities. There are other 3D Mapping APIs like WebGL Earth, that only has a small community, wrld.js (former eegeo.js) which is not yet technologically as advanced as other APIs, or the Google Earth API which has been shut down earlier this year. However, Cesium and ArcGIS API for JavaScript are considered to be the two most advanced and established 3D Web Mapping APIs currently available on the market.

2. Goals

This project aims to compare two different 3D Mapping Javascript APIs (Cesium and ArcGIS API for JavaScript). The two APIs are pursuing varying approaches of data modeling and processing, symbolization and visualization. Due to the fact that one shouldn't compare apples to oranges, a fair comparison between the two APIs can only be done by implementing identical use cases and oppose the resulting applications. Therefore, the comparison will be done by implementing two prototypes that are as similar as possible and that showcase a 3D animation of the settlement development of the city of Zurich from 1850 to 2030 (including future planned settlements).

The 3D scenes constructed in the prototypes are featuring a low-key base map as well as 3D buildings that are displayed and color-coded depending on their construction year. Therefore, the prototype includes a timeline slider with a play/stop button for the animation. The 3D scene needs to be fully interactive (spatial navigation enabled). Additionally, a legend decodes the building colors. The implementation of the prototypes targets the following common use cases:

- **Use Case 1**: Displaying and symbolizing a large dataset of 3D features
- **Use Case 2**: Add a customized base map (WMS) and elevation layer (terrain tiles)
- **Use Case 3**: Temporal animation of 3D features

The prototypes will be developed as HTML5 Web Apps that will at least work properly in Google Chrome. Eventually, the apps will be hosted on an ETH internal web server. Before constructing the prototypes, the APIs are also going to be compared to each other concerning general information (e.g. hardware requirements, application fields, supported data formats) as well as developer resources (e.g. API reference, tutorials, and samples).

The comparison addresses everyone in the field of 3D Web Cartography that is interested in finding out advantages and disadvantages of the two most common 3D Web Mapping APIs. Eventually, the comparison aims to support the decision-making for or against one of the toolkits. However, the comparison does not cover all aspects or functionalities of both APIs and therefore does not pursue to be complete.

3. Data

A. Basemap

The base map consisted of an elevation layer and a base map imagery layer. The elevation layer was provided by the City of Zurich as a digital terrain model (DTM) in a tif-format (see Appendix A). As a base map imagery layer, a Web Map Service (WMS) called “Uebersichtskarte_2016” was selected and accessed through the Open Data Portal of the City of Zurich. A grayscale base was chosen to not draw too much attention to it. Additionally, the publicly available “ESRI - World Light Gray Basemap” has been used as a base map due to technical issues with implementing a WMS layer in one of the prototypes.

B. Thematic Data

The thematic data consisting of building footprints (LOD 2.5) has been provided by the Amt für Städtebau, City of Zurich (see Appendix A). The footprints are coming with a height attribute that allowed for an extrusion to a 3D-object. Another attribute called “status” was used to determine existing buildings (status = “real”) and future upcoming buildings. For buildings with the status “erstellt” (built) a construction year of 2016 was assumed, for the status “freigegeben” (permitted) the construction year was forecasted to 2018 and for buildings with status “eingereicht” (submitted) the construction year was set to 2020.

Additionally, the City of Zurich provided a table with construction years of most (92%) of the buildings ranging from 1100 to 2015 that has been joined over the EGID (building id) attribute in ArcGIS Pro with the building footprint geometries.

4. Technology

A. Hardware

The development of the prototypes has been executed on a Laptop (Lenovo ideapad Y700) with the following hardware specifications:

Processor	i7-6700HQ CPU @ 2.60 GHz
OS	Windows 10
Graphics Card	NVIDIA GeForce GTX 960M (4GB)
Memory (RAM)	16 GB
Storage	1 TB

Table 1: Hardware Specification of Developer Laptop

B. Software

The following **client-side software** has been used to implement the prototypes:

- Cesium (1.37): 2D, 2.5D and 3D (virtual globe) Mapping API
- ArcGIS API for JavaScript (4.5): 2D and 3D (virtual globe, local scene) Mapping API

A list with general information aboug the two APIs can be found in Appendix B.

Additionally, the following JavaScript libraries have been used to create UI components:

- Bootstrap (+jQuery): HTML, CSS, and JS code designed to help build user interface components (e.g. buttons)
- noUISlider: a lightweight and freely available JavaScript range slider with additional functionalities (styling, interactivity) as opposed to the HTML5 input range
- wNumb: for number formatting, helper methods for the slider

For the **server-side software**, the following setup has been used for development:

The Web Apps can run on any web server (e.g. Apache, IIS). The Cesium prototype will run on a Node.js web server for development and the ArcGIS API for JavaScript prototype runs on an IIS (Microsoft Internet Information Services) web server for development. Additionally, to access the WMS-Layer, the HTTP request needs to go through a proxy. Therefore, a .Net proxy has been set up that is running on an IIS web server as well.

For **data processing**, the following software has been used:

- FME 2017.1: to generate 3D Tiles used in Cesium
- ArcGIS Pro 1.4.1: for joining and extruding footprints and generate Scene Layer Packages as well as Tile Packages that are uploaded to ArcGIS Online

5. Methods

The following explanations can be seen as a step-by-step guide on how the prototypes have been built as well as how the comparison of the developer resources has been done. Resources that have been used in the process of data preparation and implementation are included as hyperlinks directly in the explanations.

A. Layout and Design

The layout of the prototypes was not the biggest concern in this project, as the focus was on the implementation. Nevertheless, layout sketches have been made beforehand and a default layout has been implemented for both prototypes that featured a landing page that serves as an imprint and includes a general description of the project. From there, both prototypes can be accessed that have the same user interface featuring a header in dark blue with a title in white as well as a footer in dark blue that includes the timeline slider with year dates indicating the range held in white and orange. The footer also has a play/stop button in white and orange. The progress on the timeline slider was marked with filling the slider in orange whereas the rest of it stayed white. The main viewport has an overlay of the selected year date at the bottom left in dark blue (so it is visible against mostly orange buildings), a simple legend at the bottom right (see figure 11) and navigation tools (default placement). The final layout can be seen in figure 7-10 in Appendix D.

The colors of the buildings match the UI colors. Already built buildings are held in orange whereas the buildings of the selected construction year or shown in dark blue. Future buildings are displayed in a dark orange. The 3 colors have been selected as a color triad. Additionally, buildings with no construction year were colored in light grey and with 50% transparency.

Design	dark blue	orange	dark orange	light grey	white
color	🟦	🟧	🟥	⬜	⬜
hex	#034e7b	#FFA500	#FF6939	#d3d3d3	#ffffff

Table 2: UI color selection

B. Data Preparation

First, the data sets have been processed in ArcGIS Pro. The building footprints have been joined with the construction year table over attribute EGID (unique building id). With this 2D layer, the below explained steps needed to be performed for the two APIs in order to have data ready to be loaded in a web scene:

ArcGIS API for JavaScript	**Cesium**
In **ArcGIS Pro**, the 2D Layer of the building footprints need to be moved to the 3D Layers group in the contents panel (therefore a 2D map needs to be converted to a 3D map over View > Convert). There, the appearance of the feature layer can be adjusted and an extrusion based on an attribute (height) can be applied. As a next step, the 3D layer needs to be converted to a Multipatch feature class with the 3D Analyst tool “Layer 3D to Feature Class”. The 3D layer is exported as a **Scene Layer Package** (.slpk) from ArcGIS Pro with the data management tool “Create Scene Layer Package”. The slpk-file can then be uploaded as an item (from Computer) in the ArcGIS Online content panel. It will then be published as a	The 2D layer is imported to **FME** (Reader of type Esri GDB) and with a transformer of type extrusion the polygons will be extruded to the amount of the height attribute. Then, a Writer of type Cesium **3D Tiles** is attached to the transformer that will write the 3D Tiles to a predefined folder. After finishing the writing, the folder will contain a tileset.json with an object for each tile that describes a bounding box and points to the location of the data file. The data files are in the data folder and are in a b3dm-format (Cesium proprietary format). The tileset can be saved as a local folder and loaded with the path to that folder in the application.

hosted Scene Layer and accessible through the API with an URL pointing to *tiles.arcgis.com*.	
Three layers have been published to ArcGIS Online for the ArcGIS API for JavaScript app: 1. Buildings with construction_year = null (8%) 2. Buildings with defined construction year and STATUS = "real" (already built) 3. Buildings with STATUS = "erstellt" or "freigegeben" or "eingereicht" (just been built, in construction or in consideration for construction) This was done to make filter conditions and renderers easier in the application (explained below in chapter D).	For the Cesium app only one tileset was needed to be generated.

Apart from the building layers, also the DTM has been preprocessed. The steps are explained in chapter D2 as this belongs to a use case.

C. Set up for Web Apps

The index.html for both prototypes are created first with a default setup and then need to have the noUISlider and wNumb-references (files have been downloaded and are loaded with local path) in order to create the time slider. Next, the APIs need to be referenced as well.

For the set-up of the web applications a couple of things need to be considered for the APIs:

ArcGIS API for JavaScript	**Cesium**
The folder with the code for the app can be linked to any web server. This prototype was hosted on an IIS-webserver. To set-up an IIS web server, this Windows feature first needs to be activated. Then, IIS Manager can be started and a new website can be linked with the folder. The app is then running on localhost with the selected port.	The Cesium API (master branch) needs to be downloaded or cloned on GitHub. Additionally, Node.js needs to be installed on your computer. Then, with npm install, all dependencies are installed. Next, the node server (web server) can be started (node server.js) and the app will then be running on localhost with port 8080.
The ArcGIS API for JavaScript needs two references in the index.html: A stylesheet with the main.css and the API code itself. The stylesheet needs to be placed first in the head, then the slider APIs and finally your own CSS code. The API reference comes after the dojo config.	Cesium needs two references in the index.html that point to the local Build cesium-file (script tag) and which is on top of the header (before slider API references) and to the local Widget cesium-file (URL imported in script tag) which comes after the slider APIs and sets the viewport for the app.
As the ArcGIS API for JavaScript comes along with the Dojo Framework, the HTML5 app also features a Dojo Module Loader as well as a config file. Therefore, the app.js (prototype code) is loaded with "require" in the head.	The app.js (prototype code) is linked with a script tag at the end of the body.
Apart from the UI components, the body contains a div for the view of the application.	Apart from the UI components, the body contains a div for the cesium container.

D. Implementation of Use Cases

First, the general application is set-up by creating the default scene within the UI component. The following table explains the default set-up of app.js:

ArcGIS API for JavaScript	Cesium
An initial scene view is created by first loading the required modules with dojo-define in the app.js. The modules to get started are "esri/Map", "esri/views/SceneView". First, the map is created with a base map and a ground (e.g. world-elevation), then the scene view is constructed that is linked to the view div (DOM Node) and the map from above. Navigation tools are default UI components. Additionally, the home button (widget) was added to the UI. Each layer needs to be individually popup enabled or disabled to get rid of pop-up windows.	The initial viewer can be created in Cesium with 1 line of code and with the viewer constructor that is linked to the cesium container div (DOM Node). However, as the UI of the prototype was required to be as clean as possible, all default UI components in the viewer have been disabled, including timeline, animation, baseLayerPicker, selectionIndicator and infoBox (corresponds to pop-up window).

D1. Use Case 1: Displaying and symbolizing a large dataset of 3D features

ArcGIS API for JavaScript	Cesium
The 3D datasets are loaded as SceneLayers to the application with an URL to the ArcGIS Online resource.	The tileset is loaded as a Cesium3DTileset to the application with a link to the local tileset top-folder (not data folder).
The scene layer is symbolized and styled with properties (e.g. opacity) as well as renderers. Renderers are applied directly to the layer and contain drawing information. In this prototype, each layer has a SimpleRenderer that features a symbol of type MeshSymbol3D. MeshSymbol3D is used to render 3D mesh features in a SceneLayer and has a symbol layer of type FillSymbol3DLayer attached to it. The symbol layer defines properties like material color. Colors can be rgb, rgba, hex or string values.	The styling of the tileset can be done with the Cesium3DTileStyle that inherits properties like color and visibility (show). The expression can include rgb-color, rgba-color, hex or string values.
For advanced and customized symbolization, other renderers such as UniqueValueRenderer which assigns colors based on attributes or ClassBreaksRenderer can be used. With visualVariables numeric attributes can be visualized with start and stop values and linear color ramps.	For advanced and customized symbolization, the Cesium3DTileStyle can evaluate expressions defined using the 3D Tiles Styling language. The color property can be defined based on an attribute value. However, all attributes and corresponding colors need to be assigned manually.

D2. Use Case 2: Add a customized base map and elevation layer

ArcGIS API for JavaScript	Cesium
For displaying the elevation layer, the DTM first needs to be published as a hosted elevation layer on ArcGIS Online for which a tile package needs to be created in ArcGIS Pro. The tile package can be added as an item in ArcGIS Online and then loaded into the application as an ElevationLayer. The elevation layer then needs to be added to the ground layers of the scene. The layer will seamlessly fade into the existing elevation model around it.	For displaying a terrain in Cesium, the cesium-terrain-builder can be used, a C++ library and associated command line tools designed to create terrain tiles for use in Cesium. To prevent buildings vanishing below the terrain, the terrain provider needed to be switched off completely in the prototype and all buildings were at the same height. Additionally, the height of the whole tileset had to be adjusted by -48m because

	the buildings would otherwise float over the surface.
For displaying a customized base map from a WMS, it can be loaded as a WMSLayer in the ArcGIS API for JavaScript. The layer can have sublayers with which you can access a specific layer in a WMS. It can be added directly to the scene. To request a WMS, a proxy needs to be set up for the application.	For displaying an imagery layer from a WMS in Cesium, the WebMapServiceImageryProvider can be used. This provider takes properties such as the URL of the WMS, parameters (e.g. format), layers and the proxy.
Esri provides files with which a proxy (.NET, Java, PHP) can be set up. For the prototype, a .NET proxy was set up as described in the README file (e.g. convert the folder to an application in the IIS to start proxy). Even though the proxy was running and could be used from another application, the set-up with the Dojo Module Loader caused problems with the WMS and therefore would not show in the SceneView.	Cesium provides a default proxy that can be used in development mode with the node web server.

D3. Use Case 3: Temporal and non-temporal animation of 3D features

For both prototypes, the animation of the features has been linked to the timeline slider which was created with the noUISlider API. The timeline is listening to both inputs from users dragging the handle to a certain year and the play/stop button. The play button lets the timeline start moving for the currently selected year and the stop button stops the animation. Both interactions trigger a function called *timelineAnimation* in the code which starts the animation of the buildings. The play button additionally initializes an interval with a 2-second loop.

The temporal part of the animation was to make sure that the correct buildings are displayed in the viewport for the selected year, which means the already constructed buildings are shown as well as the ones built in the exact selected year. For the non-temporal part, the buildings needed to be animated when appearing for the selected year. This animation should be a height animation, so it should look like the buildings are growing from the ground up to their actual height. The following explanations describe how those two parts were implemented differently with the APIs:

ArcGIS API for JavaScript	**Cesium**
For temporal animation, the ArcGIS API for JavaScript provides so-called definitionExpressions on layers which can be used to filter the layers. The definition expression on the layer can take an SQL expression to query the REST endpoint in ArcGIS Online, e.g. “status = ‘real’”. The first attempt to make the animation work was using a FeatureLayer with only building footprints instead of a SceneLayer. Unfortunately, the definition expression caused the FeatureLayer to always reload completely when a new year was selected or jumped to in the animation. As this has a negative impact on the user experience, the FeatureLayer has been replaced with a SceneLayer.	For temporal animation, the Cesium tileset could be filtered with the show property of the Cesium3DTileStyle that can take conditions based on attributes. It is here important to always apply style changes as a new tile style (new Cesium3DTileStyle) that also still has all color properties from before, otherwise, the tileset will not reload properly.

For the non-temporal animation, it was initially the idea to apply an ExtrudeSymbol3DLayer to the layer with continuously changing height values (size). Unfortunately, this symbol layer does not support 3D symbols and feature layers could not be used, so the second part of the animation could not be implemented.	For the non-temporal animation, the tileset was animated by shifting all tiles of a second tileset that only showed the tiles from the exact selected year. Therefore, another interval was set with a 20ms loop which triggers the tileset height to change. The shifting is done with a translation on the model matrix of the tileset and was based on the Adjust Height sample.

E. Implementation of additional features

ArcGIS API for JavaScript	Cesium
A dynamic legend can be added with the Legend Widget where the layers, that should appear in the legend, can be defined with their respective layer names.	As there is no legend widget in Cesium, a legend had to be created manually.
To change the initial position of the view, the method goTo can be used on the view that can take various inputs, e.g. position (x, y, z, latitude, longitude), heading and tilt.	To change the initial position of the camera, in Cesium the camera can be triggered with a flyTo method that takes destination (WGS84-coordinates) and orientation (heading, pitch, roll) as inputs.
For a better quality of the visualization, the SceneView can draw scenes in a high quality by adding the property qualityProfile with value “high” to the view. Additionally, ambient occlusion and direct shadows can be enabled over the views environment property.	In Cesium, the visualization of the sky and the lighting can be adjusted with the skyAtmosphere property of the viewer.

F. Comparison of Developer Resources

Resources that are available for developers are crucial especially for users that have just started using an API. The more resources that are available and the higher their quality, the more likely a user will find a solution to a problem. Additionally, a well-documented API causes user to keep using it for future projects.

The following developer resources could be found online (status: 18.06.17) for the two APIs:

ArcGIS API for JavaScript	Cesium
API Reference	API Documentation
Samples	Sandcastle/Samples
Guide/Tutorials	Tutorials
GeoNet-Community Forum	Cesium Community Forum
Blog	Blog

Table 3: Available developer resources for both APIs

For further investigation, the API Reference/Documentation, as well as samples and tutorials as the most important resources, have been analyzed and categorized to see the amount of available information as well as which topics are covered more than others. Samples and tutorials are used as resources for developers but are also an indicator of what wants to be emphasized and explained to future developers and what was increasingly asked for from the existing community.

Both the capabilities from the API Reference/Documentation and the samples and tutorials have been categorized in the following 16 categories:

Nr.	Category	Description
1	Overview	A broad overview of all functionalities, applications, etc.
2	Getting started	How to use the API (embedding and installation instructions)
3	Programming	General information about programming patterns used
4	Interoperability	Instructions on how to combine with other APIs, frameworks, etc.
5	Data Management	Instructions on how to convert or use different data formats
6	Layout	Styling, CSS, User Interface, GUI Elements
7	Widgets	Ready-to-use or customized widgets
8	Visualization	How to visualize, display and transform data (renderers, symbols)
9	Navigation	How to use views, camera, and key inputs to manipulate navigation
10	Geometry	How to draw geometries
11	Editing	How to manipulate geometries
12	Geoprocessing	Calculate spatial components/patterns, apply changes
13	Interaction	How to interact with the displayed data, e.g. popup/info window content
14	Debugging	Tools to help to debug an application
15	Query	Filter and query data
16	Projection	Instructions of how to project data from one spatial reference to another

Table 4: Categories for comparison of API Reference capabilities and samples/tutorials

The capabilities can be classes, methods or properties that are documented and described in the API reference. What has been counted as a capability, are individual web pages in the API Reference that are dedicated to a specific class or method. The analysis consisted of summing up how many pages are documented for each of the categories which have developed organically while going through the pages and samples/tutorials. With this approach, the API Reference of the ArcGIS API for JavaScript has been compared to the Cesium API Documentation and the guide and samples for the ArcGIS API for JavaScript have been compared to the Cesium Sandcastle examples.

The following chart shows an overview of all the developer resources that have been analyzed. It depicts that there are more samples and tutorials available for the ArcGIS API for JavaScript, but almost double as many API Reference pages available for the Cesium API.

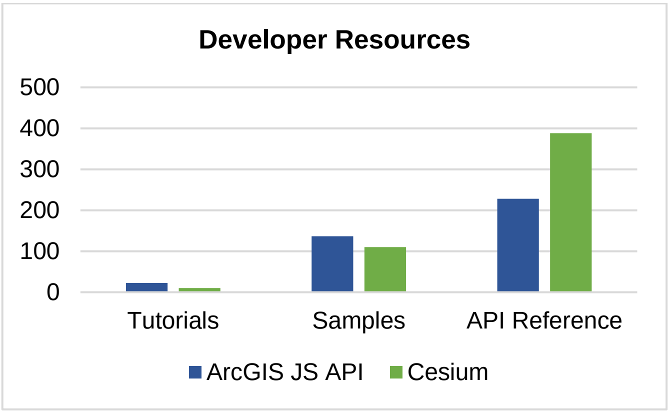


Figure 1: Number of available developer resources for both 3D Mapping APIs

The analysis of the API Reference shows that both APIs lay a big focus on data management and visualization whereas Cesium has more resources available about interaction, geometry as well as general programming/debugging topic. The ArcGIS API for JavaScript, on the other hand, has a bigger emphasis on geoprocessing and widgets. Also, proportionally there are more resources available for visualization topics in this API.

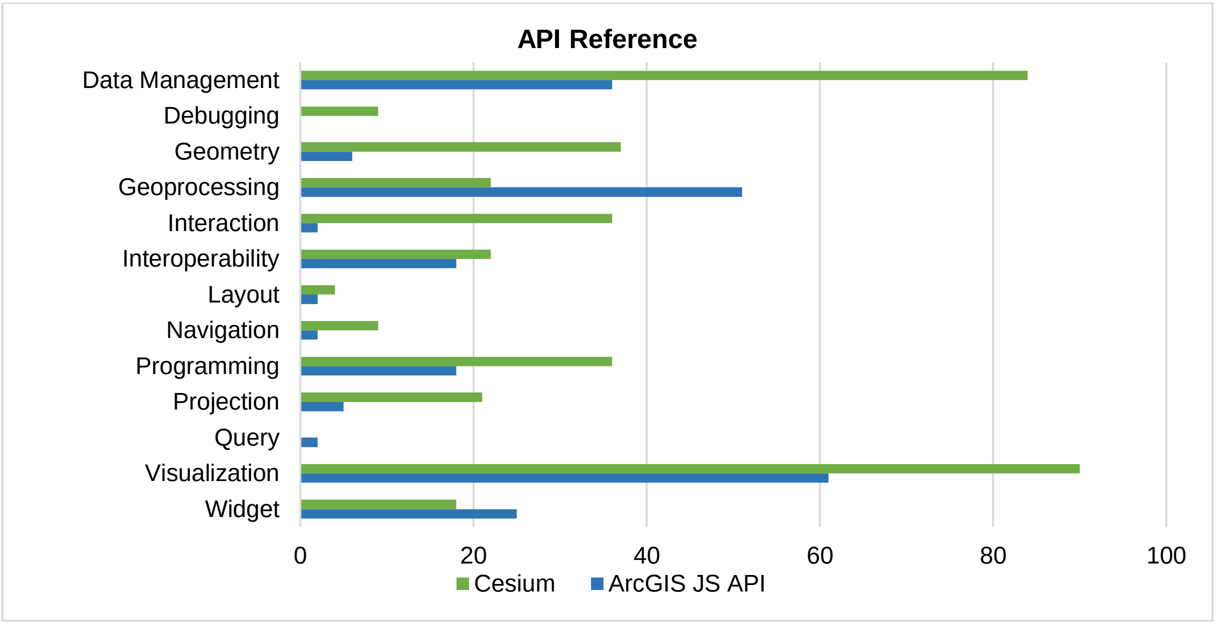


Figure 2: Comparison between categories of API Reference capabilities

The Cesium API has a lot of samples and tutorials available in the category interoperability which can be explained by many Google Earth API migration tutorials. The ArcGIS API for JavaScript, on the other hand, has a focus on widgets, visualization and data management with many tutorials about how to load data from ArcGIS Online or Portal and on how to style layers.

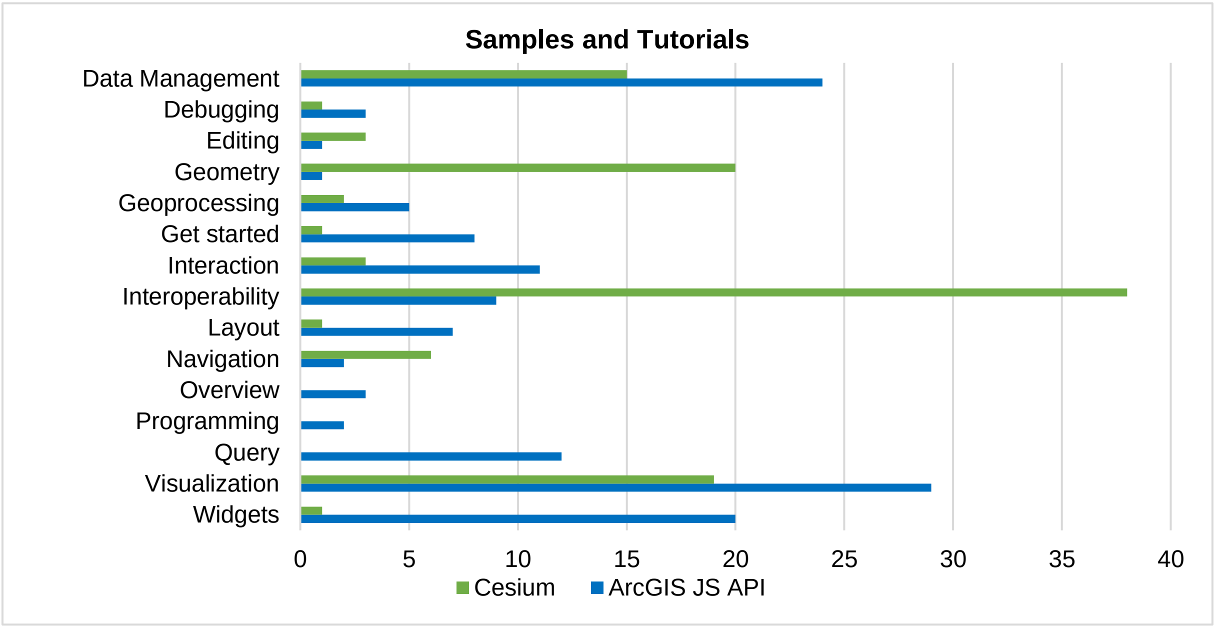


Figure 3: Comparison between categories of samples and tutorials

6. Results

The following chapter provides an analysis and summary of the implementation steps with a comparison of general approaches of the two APIs. The structure of this chapter follows the categories in the process of designing 3D web maps according to [3].

A. Data modeling and processing

Both APIs have developed their individual standards for 3D formats. Cesium uses 3D-Tiles whereas the ArcGIS API for JavaScript uses the Esri i3s standard. Both standards have been going through an OGC Community Standard Process [4] and i3s (Indexed 3D Scene Layers) has just recently been approved as an official OGC Standard [5] which also includes the Scene Layer Package that has been used in the prototypes. From a top-level perspective, the two standards do not differ much. They both are based on a tree-like structure (bounding volume tree hierarchy) of retrieving tiles (3DTiles) or nodes (i3s) with relevant data. They both support 3D objects, points and meshes (quantized mesh in 3D-Tiles and integrated mesh in i3s). Differences can be found in indexing, styling, and possibility of translation of tiles.

The main difference between the two APIs is how they handle the loading of 3D data in the web page. Cesium typically loads the local 3D-tiles (even though they can also be loaded from an external cloud, e.g. Amazon Cloud) and saves them in the cache and streams only visible tiles and those tiles which are most important for a given 3D view. The ArcGIS API for JavaScript, on the other hand, hosts the data on the cloud and streams it dynamically based on what is currently visible in the view port. The streaming and loading in the scene are faster for first loading with the ArcGIS API for JavaScript (but highly depended on the quality of the internet connection), for navigation, after the first loading, it is faster for the Cesium Application. One big advantage of the 3D-Tiles is the possibility to transform tiles and move tilesets which came in handy for the animation of buildings in the prototype.

Another difference can be found in the way different zoom levels are represented. Whereas zooming out in the Cesium 3D-Tiles does not cause the geometries to change and immediate and fast zooming in does not result in generalizations of the features (e.g. aggregate vertices), in the ArcGIS API for JavaScript application these “loading errors” can occur (see figure 4).

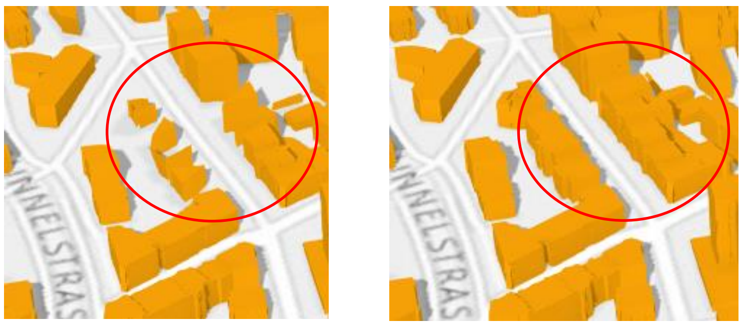


Figure 4: Features before and after properly loaded after zooming in and out

In the Cesium application, on the other hand, the tileset disappears completely at a certain zoom level which of course also occurs in the ArcGIS API for JavaScript application but it takes a higher zoom level (probably when all vertices coincide into one). It seems, this problem is handled differently in the two APIs.

B. Symbolization

The concept of renderers and symbols is a very particular one for the ArcGIS API for JavaScript. The so-called Smart Mapping paradigm found its way into almost all Esri platforms and is widely used and appreciated by users. It supports the design and styling of maps especially for users that do not have a cartographic, data science or design background or that just need to make quick visualization or even simply want to explore their data sets.

However, compared to the simplistic Cesium approach of condition-based styling with the 3DTileStyle, the renderers and symbols seem almost a bit too complicated. For users that are new to the ArcGIS API for JavaScript, the variety of renderers, symbols and symbol layers can be quite confusing and if the wrong symbol or symbol layer is applied to a layer, no visualization will be shown at all. Additionally, there is not much guidance coming from the console if wrong symbols are used. An error message indicating that a symbol was used that is not supported by the layer type would be useful.

On the other hand, the ArcGIS API for JavaScript provides a lot of different data visualization options and makes advanced styling fairly easy. Especially data-driven continuous color visualizations are easy to create with a few steps which would require a lot more thinking and preparing from the user for the Cesium approach.

C. Visualization

A general comparison of the visualization results has been made between the two screenshots in figure 5. For the building symbolization, exactly the same colors have been used. The first impression is, that the Cesium Application looks much darker. First, this might be explained by the fact that the lighting in the Cesium web globe is based on the time of the day. However, the picture has been captured at various times of the day (including noon) and the darkness or color of the buildings did not change at all. What did change is the color of the sky. It might be that 3D Tiles would need an additional pre-defined property about the material and how light is reflected. Also, shadows are not cast because there is no terrain provider.

Also, the camera position and orientation seems to be a bit different on this screenshot. This can be explained by either the fact that the Cesium App does not have a terrain or by different camera models (e.g. field-of-view).

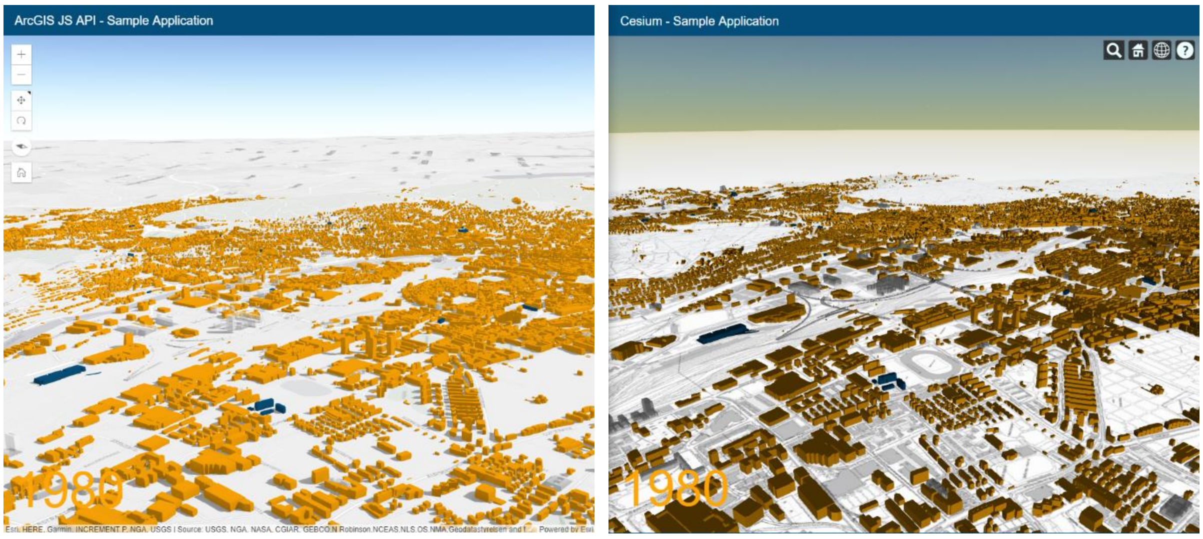

ArcGIS JS API - Sample Application
1980
Cesium - Sample Application
1980

Figure 5: Comparison of rendering results of similar viewing points (captured at 5 pm)

The ArcGIS API for JavaScript offers more options for scene rendering whereas Cesium provides many options to change the appearance of the sky. This can probably be explained by the fact that Cesium originally evolved from the space industry where atmospheric parameters are considered to be important to be changed.

7. Conclusion

The comparison of the APIs was challenging but worth investigating. Both APIs provide a lot of options and most of their approaches concerning implementation are very different. Therefore, it makes sense that users are not willing to switch back and forth on the two depending on their use case but that they stick with either one of them after learning it.

The following table is a summary of this study investigating both APIs for the specific use cases and sum up the author's experience working with the APIs:

	ArcGIS API for JavaScript	Cesium
Highlights	Beautiful visualizations/rendering Fast streaming and filtering Great resources (easy, appealing) Data hosting on the cloud	Open Source Fast streaming of 3D-Tiles (in cache) Big and active community Tileset can be transformed Easy styling and filtering of 3D-Tiles
Lowlights	Data hosting on the cloud (not suitable for sensitive data) Moving of features not possible	Resources partly very technical Visualizations could be more sophisticated

Table 5: Highlights and lowlights of the study comparing both APIs

As a very general conclusion, deciding on one or the other API depends highly on the user preferences and style. A pure developer will probably rather go with Cesium as the resources are not a challenge and open source is more accessible. A cartographer (or designer) might want to rather go with the ArcGIS API for JavaScript as visualizations look better and there are a lot of common cartography tools and widgets already built in and easy to create (e.g. Legend Widget, UniqueValueRenderer).

To sum up, both APIs are very powerful and in constant development and therefore suitable to build all sorts of 3D web scenes. Hopefully, the will influence each other in a positive way in the future by implementing the most promising features and methods on both platforms.

8. Bibliography

Most of the references have been made directly in the text as hyperlinks due to simplicity.

[1] ArcGIS API for JavaScript Discover 4.0 the Next Generation, Presentation at Esri Developer Summit 2016.
Link: http://www.esri.com/videos/watch?videoid=5024&isLegacy=true&title=arcgis-api-for-javascript-discover-40-the-next-generation (17.06.2017)

[2] About Cesium. Link: https://cesiumjs.org/about.html (17.06.2017)

[3] Häberling, Christian (2003). *Topografische 3D-Karten: Thesen für kartografische Gestaltungsgrundsätze*. Doctoral Thesis.

[4] The OGC Community Standard Process (15.02.2017). OGC Blog.
Link: http://www.opengeospatial.org/blog/2543 (28.09.2017)

[5] OGC approves Community Standard for streaming 3D Content (06.09.2017). Press Releases.
Link: http://www.opengeospatial.org/pressroom/pressreleases/2639 (28.09.2017)

9. Appendix

A. Terms of data contract with Amt für Städtebau, city of Zurich (German)

Im Zusammenhang mit dem Projekt Kartographie-Lab (Studentenprojekt) „Cesium vs. ArcGIS JS API“ benötigt der Nutzer die in diesem Vertrag aufgeführten Geodaten. Bezüglich der Nutzung der Geodaten vereinbaren die Parteien folgendes:

1. Die im Datenverzeichnis aufgeführten Geodaten (vgl. Anhang) sind Gegenstand dieses Vertrages und werden an den Nutzer in elektronischer Form unentgeltlich abgegeben.

2. Die abgegebenen Daten werden für folgendes Projekt genutzt:

Kartographie-Lab (Studentenprojekt) „Cesium vs. ArcGIS JS API“

3. Die zur Verfügung gestellten Geodaten dürfen ausschliesslich für das oben erwähnte Vorhaben genutzt werden; sie dürfen auf Datenträgern des Nutzers verwaltet, genutzt, ausgewertet, verändert, kopiert und gelöscht werden.

4. Das Recht zur Nutzung der erwähnten Geodaten erlischt nach Abschluss der erwähnten Arbeiten.

5. Nach Abschluss der Arbeiten müssen alle zur Verfügung gestellten wie auch davon abgeleiteten Geodaten von allen Datenträgern des Nutzers gelöscht werden. Die zur Verfügung gestellten Geodaten sind vollständig an den Datengeber zurückzugeben.

6. Weitergehende oder von diesem Vertrag abweichende Nutzungen der Geodaten müssen zuvor mit dem Datengeber schriftlich vereinbart werden.

7. Der Nutzer geniesst keine Ausschliesslichkeit für die Nutzung der Daten.

8. Der Dateninhalt und die Datenstruktur können vom Datengeber nach Bedürfnis und ohne vorherige Rücksprache mit dem jeweiligen Nutzer geändert werden.

9. Sämtliche Rechte an den abgegebenen Geodaten verbleiben beim Datengeber.

10. Alle abgegebenen Geodaten sowie daraus abgeleitete Daten sind vertraulich zu behandeln. Diese Daten dürfen ohne Genehmigung des Datengebers weder Dritten zugänglich gemacht noch weitergegeben werden. Jede Veröffentlichung (analog oder digital) setzt das schriftliche Einverständnis des Datengebers voraus.

11. Der Nutzer bestätigt mit seiner Unterschrift, dass ihm bekannt ist, dass er im Rahmen der Nutzung der abgegebenen Geodaten die eidg. und kantonalen Vorschriften zum Datenschutz zu beachten hat.

12. Der Datengeber gewährleistet die Gültigkeit der unveränderten Geodaten für den Ausgabetag, der auf dem Datenträger angegeben ist. Jeder Eingriff in den Datensatz hebt die Gewährleistung für den ganzen Datensatz auf. Abgegebene Datensätze können nicht nachgeführt werden. Datenaktualisierungen bedingen einen Neubezug aller Geodaten.

13. Der Nutzer bestätigt mit seiner Unterschrift, dass er darauf hingewiesen wurde, dass eine diesem Vertrag oder den gesetzlichen Vorschriften zuwiderlaufende Nutzung der Geodaten strafrechtliche Folgen haben kann.

14. Die Parteien einigen sich darauf, dass auf das Vertragsverhältnis nur schweizerisches Recht anwendbar ist.

15. Ausschliesslicher Gerichtsstand ist Zürich.

16. Dieser Vertrag wird 2-fach ausgefertigt (1-mal für Datengeber; 1-mal für Nutzer).

B. General Comparison

The following table shows a general comparison between the two APIs based on the information that is available on the web.

	ArcGIS API for JavaScript 4.4	**Cesium 1.32**
Project Start	2013 [1]	2011 [2]
Founder	Esri Inc	Analytical Graphics, Inc (AGI)
Operator	Esri Inc	AGI, Open Source Community
Description	"JavaScript API that integrates 2D and 3D into a single, easy-to-use, powerful API"	"an open-source JavaScript library for world-class 3D globes and maps"
Supported Browsers	Chrome Firefox Edge IE11* (no WebGL) Safari 9 and later	Chrome Firefox Edge IE 11 Safari
Hardware Requirements	2 GB system memory and a video graphics card that supports WebGL, nonmobile graphics card with at least 512 MB of video memory	WebGL support
How to get started for development	1. Include CDN in your HTML 2. Set up a web server (any) 3. Open website (e.g. localhost) 4. Start modifying your code	1. Download Cesium (zip-File) 2. Set up web server (Node.js) - Install Node.js - npm install - start node server.js 3. Open localhost:8080 (default) 4. Start modifying the available code (HTML, CSS, JS)
Data Processing	Convert and upload your datasets to ArcGIS Online or ArcGIS Portal by either publishing with ArcGIS Pro or uploading a zipped FileGDB or Scene Layer Package (generated in ArcGIS Pro)	Convert to CZML with a QGIS Plugin or convert to 3DTiles with FME/other plugins
Most common Application Fields	Urban Planning, Education and Data Exploration (Story Maps)	Space Geodesy, Aviation, Photogrammetry, Sports Tracking, Real-Time Data, Real Estate Management, Education, and Exploration, Urban Planning (see Appendix C)
Support for	Web, (Mobile, Tablets)	Web, Mobile, VR, Tablets
Proprietary Data Formats	Esri FileGDB, Shape, Scene Layer Package (i3s)	CZML, quantized-mesh, 3DTiles
Supported external Data Formats, Standards, Libraries and Frameworks	in general, various data formats but need to be converted in ArcMap/ ArcGIS Pro beforehand, additionally within the API: GeoJSON, OpenStreetMap	glTF, GeoJSON, TopoJSON, Collada, ArcGIS ImageLayer, WMS, TMS, WMTS, OpenStreetMap, Bing Maps, ArcGIS MapServer, Google Earth Enterprise (KML)
Supported Libraries and Frameworks	React, React-Redux, Riot, Angular 1, Vue and a plugin for Ember	Leaflet, OpenLayers

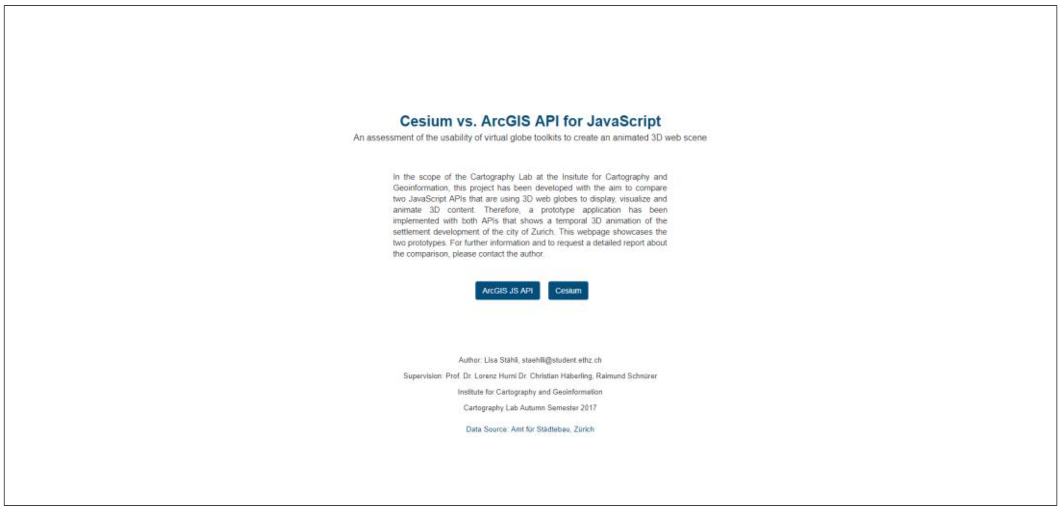
Table 6: General information of the two APIs compared to each other

C. Application/Demos for Cesium

List of Applications/Demos developed in Cesium (to extract most important application fields)

- High-Resolution 3D Models with photorealistic textures
- Satellite Tracking Measurements and Imagery Availability
- Exploring Mars
- Swiss Federal Geoportal (spatial infrastructure data)
- Real-time flight tracking
- Drone surveys and inspections
- Advanced data analytics
- Orbit Logic Collection Planning, Flight Planner for Satellites, UAVs, Radar, etc.
- Real-Time Data Visualization (e.g. Twitter Streams, Weather Forecasts)
- LiDAR Point Clouds (billions of points)
- Visualizations of natural phenomena (earthquakes, hurricanes, storms, weather, groundwater well)
- Crowdsensing (e.g. reporting wildfires)
- Paragliding / Skydiving Logbook and other Sports Analytics (sailing races, downhill skiing, GPS Tracks, cycling)
- Waterway Information
- High-resolution streaming “smart” 3D Buildings (CyberCity 3D)
- OpenWebGIS (calculate and analyze data)
- Planning and Management (e.g. forestry, airspace planning, tactical scenarios)
- 3D Virtual Drone Flights (with Google Cardboard and Oculus Rift)
- Education and Exploration (earth science, geography, maps, historical data)
- Car Navigation
- Real Estate Management, Office Space Search (Cube Cities)
- Solar Radiation Potential

D. A selection of images from the prototypes


Cesium vs. ArcGIS API for JavaScript

An assessment of the usability of virtual globe toolkits to create an animated 3D web scene

In the scope of the Cartography Lab at the Insitute for Cartography and Geoinformation, this project has been developed with the aim to compare two JavaScript APIs that are using 3D web globes to display, visualize and animate 3D content. Therefore, a prototype application has been implemented with both APIs that shows a temporal 3D animation of the settlement development of the city of Zurich. This webpage showcases the two prototypes. For further information and to request a detailed report about the comparison, please contact the author.

Author: Lisa Stähli, staehlli@student.ethz.ch

Supervision: Prof. Dr. Lorenz Hurni Dr. Christian Häberling, Raimund Schnürer

Institute for Cartography and Geoinformation

Cartography Lab Autumn Semester 2017

Data Source: Amt für Städtebau, Zürich

Figure 6: Landing page with imprint and project description

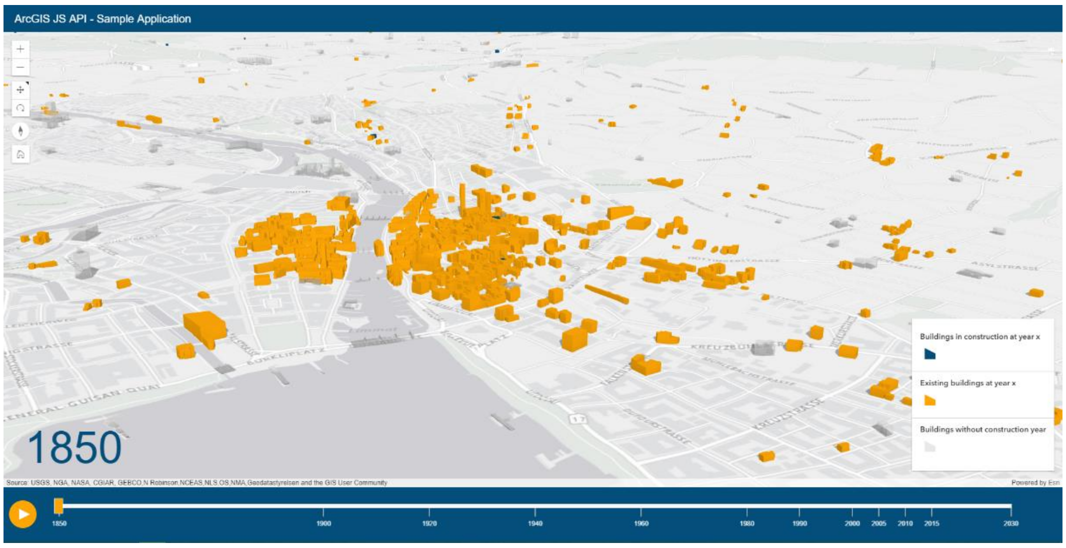

ArcGIS JS API - Sample Application
Buildings in construction at year x
Existing buildings at year x
Buildings without construction year
1850
Source: USGS, NGA, NASA, CGIAR, GEBCO,N Robinson,NCEAS,NLS,OS,NMA,Geodatastyrelsen and the GIS User Community
Powered by Esri
1850
1900
1920
1940
1960
1980
1990
2000
2005
2010
2015
2030

Figure 7: Start page of ArcGIS API for JavaScript application

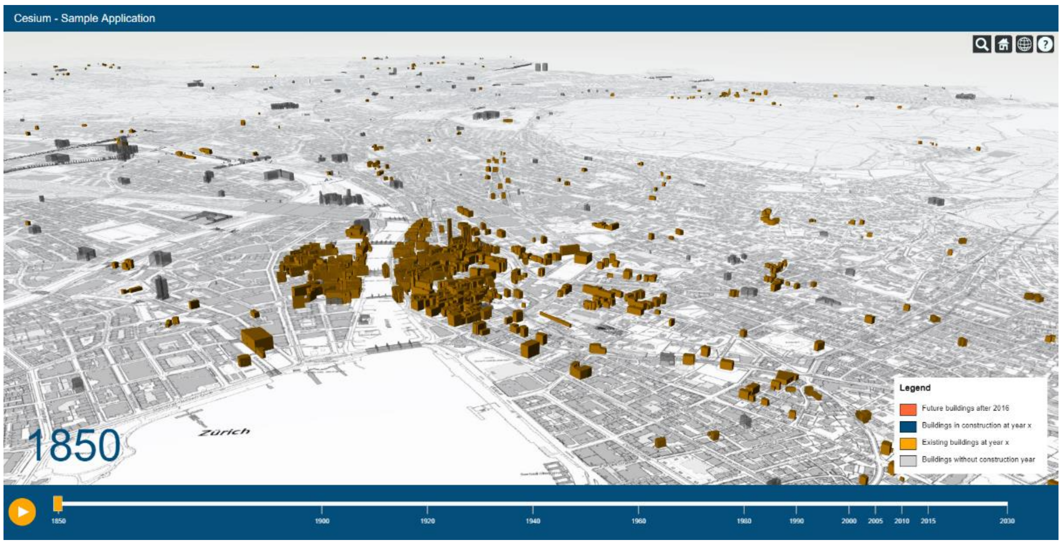

Cesium - Sample Application
Legend
Future buildings after 2016
Buildings in construction at year x
Existing buildings at year x
Buildings without construction year
1850
Zürich
1850
1900
1920
1940
1960
1980
1990
2000
2005
2010
2015
2030

Figure 8: Start page of Cesium application

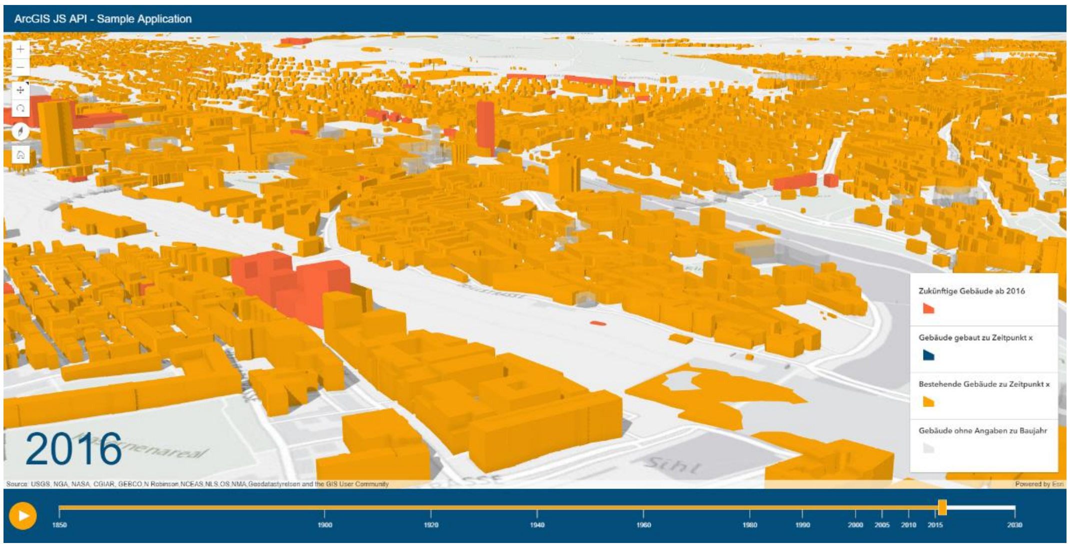

ArcGIS JS API - Sample Application
Zukünftige Gebäude ab 2016
Gebäude gebaut zu Zeitpunkt x
Bestehende Gebäude zu Zeitpunkt x
Gebäude ohne Angaben zu Baujahr
2016
Powered by Esri
1850
1900
1920
1940
1960
1980
1990
2000
2005
2010
2015
2030

Figure 9: ArcGIS API for JavaScript application with future buildings (on title page)

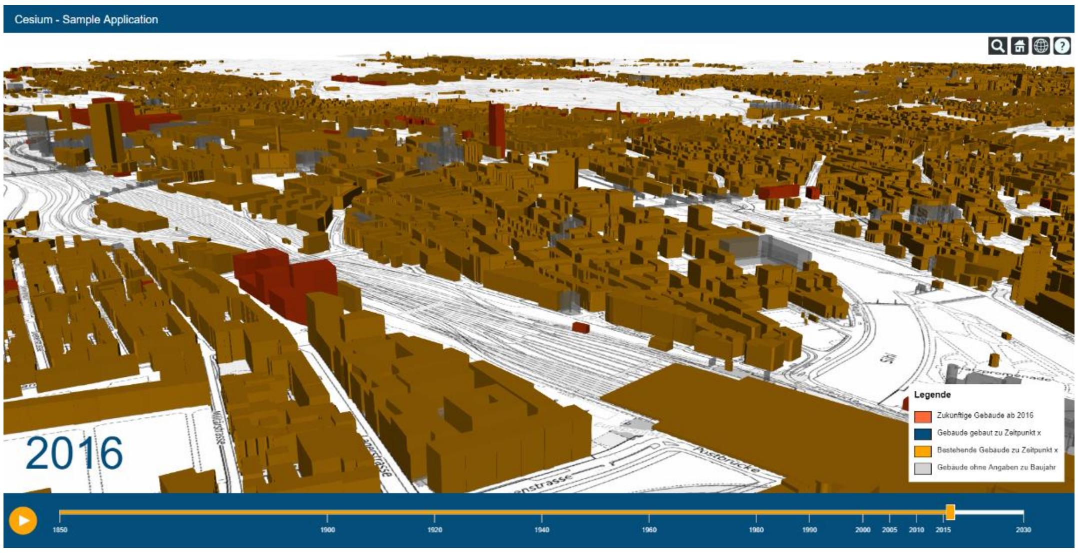

Cesium - Sample Application
Legende
Zukünftige Gebäude ab 2016
Gebäude gebaut zu Zeitpunkt x
Bestehende Gebäude zu Zeitpunkt x
Gebäude ohne Angaben zu Baujahr
2016
1850
1900
1920
1940
1960
1980
1990
2000
2005
2010
2015
2030

Figure 10: Cesium application with future buildings (on title page)

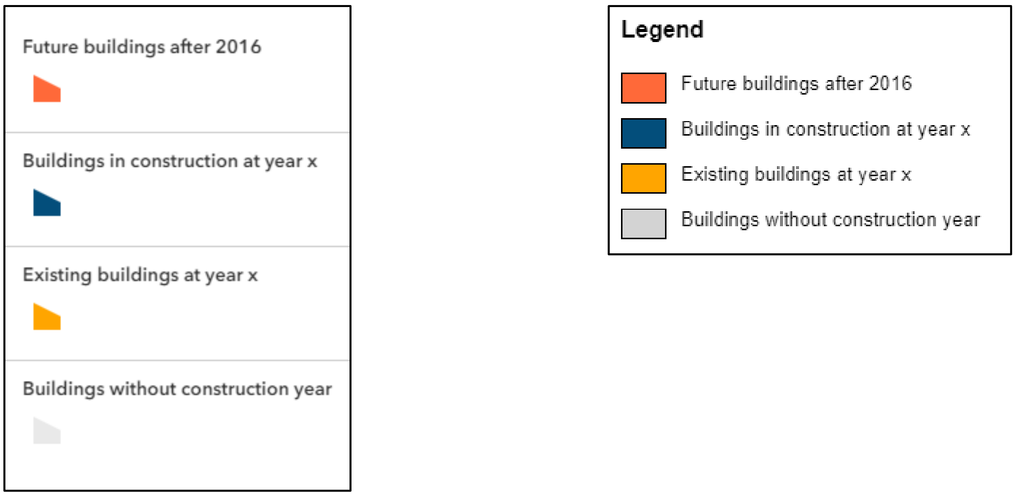

Future buildings after 2016
Buildings in construction at year x
Existing buildings at year x
Buildings without construction year
Legend
Future buildings after 2016
Buildings in construction at year x
Existing buildings at year x
Buildings without construction year

Figure 11: Legend Design (left: ArcGIS API for JavaScript (Widget), left: Cesium (manual))