S H O W R O O M D E S I G N

Datei *showroomDesign.js*

Materialien m[.] Multi-Materialien mm[.] und Farben c[.] sind in der Datei materials.js zu definieren.

Die Kategorien sollten in dargestellter Reihenfolge definiert werden. Wichtig z.B. bei Leibungen. Die einzelnen Elemente werden in beliebiger Reihenfolge in die jeweiligen Kategorien eingefügt und mit Komma getrennt.

Schreibe die array-Klammern [ ].
Man kann [ ], als Platzhalter benutzen, um später Daten ohne Indexänderung hinzuzufügen.

In der Beschreibung stehen optionale Einträge in ( ). Die Klammern dürfen nicht notiert werden.

Definiere Defaults für optionale Daten ...Default = ... ;

x nach rechts, z nach vorn
y nach oben (y0 unten, y1 oben)

Beleuchtungen

lightings = [ ];

[Ambient, Farbe, Helligkeit], *// sollte als Grundbeleuchtung vorhanden sein*
[Point, x,z,y -Position, Farbe, Helligkeit],
[Spot, x,z,y -Position, x,z,y -Zielposition, Farbe, Helligkeit, Winkel° , Halbschatten],
[LongLamp, x,z,y -Position, Farbe, Helligkeit],

Grundriss (Böden und Decken)

floors = [ ];
ceilings = [ ];

Variante (a) mit individuellem Niveau y.
[x,z,y , .. m[.] (,[uv's])], *// uv's entsprechend Umlauf der Grundrisslinie*

Variante (b) mit einheitlichem Niveau [y].
[x,z, .. [y], m[.] (,[uv's])], *// uv's, entsprechend Umlauf der Grundrisslinie*

x,z Koordinaten für Boden und Decke. Es sind jeweils mehrere unabhängige Flächen möglich. Punkte x,z der Grundrisslinien (im Uhrzeigersinn). Die Linien Ecke zu Schwerpunkt müssen innerhalb der Fläche liegen! Der letzte Punkt wird mit dem ersten Punkt verbunden. Werden keine uv-Werte angegeben, werden sie intern berechnet.

Wände (Strukturbauteile, Innenwände, dicke Trennwände, Aufblockungen, Podeste):
(*ohne [Dicke] nur von innen sichtbar!*)

wallDepthDefault = ... ;
wallIndentDefault = ... ;

walls = [ ];

[x0,z0, (y00,y01,) x1,z1, y10,y11
(, [wd (, Wanddicke)]) oder (,[*wo, Wandöffnung*, ... (, *Einrückung*)])
, m[.] oder mm[.] (,[uv's vorn])],

Werden die Niveaus links (y00,y01,) weggelassen, gelten auf beiden Seiten die Niveaus y10,y11.
mehrere Wandöffnungen, jeweils: Abstand(von links) , Breite, Niveau Unterkante, Höhe
uv's vorn: [u0,v00,v01, u1,v10,v11] passend zu den Abmessungen

---

Leibungen WICHTIG: nach Wänden definieren!

soffitPartsDefault = '...';
soffitSillThicknessDefault = ... ;
soffitSillOverhangDefault = ... ;

soffits = [ ];

[Wandindex, Index Wandöffnung, Tiefe, mm[.] (, 'Teile') (, Schwellendicke, Schwellenüberhang)],

'Teile': l links, r rechts, t oben, b unten, s Schwelle

Torbögen

archwayPartsDefault = '...';
archwaySillThicknessDefault = ... ;
archwaySillOverhangDefault = ... ;

archways = [ ];

Variante (a)
[x0,z0, x1,z1, y0,y1, Dicke, mm[.] (, 'Teile') (, Schwellendicke, Schwellenüberhang)],
x0,z0, x1,z1, Koordinaten der Mitte der Seiten

Variante (b) WICHTIG: nach Wänden definieren!
[Wandindex, Index Öffnung, Dicke, mm[.] (, Schwellendicke, Schwellenüberhang)],

'Teile' *für beide Varianten*:
f Front, o Rückseite, a Wölbung, l links, r rechts, b unten, s Schwelle

---

runde Wände (Polygon):

roundWallViewDefault = '...';

roundWalls = [ ];

[x,z -Zentrum, y0, y1, Radius, Radius Segmente, Startwinkel°, Winkel°, m[.] (, 'e' oder 'c')],

'e' externe oder 'c' Centrums -Ansicht

Bauteile

Mantelfunktionen für Bauteile (Höhe => Durchmesser)
Funktionen: dH[.] = h => term; def[0,1] => value[0,1]

dHdefault = h => 1; nicht ändern, Durchmesser konstant

Beispiel: `dH[0] = h => 0.5 + 0.25 * sin( 10 * h );`

componentRotationDefault = ... ;

components = [ ];

[x,z -Zentrum, y0, y1, Radialsegm., Höhensegm., Durchmesser, mm[.] (, dH[.]) (, Rotation°)],
mm[.] Reihenfolge: Mantel (wenn nicht dHdefault const. nur Farbe als Material), Deckel, Boden

---

Spiegel

mirrorTiltDefault = ... ;
mirrorColorDefault = c[...]; // siehe colors in material.js

mirrors = [ ];

[Rectangle, x,z, y , Breite, Höhe, y-rotation° (, [ti, Neigung°]) (, [color])
[Polygon, x,z, y, Radius, n.rotation°, yrotation° (, [ti, Neigung°]) (, [color])

WICHTIG: nach Wänden definieren!
[WallRectangle, Wandindex, links, unten, Breite, Höhe, Abstand (, [ti, Neigung°]) (, [color])
[WallPolygon, Wandindex, llinks, unten, Radius, n.rotation°, Abstand (, [ti, Neigung°]) (, [color])

n.rotation°
ganzzahliger Teil der Zahl ist die Eckenzahl, 3 Ziffern hinter dem Punkt ist die Drehung der Ecken

---

Rahmen (js Dateien im Unterordner *frames*)

Rahmengeometrie gFrame[.] definiert in frames/ frames.js
(erzeugt mit ConstructFrameShowroom.html)

frameIndentDefault = ... ;
frameXscaleDefault = ... ;
frameZscaleDefault = ... ;
frameYscaleDefault = ... ;

frames = [ ];

Variante (a)
[frameID, x,z, y, y-Rotation°, mm[.] (, xScale (, zScale(, yScale)))],

Variante (b) WICHTIG: nach Wänden definieren!
[frameID, Wandindex, Index Öffnung (, Einrückung) , mm[.] (, xScale (, zScale(, yScale)))],

---

Objekte (3D-Formate: Gltf, Obj im Unter-Unterordner *objects3D*/*Dateiname*)

objects3D = [ ];

[3D-format, 'Dateiname', x,z, y- Skalierung, x,z, y- Position, x,z, y- Rotation°],

Bei Obj muss eine gleichnamige .mtl im Ordner existieren.

Umgebung

surroundingTexture = ['posx.jpg', 'negx.jpg', 'posy.jpg', 'negy.jpg', 'posz.jpg', 'negz.jpg'];

Die Dateinamen sind frei wählbar, wichtig ist die Reihenfolge.

Texturen für +x,-x,+y,-y,+z,-z, im Unterordner *CubeMap*. Beachte: three.js Reihenfolge x,y,z

---

Betrachter (nur 3D-Format gltf, im Unter-Unterordner *objects3D*/*Dateiname*)

visitor = ['Dateiname', x,z, y- Skalierung, Augenhöhe, x,z- Startposition, y- Startrotation°];

Wird mit dem Control in den begehbaren Bereichen bewegt, kann sich selbst nur im Spiegel sehen.

Begehbare Bereiche

walkableAreas = [ ];

[Rectangle, xMin, zMin, xMax, zMax],
[Circle, x, z, Radius],
[Triangle, xa,za, xb,zb, xc,zc], (*Dreieckspunkte im Uhrzeigersinn*)

Die Bereiche sollten sich ausreichend berühren bzw. überschneiden um einen Durchgang zu ermöglichen.

M A T E R I A L I E N

Datei *materials.js*

definiere Farben, Materialien, Multi-Materialien
schreibe die array Klammern [ ] , optionale Data in ( ) - diese Klammern nicht notieren

---

colors (Farbschema zur Verwendung in m, benutze dort c[.])

c = [ ]; // array von hexadezimalen Farbwerten

Beispiel:

```
c = [

  0x000000,   // 0 Schwarz
  0xffffff,   // 1 Weiß
  0xff0000,   // 2 Rot
  0x00ff00,   // 3 Grün
  0x0000ff,   // 4 Blau
  0xdedede,   // 5 helles Grau

];
```

---

materials

m = [ ];

Side: Front (ist default), Back, Double,

Empty - Nichts wird erzeugt, nutzbar für leere Wandöffnungen, auch Platzhalter.
Basic oder Phong oder Lambert, c[.] (, Side) (, opacity)
Texture, Grafikdatei (, Side) (, [wrap S, wrap T]) (, opacity)
Wireframe, c[.] (, Side)
Video, video file (, Side)

example:

```
m = [

  [ Empty ],                                    //   0
  [ Texture, 'uvgrid01.png', Double ],          //   1
  [ Texture, 'floorTile.png',[ 40, 32 ] ],      //   2
  [ Texture, 'tile.png', [ 20, 20 ], 0.85 ],    //   3
  [ Texture, 'beech.jpg', Double, 0.5 ],        //   4
  [ Phong, c[2] ],                              //   5
  [ Lambert, c[0], 0.66 ],                      //   6
  [ Wireframe, c[5], Double ],                  //   7
  [ Empty ],                                    //   8 Platzhalter
  [ Video, 'Raindrops_Videvo.mp4' ],            //   9

];
```

---

multi materials

arrays von Indizes des Material arrays m[ ]

1 + Öffnungsanzahl Materialiens für Wände mit Öffnungen, Index 0 für die Wand

6 Materialien für dicke Wände (... Bauteile usw.)

6 oder 10 Materialien für Torbögen:
vorn, Wölbung, hinten, links, rechts, unten oder Schwelle(oben, vorn, Unterseite, links, rechts)

3 Materialien für Bauteile :
Mantel (nur Farbe, wenn nicht dHdefault const.), Oberseite, Unterseite

Wenn in der [ ] zu wenige Materialien angegeben werden, wird das letzte Material weiterhin genutzt.

mm = [ ];

example:

```
mm = [

  [ 0 ],                //  0 immer Material 0 ( Empty / Leer )
  [ 1 ],                //  1 immer Material 1
  [ 1, 9, 4 ],          //  2
  [ 0 ],                //  3 Platzhalter
  [ 4, 6, 5, 9 ],       //  4
  [ 0 ],                //  5
  [ 0 ],                //  6

];
```
