VTS: Getting Started with 3D Map Application Development

Ladislav Horký, Head of Photogrammetry, Melown Technologies SE

melown
TECHNOLOGIES

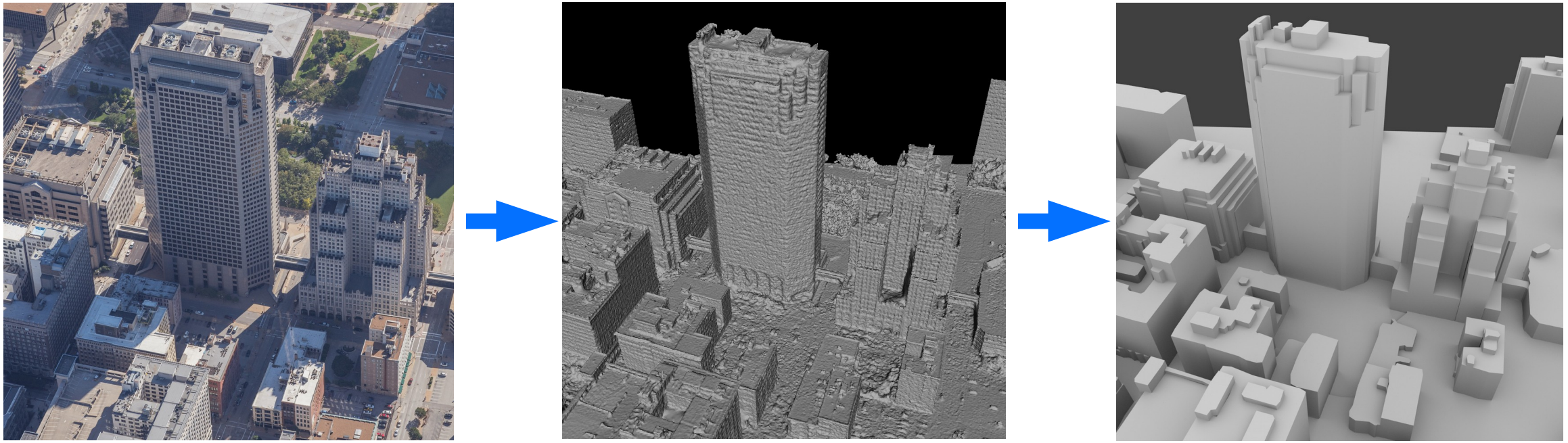
Melown Tech is a software-development company in the 3D mapping business

melown TECHNOLOGIES

www.melown.com

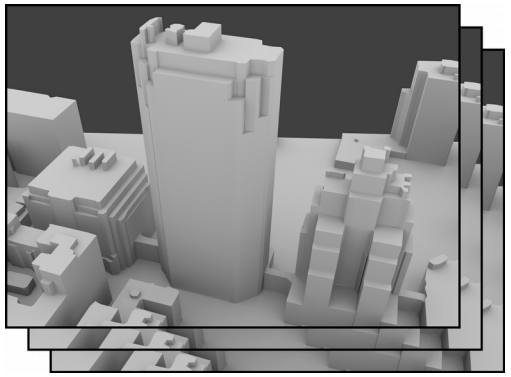
What our software does?

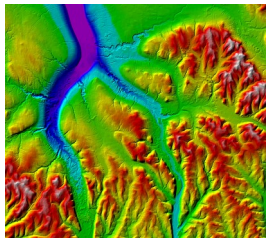
I) Computer-vision and deep-learning driven reality capture



What our software does?

II) 3D data fusion, virtual landscape streaming and rendering





Applications

- VR and AR
- interactive simulations
- gaming
- geospatial (3D mapping)

What this talk is about

I. VTS – an alternative approach to 3D mapping
II. VTS componenets and concepts
III. Your first JavaScript 3D map application

VTS – an alternative approach to 3D mapping

There is CesiumJS …

… there are Google maps, so …

… who needs another 3D map browser?

Who needs another 3D ~~map~~ ~~browser~~?

virtual landscape

integrated system

- VTS consists of the streaming servers, data management tool suite and the clients
- Tight client-server integration: repetitive and static tasks are moved to BE

- Generic geospatial applications
- Augmented reality, virtual reality, gaming

Who needs ~~another~~ mature 3D ~~map~~ ~~browser~~?

- VTS is already the second generation of the technology
- Successor of the proprietary TS renderer, developed in 2013
- Designed to directly compete with Google Maps in Czechia
- Pioneer of 3D content streaming

Everyone who works with lots of 3D data

… ~~who~~ needs mature integrated system for virtual-landscape streaming and rendering?!!

VTS strong points

- **Data scalability:** TBs of 3D and 2.5D data can be fused together within VTS
- **High performance streaming servers:** bandwidth-optimized dynamic TIN and orthophoto generation, static tiles streaming
- **Lightweight and fast client:** available for web and desktop

“The more data you have, the more VTS shines.”

VTS components and concepts

Streaming servers

VTS Mapproxy

- Dynamically converts non-VTS resources to surfaces, bound layers and free layers
- Powerful SRS transforming TMS server

VTSD

- Streams static data and fusion products (glues) from VTS storage
- Serves mapConfig.json(s) – translated from storage view

Data management tool suite

vts tool

- Swiss-knife of the VTS data management
- Add, removes data from the VTS storage, performs the data fusion

generatevrtwo, mapproxy-calipers, mapproxy-tiling, ...

- Preparation and preprocessing of data for the VTS Mapproxy

VTS clients

Web

- WebGL, JavaScript based

Desktop

- Available on Linux, Windows, Mac, iOS
- Common core is available as a rendering-engine independent library

UNITY

- Game engine plugin
- Coming soon…

Your first JavaScript 3D map application

What you need to write your first app

Browser

- cdn.melown.com
- Clone it from GitHub
- Add it to your project using NPM

mapConfig.json

- Grab one from live JSFiddle examples
- Grab one from Melown Tech website
- Install your instance VTS-Backend: Linux, single debian package, sample data included

Basic app: HTML

```
<html>
<head>
    <title>Cadastre sample</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="//cdn.melown.com/libs/vtsjs/builtin/v2/browser/vts-browser.min.css"></link>
    <script type="text/javascript" src="//cdn.melown.com/libs/vtsjs/builtin/v2/browser/vts-browser.min.js"></script>
    <script type="text/javascript" src="./main.js"></script>
</head>
<body style = "padding: 0; margin: 0;" onload="startDemo()">
    <div id="map-div" style="width:100%; height:100%;"></div>
</body>
</html>
```

Load the browser from CDN

Basic app: HTML

```
<html>
<head>
    <title>Cadastre sample</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="//cdn.melown.com/libs/vtsjs/builtin/v2/browser/vts-browser.min.css"></link>
    <script type="text/javascript" src="//cdn.melown.com/libs/vtsjs/builtin/v2/browser/vts-browser.min.js"></script>
    <script type="text/javascript" src="./main.js"></script>
</head>
<body style = "padding: 0; margin: 0;" onload="startDemo()">
    <div id="map-div" style="width:100%; height:100%;"></div>
</body>
</html>
```

Load JavaScript of our app

Full page div that will contain the map, startDemo() called once the page is loaded

Basic app: JavaScript

```
var browser = null;


function startDemo() {
    browser = vts.browser('map-div', {
        map: 'http://localhost:8070/store/map-config/cadastre/mapConfig.json'
    });
}
```

MapConfig.json contains URLs of resources and basic metadata to allow the browser render the scene.

Basic app: the result

Setting custom initial view: JavaScript

```
...
browser = vts.browser('map-div', {
    map: 'http://localhost:8070/store/map-config/cadastre/mapConfig.json',
    view: {
        surfaces: {
            gtopo30-dem-global: [ "bmng" ],
            cz-dem-local: [ "mapy-cz" ],
            Jenstejn-village: [],
            Jenstejn-center: []
        },
        freeLayers: {}
    }
});
...
```

Surfaces derived from DEMs, dynamically generated

Static 3D surfaces

Setting custom initial view: JavaScript

```
...
browser = vts.browser('map-div', {
    map: 'http://localhost:8070/store/map-config/cadastre/mapConfig.json',
    view: {
        surfaces: {
            gtopo30-dem-global: [ "bmng" ],
            cz-dem-local: [ "mapy-cz" ],
            Jenstejn-village: [],
            Jenstejn-center: []
        },
        freeLayers: {}
    }
});
...
```

Bound layers (imagery) draped over DEM surfaces.

Hide vector and raster cadastre

Setting custom initial view: JavaScript

```
...
browser = vts.browser('map-div', {
    map: 'http://localhost:8070/store/map-config/cadastre/mapConfig.json',
    view: {
        surfaces: {
            gtopo30-dem-global: [ "bmng" ],
            cz-dem-local: [ "mapy-cz" ],
            Jenstejn-village: [],
            Jenstejn-center: []
        },
        freeLayers: {}
    }
});
...
```

Symbolic names of resources are defined in mapConfig.json

Basic app: after hiding free layers

Switching cadastre overlays: JavaScript

```
var flswitch = null;
. . .
Function startDemo () {
    . . .
    var panel = browser.ui.addControl('flswitch-panel',
       '<form id="fl-switch-div" class="fl-switch-div">' +
          '<legend>Display cadastre</legend>' +
          '<input name="cadastre" type="radio" value="none" checked="checked"> None <br>' +
          '<input name="cadastre" type="radio" value="vector"> Vector <br>' +
          '<input name="cadastre" type="radio" value="raster"> Raster' +
       '</form>'
    );
    flswitch = panel.getElement('fl-switch-div');
    flswitch.on('change', onSwitchFreeLayer);
}
```

browser.ui takes care of event propagation, provides convenience wrapper around DOM elements

HTML snippet of control element: form with three radio buttons

Switching cadastre overlays: JavaScript

```
var flswitch = null;
. . .
Function startDemo () {
    . . .
    var panel = browser.ui.addControl('flswitch-panel',
        '<form id="fl-switch-div" class="fl-switch-div">' +
           '<legend>Display cadastre</legend>' +
           '<input name="cadastre" type="radio" value="none" checked="checked"> None <br>' +
           '<input name="cadastre" type="radio" value="vector"> Vector <br>' +
           '<input name="cadastre" type="radio" value="raster"> Raster' +
        '</form>'
    );
    flswitch = panel.getElement('fl-switch-div');
    flswitch.on('change', onSwitchFreeLayer);
}
```

Storing form handle as a global variable, registering on-change callback

Switching cadastre overlays: JavaScript

```
function onSwitchFreeLayer() {
    var view = browser.map.getView();

    switch (flswitch.getElement().elements['cadastre'].value) {
    case 'none':
        view.freeLayers = {};
        break;
    case 'vector':
        view.freeLayers = { "cadastre" :  { "style" : "/store/stylesheet/cuzk-cadastre-style.json" } };
        break;
    case 'raster':
        view.freeLayers = { "samples-czdem" : { "boundLayers": ["cadastre-raster"],
                            "depthOffset" : [-5, 0, -40] } };
        break;
    }
    browser.map.setView(view);
}
```

Query the current view

Hide all free layers

Display vector free layer, set how the lines, points and labels will look

Display transparent raster cadastre draped over mesh free layer

Apply modified view

Switching cadastre overlays: HTML

```
<html>
<head>
    ...
    <style>
    .fl-switch-div {
        font-family: Arial, 'Helvetica Neue', Helvetica, sans-serif;
        position: absolute;
        right: 30px;
        bottom: 30px;
        background-color: #eee;
        padding: 10px 20px;
        border-radius: 5px;
        border: solid 1px #000;
    }
    </style>
</head>
...
```

Style for the control element div

Switching cadastre overlays

Switching surfaces: JavaScript

```
var villageCenterCheckbox = null;
var villageCheckbox = null;
. . .
Function startDemo () {
    . . .
    var panel2 = browser.ui.addControl('surface-switch',
        '<form id="surface-switch-div" class="surface-panel-div">' +
           '<legend>Display 3D</legend>' +
           '<input id="Jenstejn" type="checkbox" checked="checked"> Village center <br>' +
           '<input id="Jenstejn-village" type="checkbox" checked="checked"> Village' +
        '</form>'
    );
    panel2.getElement('surface-switch-div').on('change', onSwitchSurface);
    villageCenterCheckbox = panel2.getElement('Jenstejn');
    villageCheckbox = panel2.getElement('Jenstejn-village');
}
```

Register on-change callback on the form, store handles of checkboxes

Switching surfaces: JavaScript

```
function onSwitchSurface() {
    var view = browser.map.getView();
    if (villageCenterCheckbox.getElement().checked) {
        view.surfaces['Jenstejn'] = [];
    } else {
        delete view.surfaces['Jenstejn'];
    }

    if (villageCheckbox.getElement().checked) {
        view.surfaces['Jenstejn-village'] = [];
    } else {
        delete view.surfaces['Jenstejn-village'];
    }

    browser.map.setView(view);
}
```

Add surface corresponding to the given checkbox to view.surfaces or delete it from there

Apply modified view

Switching surfaces: HTML

```
<html>
<head>
     ...
     <style>
     .surface-panel-div {
         font-family: Arial, 'Helvetica Neue', Helvetica, sans-serif;
         position: absolute;
         right: 30px;
         bottom: 140px;
         background-color: #eee;
         padding: 10px 20px;
         border-radius: 5px;
         border: solid 1px #000;
     }
     </style>
</head>
...
```

Switching surfaces

Sources of VTS information

vtsdocs.melown.com

github.com/Melown

github.com/Melown/vts-browser-js/wiki/Examples

Getting involved in VTS development

contact: community@melown.com

or fork us on GitHub ;-)