Code Listing: 3d

Max Mathys

February 4, 2016

1 tests/3d/index.html

```
<!DOCTYPE html>
<html lang="de-ch">
  <head>
      <meta charset="utf-8">
      <meta name="viewport" content=
      ↪  "width=device-width; initial-scale=1.0; maximum-scale=1.0; user-scalable=0;"
      ↪  >
      <meta name="description" content="Template">

      <title>3D</title>

      <!-- Gemeinsames CSS, für alle Tests -->
      <link rel="stylesheet" href="/css/style.css" type="text/css">

      <!-- Testspezifisches CSS -->
      <link rel="stylesheet" href="style.css" type="text/css">
  </head>

  <body>

    <!-- Titel, Auswahl Datensätze -->
    <div class="toggle container">
      <p class="caps toggle-title">Dreidimensionaler Scatterplot</p>
      <div class="clearfix">
        <div class="sm-col sm-col-4 px2">
          <div class="red box"></div> <span id="xtext" style="color:black;"
          ↪  ></span> (X)
        </div>
        <div class="sm-col sm-col-4 px2">
          <div class="green box"></div> <span id="ytext" style="color:black;"
          ↪  ></span> (Y)
        </div>
        <div class="sm-col sm-col-4 px2">
          <div class="blue box"></div> <span id="ztext" style="color:black;"
          ↪  ></span> (Z)
        </div>
```

```
32        </div>
33      </div>
34
35      <!-- Scroll-Block und Container für die Visualisation -->
36      <div id="scroll">
37        <div id="visualization-container">
38          <div id="visualization-wrap">
39          </div>
40        </div>
41        <div id="visualization-container">
42          <div id="visualization-wrap2">
43          </div>
44        </div>
45      </div>
46
47      <hr>
48
49      <!-- Auswahl der Ansichten, Projektionen -->
50      <div class="container clearfix btns">
51        <button class="btn btn-primary" id="toPerspective">Zentralprojektion
          ↪  </button>
52        <button class="btn btn-primary" id="toOrtho">Parallelprojektion</button>
53        <div class="divider"></div>
54        <button class="btn btn-outline" id="xy"><div class="red box"></div> +
          ↪  <div class="green box"></div>:<br>xy-Projektion</button>
55        <button class="btn btn-outline" id="xz"><div class="red box"></div> +
          ↪  <div class="blue box"></div>:<br>xz-Projektion</button>
56        <button class="btn btn-outline" id="zy"><div class="blue box"></div> +
          ↪  <div class="green box"></div>:<br>zy-Projektion</button>
57      </div>
58
59      <br><br>
60
61      <!-- JavaScript Bibliotheken -->
62      <script src="/js/d3.js"></script>
63      <script src="/js/jquery-2.1.4.min.js"></script>
64      <script src="/js/three.min.js"></script>
65      <script src="/js/OrbitControls.js"></script>
66      <script src="/js/CombinedCamera.js"></script>
67      <script src="/js/tween.min.js"></script>
68
69      <!-- JavaScript des Tests -->
70      <script src="script.js"></script>
71    </body>
72  </html>
```

2 tests/3d/meta.json

```
{
  "datasets": [
    {
      "url":"data/CHE_LP_M.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Bevölkerung",
          "unit": "Mio. Personen"
        }
      ]
    },
    {
      "url":"data/CHE_LE_M.csv",
      "config":
      [
        {
          "row": "Date",
          "type": "index",
          "data_type": "Date",
          "date_format": "%Y-%m-%d",
          "name": "Datum"
        },
        {
          "row": "Value",
          "type": "value",
          "data_type": "Number",
          "name": "Anstellung",
          "unit":"Mio. Personen"
        }
      ]
    }
  ]
}
```

3 tests/3d/script.js

```
var sort = require('./modules/sort')
var id = require('./modules/id')
var format = require('./modules/format')
var filter = require('./modules/filter')
var domain = require('./modules/domain')
var range = require('./modules/range')

/* global d3, alert, £, THREE, TWEEN, requestAnimationFrame */

/*
 *
 *
 * Initialisierung Visualisation
 *
 *
 */

// Für die Visualisation benötigte Variablen

var config,           // Config-Array für _alle_ Elemente
  datasetsMeta,       // Das 'datasets'-Attribut von meta.json
  index,              // Config-Objekt für die Index-Spalte (X-Wert)
  values,             // Config-Array für Werte-Spalten (Y-Werte)
  v_accessor,         // Funktion, die den Werteaccessor zurückgibt
  v_accessor_cord,    // Funktion, die den Koordinatenaccessor zurückgibt
  v_accessor_scaled,  // Funktion, die den skalierten Wert zurückgibt.
  v_bundle,           // Objekt, das die drei v-Funktionen enthält.

  xScale,             // x-Skala
  yScale,             // y-Skala
  zScale,             // z-Skala
  xWertebereich,      // Bereich der x-Werte
  yWertebereich,      // Bereich der y-Werte
  zWertebereich       // Bereich der z-Werte

/**
 * Laden der Konfigurationsdatei
 * @param  {[String]} "meta.json"             Der Dateiname für die
 *                                            Konfigurationsdatei
 * @param  {[Function]} function(err, config) Callback
 */
d3.json('meta.json', function (err, res) {
  if (err) {
    console.log(err)
    alert(err)
    return
  }
```

```
48
49    config = []
50    datasetsMeta = res.datasets
51
52    index = {}
53    values = []
54
55    var colors = d3.scale.category20()
56
57    for (var i = 0; i < datasetsMeta.length; i++) {
58      var dataset = datasetsMeta[i]
59      var url = dataset.url
60
61      for (var j = 0; j < dataset.config.length; j++) {
62        var c = dataset.config[j]
63        c.url = url
64
65        // ID generieren
66        c.rowId = id.get(c)
67
68        config.push(c)
69
70        // Einfügen der Config in index oder values
71        if (c.type === 'index') {
72          index = c
73        } else if (c.type === 'value') {
74          // Spaltenspezifische Farbe generieren
75          c.color = colors(values.length + 1)
76
77          // Wenn das Attribut activated nicht gesetzt ist, setze es auf true.
78          if (typeof c.activated === 'undefined') {
79            c.activated = true
80          }
81          values.push(c)
82        }
83      }
84    // Bei unbekannten Typen: nicht in values oder index einfügen.
85    }
86
87    // Datentyp der Skalen festlegen
88    if (index.data_type === 'Number') {
89      xScale = d3.scale.linear()
90    } else if (index.data_type === 'Date') {
91      xScale = d3.time.scale()
92    }
93
94    if (values[0].data_type === 'Number') {
95      yScale = d3.scale.linear()
96    } else if (values[0].data_type === 'Date') {
```

```
97        yScale = d3.time.scale()
98      }
99
100     if (values[1].data_type === 'Number') {
101       zScale = d3.scale.linear()
102     } else if (values[1].data_type === 'Date') {
103       zScale = d3.time.scale()
104     }
105
106
        ↪  // Wertebereich der Achsenskalierungen definieren. Hier ist die Anzahl der Pixel
107     // gemeint, über die sich die Achsen erstrecken. Die x-Achse und die y-Achse
108     // verschieben wir um 50 nach rechts, damit man die y-Achse beschriften kann.
109     xScale.range([0, 100])
110     yScale.range([0, 100])
111     zScale.range([0, 100])
112
113     /*
114      *
115      *
116      * Accessors für die Daten
117      *
118      *
119      */
120
121     // Index-Accessor-Funktion: Gibt für eine bestimmte Datenreihe den Wert der
122       // Index-Spalte zurück.
123
124     index.accessor = function (d) {
125       return d[index.row]
126     }
127
128     // ..._scaled: Gibt den skalierten Wert von accessor zurück.
129     index.accessor_scaled = function (d) {
130       return xScale(d[index.row])
131     }
132
133
        ↪  // Funktion, welche die Werte-Accessor-Funktion zurückgibt. Da sich die Werte-
134     // Accessor-Funktionen im Gegensatz zum statischen Index-Accessor unterschei-
135     // den, müssen sie für jede Spalte neu generiert werden. Diese Funktion ist
136     // dafür zuständig.
137
138     v_accessor = function (entry) {
139       return function (d) {
140         return d[entry.rowId]
141       }
142     }
143
```

```
144     v_accessor_scaled = function (entry) {
145       return function (d) {
146         return yScale(d[entry.rowId])
147       }
148     }
149
150     // Funktion, die den Koordinatenaccessor für die in entry angegebene Spalte
151     // zurückgibt.
152     v_accessor_cord = function (index, entry) {
153       return function (d) {
154         return [index.accessor_scaled(d), v_accessor_scaled(entry)(d)]
155       }
156     }
157
158     v_bundle = {
159       'raw': v_accessor,
160       'scaled': v_accessor_scaled,
161       'cord': v_accessor_cord
162     }
163
164     // Die Daten laden
165     loadFiles()
166   })
167
168   /*
169    *
170    *
171    * Laden der Daten
172    *
173    *
174    */
175
176   /**
177    * Die Funktion, die den Datensatz lädt und vorbereitet
178    *
179    * Vorgehen:  1. Laden der Daten
180
       ↪    *                                                    2. Formatieren des Datensatzes (data_types
181
       ↪    *                                                    3. 'Mergen' mit den anderen Datensätzen, d
182    *                                                  4. Sortieren
183
       ↪    *                                                    5. Die gemergten Datensätze weitergeben
184    */
185   function loadFiles () {
186     // Anzahl von Dateien, die schon heruntergeladen wurde
187     var loaded = 0
188
189     // Die Variable für die gemergten Datensätze
```

```
190     var data = []
191
192     // Jedes einzelne File herunterladen (1)
193     for (var i = 0; i < datasetsMeta.length; i++) {
194       d3.csv(datasetsMeta[i].url, mkcb(i))
195     }
196
197     /**
198      * Funktion, die die Callback-Funktion für einen bestimmten Datensatz-Meta-
199      * daten-Objekt mit Index i zurückgibt. Siehe auch: MKCB-Problem
200      * @param  {[Number]} i   Index des Datensatz-Metadaten-Objekts aus
201      *                        datasetsMeta
202      * @return {[Function]}  Das generierte Callback, das nach dem Laden der
203      *                        Datei ausgeführt wird
204      */
205     function mkcb (i) {
206       return function (err, resp) {
207         if (err) {
208           alert(err)
209           console.log(err)
210           return
211         }
212
213         // Formatieren (2)
214         resp = format.data_types(resp, datasetsMeta[i].config)
215         resp = format.ids(resp, datasetsMeta[i].config)
216
217         // Merge (3)
218         for (var j = 0; j < resp.length; j++) {
219           data.push(resp[j])
220         }
221
222         if (++loaded === datasetsMeta.length) {
223           // Alle Datein sind heruntergeladen worden und gemergt.
224
225           // Sortieren (4)
226           data = sort(data, index)
227
228           // Weitergeben (5)
229           loadVisualization(data)
230         }
231       }
232     }
233   }
234
235   /*
236    *
237    *
238    * Laden der Visualisation
```

```
239   *
240   *
241   */
242
243  /**
244   * Lädt die Visualisation
245   * @param  {[Array]} data Die gemergten Datensätze
246   */
247  function loadVisualization (data) {
248    $('#xtext').html((index.name ? index.name : index.row))
249    $('#ytext').html((values[0].name ? values[0].name : values[0].row) + ' in ' +
       ↪  values[0].unit)
250    $('#ztext').html((values[1].name ? values[1].name : values[1].row) + ' in ' +
       ↪  values[1].unit)
251
252    xWertebereich = domain.overflowX(data, index, 1.1)
253
254    yWertebereich = []
255    zWertebereich = []
256
257    yWertebereich[0] = range.min(data, v_bundle.raw(values[0]))
258    yWertebereich[1] = range.max(data, v_bundle.raw(values[0]))
259    yWertebereich[1] = range.applyOverflow(yWertebereich[0], yWertebereich[1],
260      1.1, values[0].data_type)
261
262    zWertebereich[0] = range.min(data, v_bundle.raw(values[1]))
263    zWertebereich[1] = range.max(data, v_bundle.raw(values[1]))
264    zWertebereich[1] = range.applyOverflow(zWertebereich[0], zWertebereich[1],
265      1.1, values[1].data_type)
266
267    xScale.domain(xWertebereich)
268    yScale.domain(yWertebereich)
269    zScale.domain(zWertebereich)
270
271    var camera, controls, scene, renderer, material
272    init()
273    render()
274    function animate () {
275      requestAnimationFrame(animate)
276      controls.update()
277      TWEEN.update()
278    }
279    function init () {
280      scene = new THREE.Scene()
281
282      var w = window.innerWidth * 0.8
283      var h = window.innerHeight * 0.8
284
285      // var cameraP = new THREE.PerspectiveCamera(45, w / h, 1, 10000)
```

```
286
        // var camera0 = new THREE.OrthographicCamera(w / -2, w / 2, h / 2, h / -2, 1, 1000)
287     camera = new THREE.CombinedCamera(w / 2, h / 2, 70, 1, 1000, -500, 1000)
288
289     // override.
290     camera.setZoom(1)
291     camera.toPerspective()
292
293     camera.position.x = 80
294     camera.position.y = 70
295     camera.position.z = 150
296
297     material = new THREE.MeshBasicMaterial({ color: 0x000000, wireframe: false
        })
298
299     renderer = new THREE.WebGLRenderer({ alpha: true, antialias: true })
300     renderer.setSize(w, h)
301
302     controls = new THREE.OrbitControls(camera, renderer.domElement)
303     controls.damping = 0.2
304     controls.addEventListener('change', render)
305
306     axis()
307
308     document.getElementById('visualization-wrap').appendChild(renderer.
        domElement)
309
310     animate()
311
312     xScale.range([0, 100])
313     yScale.range([0, 100])
314
315     points()
316   }
317
318   function render () {
319     renderer.render(scene, camera)
320
        //console.log('cords: ', camera.position.x, camera.position.y, camera.position.z)
321
        //console.log('in scene: ', camera.position.x + 50, camera.position.y + 50, camera.positi
322
        //console.log(camera.rotation.x*180/Math.PI, camera.rotation.y*180/Math.PI, camera.rotati
323   }
324
325   function toScene (x, y, z) {
326     return [x - 50, y - 50, z - 50]
327   }
328
```

```
329   function axis () {
330     var origin = new THREE.Vector3(-50, -50, -50)
331     var length = 100
332
333     scene.add(new THREE.ArrowHelper(new THREE.Vector3(1, 0, 0), origin, length,
        ↪ 0xff0000))
334     scene.add(new THREE.ArrowHelper(new THREE.Vector3(0, 1, 0), origin, length,
        ↪ 0x00ff00))
335     scene.add(new THREE.ArrowHelper(new THREE.Vector3(0, 0, 1), origin, length,
        ↪ 0x0000ff))
336
337     var dashed = new THREE.LineDashedMaterial({
338       color: 0xdedede,
339       dashSize: 3,
340       gapSize: 2,
341       scale: 1
342     })
343
344     // Box zeichnen
345     var xy1 = [toScene(100, 100, 0), toScene(100, 0, 0)]
346     var xy2 = [toScene(100, 100, 0), toScene(0, 100, 0)]
347     var yz1 = [toScene(0, 100, 100), toScene(0, 100, 0)]
348     var yz2 = [toScene(0, 100, 100), toScene(0, 0, 100)]
349     var xz1 = [toScene(100, 0, 100), toScene(100, 0, 0)]
350     var xz2 = [toScene(100, 0, 100), toScene(0, 0, 100)]
351     var xyz1 = [toScene(100, 100, 100), toScene(100, 100, 0)]
352     var xyz2 = [toScene(100, 100, 100), toScene(100, 0, 100)]
353     var xyz3 = [toScene(100, 100, 100), toScene(0, 100, 100)]
354
355     for (var i = 0; i < 9; i++) {
356       var a = [xy1, xy2, xz1, xz2, yz1, yz2, xyz1, xyz2, xyz3][i]
357       var lg = new THREE.Geometry()
358       lg.vertices.push(new THREE.Vector3(a[0][0], a[0][1], a[0][2]))
359       lg.vertices.push(new THREE.Vector3(a[1][0], a[1][1], a[1][2]))
360       lg.computeLineDistances()
361       var line = new THREE.Line(lg, dashed)
362       scene.add(line)
363     }
364   }
365
366   function points () {
367     var dataY = filter.row(data, values[0].rowId)
368     var dataZ = filter.row(data, values[1].rowId)
369     for (var i = 0; i < dataY.length; i++) {
370       var x = xScale(index.accessor(dataY[i]))
371       var y = yScale(dataY[i][values[0].rowId])
372       var z = zScale(dataZ[i][values[1].rowId])
373
374       var sphere = new THREE.SphereGeometry(0.5, 8, 6)
```

```
375         var smesh = new THREE.Mesh(sphere, material)
376         var arr = toScene(x, y, z)
377         smesh.translateX(arr[0])
378         smesh.translateY(arr[1])
379         smesh.translateZ(arr[2])
380         scene.add(smesh)
381       }
382     }
383
384     $('#toPerspective').click(function () {
385       camera.toPerspective()
386       camera.setZoom(1)
387       camera.updateProjectionMatrix()
388       render()
389     })
390
391     $('#toOrtho').click(function () {
392       camera.toOrthographic()
393       camera.setZoom(5)
394       camera.updateProjectionMatrix()
395       render()
396     })
397
398     $('#xy').click(function () {
399       ortho('xy')
400     })
401
402     $('#xz').click(function () {
403       ortho('xz')
404     })
405
406     $('#zy').click(function () {
407       ortho('zy')
408     })
409
410     function ortho (mode) {
411       if (camera.inPerspectiveMode) {
412         camera.setZoom(1)
413       } else {
414         camera.setZoom(5)
415       }
416       var cc
417       if (mode === 'xy') {
418         cc = toScene(50, 50, 200)
419       } else if (mode === 'xz') {
420         cc = toScene(50, -100, 50)
421       } else if (mode === 'zy') {
422         cc = toScene(-100, 50, 50)
423       }
```

```
424
425       var posx = { x: camera.position.x }
426       var posy = { x: camera.position.y }
427       var posz = { x: camera.position.z }
428
429       var tarx = { x: cc[0] }
430       var tary = { x: cc[1] }
431       var tarz = { x: cc[2] }
432
433       var tx = new TWEEN.Tween(posx).to(tarx, 1400)
434       var ty = new TWEEN.Tween(posy).to(tary, 1400)
435       var tz = new TWEEN.Tween(posz).to(tarz, 1400)
436
437       tx.easing(TWEEN.Easing.Cubic.InOut)
438       ty.easing(TWEEN.Easing.Cubic.InOut)
439       tz.easing(TWEEN.Easing.Cubic.InOut)
440
441       tx.onUpdate(function () {
442         camera.position.x = posx.x
443       })
444       ty.onUpdate(function () {
445         camera.position.y = posy.x
446       })
447       tz.onUpdate(function () {
448         camera.position.z = posz.x
449       })
450
451       tx.start()
452       ty.start()
453       tz.start()
454     }
455   }
```

4 tests/3d/style.scss

```
1  $graph-borders-color: rgba(0,0,0,.2);
2  $graph-axis-text-color: rgba(0,0,0,.6);
3  $graph-axis-font-size: 14px;
4  $graph-axis-size: 1.5px;
5  $graph-axis-tick-size: 1px;
6  $data-title-font-size: 16px;
7
8  #visualization-container {
9    display: flex;
10   justify-content: center;
11   margin: 0;
12 }
13
```

```
14  #visualization-wrap {
15    position: relative;
16  }
17
18  #display-overlay {
19    position:absolute;
20    display: inline-flex;
21    flex-wrap: nowrap;
22    pointer-events: none;
23    align-content: space-between;
24  }
25
26  .tip-element {
27    border-left-width: 3px;
28    border-left-style: solid;
29    border-left-color: #000;
30    padding: 2px 15px 2px 5px;
31  }
32
33  .tip-title {
34    font-size: 12px;
35    margin-bottom: 2px;
36  }
37
38  .tip-attribute {
39    font-size: 12px;
40    margin: 0;
41  }
42
43  #overlay {
44    // Die Applikation soll die auf Bewegung der Maus zugreifen können. Dieses
45    // Attribut wird gebraucht.
46    pointer-events: all;
47  }
48
49  #graph {
50
51  }
52
53  .domain {
54    fill: none;
55    stroke: $graph-borders-color;
56    stroke-width: $graph-axis-size;
57  }
58
59  .tick > line {
60    fill: none;
61    stroke: $graph-borders-color;
62    stroke-width: $graph-axis-tick-size;
```

```
63
64  }
65
66  .tick > text {
67    fill: $graph-axis-text-color;
68    font-size: $graph-axis-font-size;
69  }
70
71  circle.data-point {
72    r: 2px;
73  }
74
75  circle#tooltip-circle {
76    r: 7px;
77    fill: rgba(0, 0, 0, .4);
78  }
79
80  #label > text {
81    fill: black;
82    font-weight: 400;
83    font-size: $graph-axis-font-size;
84  }
85
86  #overlay {
87    visibility: hidden;
88  }
89
90  .line {
91    fill: none;
92    stroke: rgba(0,0,0,.7);
93    stroke-width: 2px;
94  }
95
96  .hidden {
97    display: none;
98  }
99
100 #select-row {
101   padding: 0 0 30px;
102   margin: 0;
103   display: flex;
104   flex-direction: row;
105   flex-wrap: wrap;
106 }
107
108 .select-row-item {
109   color: #000;
110   padding-bottom: 6px;
111   margin: 12px 20px 0 0;
```

```
    cursor: pointer;
    border-bottom: 2px solid rgba(0, 0, 0, 1);
    border-color: #000;

    transition: all 0.2s cubic-bezier(.5,1,.8,1);

    user-select: none;
  }

  .select-row-item:hover {
    color: black;
    text-decoration: none;
  }

  .inactive {
    color: LightGray;
    border-color: #fff !important;
  }

  .toggle-title {
    margin-bottom: 8px;
  }

  .toggle {
    margin-top: 20px;
  }

  .btn{
    margin: 0 5px;
  }

  .btns {
    display:flex;
    justify-content: center;
  }

  .box {
      width:11px;
      height:11px;
      display:inline-block;
  }

  .red {
    background-color:red;
  }
  .green {
    background-color:green;
  }
  .blue {
```

```
  background-color:blue;
}

.divider {
  border-left:1px solid #000;
  width: 1px;
  margin: 0 5px;
}
```

5 tests/3d/data/CHE_LE_M.csv

```
Date,Value
2016-12-31,4.813
2015-12-31,4.895
2014-12-31,4.918
2013-12-31,4.837
2012-12-31,4.776
2011-12-31,4.705
2010-12-31,4.593
2009-12-31,4.568
2008-12-31,4.548
2007-12-31,4.44
2006-12-31,4.328
2005-12-31,4.24
2004-12-31,4.21
2003-12-31,4.199
2002-12-31,4.213
2001-12-31,4.183
2000-12-31,4.116
1999-12-31,4.076
1998-12-31,4.045
1997-12-31,3.991
1996-12-31,3.994
1995-12-31,3.997
1994-12-31,3.999
1993-12-31,4.024
1992-12-31,4.068
1991-12-31,4.066
1990-12-31,3.819
1989-12-31,3.703
1988-12-31,3.607
1987-12-31,3.515
1986-12-31,3.431
1985-12-31,3.354
1984-12-31,3.288
1983-12-31,3.256
1982-12-31,3.256
1981-12-31,3.24
```

```
1980-12-31,3.166
```

6 tests/3d/data/CHE_LP_M.csv

```
Date,Value
2016-12-31,8.238
2015-12-31,8.189
2014-12-31,8.14
2013-12-31,8.039
2012-12-31,7.955
2011-12-31,7.87
2010-12-31,7.786
2009-12-31,7.702
2008-12-31,7.593
2007-12-31,7.509
2006-12-31,7.459
2005-12-31,7.415
2004-12-31,7.364
2003-12-31,7.314
2002-12-31,7.256
2001-12-31,7.198
2000-12-31,7.164
1999-12-31,7.124
1998-12-31,7.096
1997-12-31,7.081
1996-12-31,7.062
1995-12-31,7.019
1994-12-31,6.969
1993-12-31,6.908
1992-12-31,6.843
1991-12-31,6.757
1990-12-31,6.674
1989-12-31,6.62
1988-12-31,6.567
1987-12-31,6.523
1986-12-31,6.485
1985-12-31,6.456
1984-12-31,6.428
1983-12-31,6.41
1982-12-31,6.373
1981-12-31,6.335
1980-12-31,6.304
```

7 tests/3d/modules/domain.js

```
var range = require('./range')

/**
```

```
 * Modul: Domain
 * -------------
 * Gibt einen überhöhten Wertbereich zurück für X und Y.
 * Überhöhte Wertebereiche werden hier benutzt, damit ein wenig Platz links und
 * oberhalb der Linie entsteht.
 */

/**
 * Gibt überhöhten Wertebereich für X zurück.
 * @param  {[Array]} data             Gemergter Datensatz, ungefiltert
 * @param  {[Object]} index           Config-Objekt für die Index-Spalte
 * @param  {[Number]} overflowFactor  Überhöhungsfaktor
 * @return {[Array]}                  Das Minimum und Maximum in einem Array.
 */
module.exports.overflowX = function (data, index, overflowFactor) {
  var xWertebereich = []
  xWertebereich[0] = range.min(data, index.accessor)
  xWertebereich[1] = range.max(data, index.accessor)
  xWertebereich[1] = range.applyOverflow(xWertebereich[0], xWertebereich[1],
    overflowFactor, index.data_type)
  return xWertebereich
}
/**
 * Gibt überhöhten Wertebereich für Y zurück.
 * @param  {[Array]} data             Gemergter Datensatz, ungefiltert

  ↪   * @param  {[Array]} values           Array von Config-Objekten der Wertespalten
 * @param  {[Object]} v_bundle        Accessor-Bundle
 * @param  {[Number]} overflowFactor  Überhöhungsfaktor
 * @return {[Array]}                  Das Minimum und Maximum in einem Array.
 */
module.exports.overflowY = function (data, values, v_bundle, overflowFactor) {
  var yWertebereich = []
  yWertebereich[0] = range.minMultipleSets(data, values, v_bundle)
  yWertebereich[1] = range.maxMultipleSets(data, values, v_bundle)
  yWertebereich[1] = range.applyOverflow(yWertebereich[0], yWertebereich[1],
    overflowFactor, values[0].data_type)
  return yWertebereich
}
```

8 tests/3d/modules/filter.js

```
/**
 * Modul: filter
 * -------------

  ↪   * Filtert den gemergten Datensatz. Gibt die Zeilen zurück, bei denen die Spalte
 * 'row' gesetzt ist.
```

```
 */

/**
 * Gibt den gefilterten Datensatz zurück.
 * @param  {[Array]} data Ungefilterter, gemergter Datensatz.
 * @param  {[String]} row Name der Spalte, nach der gefiltert werden soll
 * @return {[Array]}      Gefilterter, gemergter Datensatz.
 */

module.exports.row = function (data, row) {
  var ret = []
  for (var i = 0; i < data.length; i++) {
    if (typeof data[i][row] !== 'undefined') {
      ret.push(data[i])
    }
  }
  return ret
}
```

9 tests/3d/modules/format.js

```
var sort = require('./sort')
var id = require('./id')

/**
 * Modul: Format
 * -------------
 * Formatiert den Datensatz
 */

/**
 * Konvertiert die Zeichenketten (Strings) im Datensatz in Javascript-
 * Objekte, wie zum Beispiel Zahlen (Float) oder Daten (Date).
 * @param  {[Array]} data   Unformatierter Datensatz
 * @param  {{Array}} config Array von Config-Objekten
 * @return {[Array]}        Gefilterter Datensatz
 */
module.exports.data_types = function (data, config) {
  // index suchen
  for (var i = 0; i < data.length; i++) {
    for (var j = 0; j < config.length; j++) {
      if (config[j].data_type === 'Number') {
        data[i][config[j].row] = parseFloat(data[i][config[j].row])
      } else if (config[j].data_type === 'Date') {
        data[i][config[j].row] =  d3.time.format(config[j].date_format)
                                    .parse(data[i][config[j].row])
      }
    }
```

```
  }
  return data
}

/**
 * Fügt das Attribut 'rowId' für jedes Objekt hinzu. 'rowId' ist eine aus dem
 * Reihennamen und dem Pfad des Datensatzes generierte einzigartige ID.
 * @param  {[Array]} data   Datensatz ohne rowIds
 * @param  {{Array}} config Array von Config-Objekten
 * @return {[Array]}        Datensatz mit rowIds
 */
module.exports.ids = function (data, config) {
  for (var i = 0; i < data.length; i++) {
    for (var j = 0; j < config.length; j++) {
      if (config[j].type === 'index') {
        continue
      }
      data[i][id.get(config[j])] = data[i][config[j].row]
      delete data[i][config[j].row]
    }
  }

  return data
}
```

10 tests/3d/modules/id.js

```
/**
 * Modul: Id
 * ---------
 * Generiert einzigartige ID für eine Spalte.
 */

/**
 * Gibt generierte ID zurück.
 * @param  {[Object]} config Config-Objekt
 * @return {[String]}        ID
 */
module.exports.get = function (config) {
  return config.row + '#' + config.url
}

/**
 * Gibt Config-Objekt für eine Spalte zurück
 * @param  {[String]} id    ID der Spalte
 * @param  {[Array]} values Array von Config-Objekten aller Datenspalten
 * @return {[Object]}       Config-Objekt der Spalte
 */
```

```
22  module.exports.invert = function (id, values) {
23    for (var i = 0; i < values.length; i++) {
24      if (id === values[i].rowId) {
25        return values[i]
26      }
27    }
28  }
29
30  /**
31   * Gibt ID für benutzerdefinierte Attribute zurück.
32   * @param  {[String]} attr Benutzerdefiniertes Attribut
33   * @param  {[String]} url  URL des Datensatzes
34   * @return {[String]}      ID
35   */
36  module.exports.raw = function (attr, url) {
37    return attr + '#' + url
38  }
```

11 tests/3d/modules/range.js

```
1   /**
2    * Modul: Range
3    * ------------
4    * Wertebereich von Datenspalten bestimmen
5    */
6
7   /**
8    * Gibt das Minimum einer einzelnen Datenspalte zurück
9    * @param  {[Array]} data         Der Datensatz
10
    ↪    * @param  {{Function}} accessor  Der Accessor für die zu untersuchende Datenreihe
11   * @return {[Number]}             Das Minimum
12   */
13  module.exports.min = function (data, accessor) {
14    return d3.min(data, accessor)
15  }
16
17  /**
18   * Gibt das Maximum einer einzelnen Datenspalte zurück
19   * @param  {[Array]} data     Der Datensatz
20   * @param  {{Function}} index Der Accessor für die zu untersuchende Datenreihe
21   * @return {[Object]}         Das Maximum
22   */
23  module.exports.max = function (data, accessor) {
24    return d3.max(data, accessor)
25  }
26
27  /**
```

```
28   * Gibt das Minimum für mehrere Datenspalten zurück.
29   * @param  {[Array]} data           Der Datensatz
30   * @param  {[Array]} values         Der Config-Array für die zu untersuchenden
31   *                                                    Datenreihen
32   * @param  {{Function}} v_accessor  Die Funktion, die für eine bestimmte value-
33   *                                  Reihe den Accessor zurückgibt
34   * @return {[Object]}               Das Minimum
35   */
36  module.exports.minMultipleSets = function (data, values, v_bundle) {
37    var min
38    for (var i = 0; i < values.length; i++) {
39      if (!values[i].activated) {
40        continue
41      }
42      var lmin = d3.min(data, v_bundle.raw(values[i]))
43      if (typeof lmin === 'undefined') {
44        continue
45      }
46      if (typeof min === 'undefined' || lmin < min) {
47        min = lmin
48      }
49    }
50    return min
51  }
52
53  /**
54   * Gibt das Maximum für mehrere Datenspalten zurück.
55   * @param  {[Array]} data           Der Datensatz
56   * @param  {[Array]} values         Der Config-Array für die zu untersuchenden
57   *                                  Datenreihen.
58   * @param  {{Function}} v_accessor  Die Funktion, die für eine bestimmte value-
59   *                                  Reihe den Accessor zurückgibt.
60   * @return {[Object]}               Das Maximum
61   */
62  module.exports.maxMultipleSets = function (data, values, v_bundle) {
63    var max
64    for (var i = 0; i < values.length; i++) {
65      if (!values[i].activated) {
66        continue
67      }
68      var lmax = d3.max(data, v_bundle.raw(values[i]))
69
70      if (typeof max === 'undefined' || lmax > max) {
71        max = lmax
72      }
73    }
74    return max
75  }
76
```

```
77  //  Wertebereich, der Daten bestimmen mit d3: Um einen kleinen Abstand zwischen
78  //  den maximalen Punkten und den Rändern des Diagrammes zu bewahren,
79  //  wird der Unterschied (Δ) des Minimums und dem untersuchten Wert mit 1.1
80  //  mulitpliziert. Anschliessend wird die Summe des Minimums und des
81  //  multiplizierten Wertes an d3 zurückgegeben.
82
83  /**
84   * Gibt die Summe des Minimums und des mit dem Faktor factor multiplizierten
85   * Unterschieds von min und max zurück.
86   * Wird verwendet, damit oben und rechts von Graphen Platz ausgelassen wird.
87   * @param  {[Number]} min       Minimum ohne Overflow
88   * @param  {[Number]} max       Maximum ohne Overflow
89   * @param  {[Number]} factor    Overflow-Faktor
90   * @param  {{String}} data_type Der Datentyp von min und max
91   * @return {[Number]}           Das Maximum mit Overflow.
92   */
93  module.exports.applyOverflow = function (min, max, factor, data_type) {
94    if (data_type === 'Date') {
95      return new Date(min.getTime() + (max.getTime() - min.getTime()) * factor)
96    } else if (data_type === 'Number') {
97      return min + (max - min) * factor
98    }
99  }
```

12 tests/3d/modules/sort.js

```
1   /**
2    * Modul: Sort
3    * -----------
4    * Sortiert einen Datensatz nach der Index-Spalte
5    */
6
7   /**
8    * Array sortieren, aufsteigend
9
    ↪    * https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sor
10   * @return {[Array]} Sortierter Datensatz
11   */
12  module.exports = function (data, index) {
13    data.sort(function (a, b) {
14      if (index.accessor(a) < index.accessor(b)) {
15        return -1
16      }
17      if (index.accessor(a) > index.accessor(b)) {
18        return 1
19      }
20      return 0
21    })
```

```
22
23      return data
24  }
```