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

file *showroomDesign.js*

Materials m[.] multi-materials mm[.] and colors c[.] must be defined in the file *materials.js*.

The categories should be defined in the order shown. Important e.g. for soffits.
The individual elements are inserted into the respective categories in any order
and separated by commas.

Write array brackets [ ] .
One can use [ ], as placeholder to add data later without changing the indices.

In the description, optional entries are set in ( ). These brackets must not be written.

Define your defaults for optional data ...Default = ... ;

x to the right, z to the front
y up (y0 down, y1 up)

---

lightings

lightings = [ ];

[Ambient, color, intensity], // should be available as basic lighting
[Point, x,z,y -position, color, intensity],
[Spot, x,z,y -position, x,z,y -target position, color, intensity, angle° , penumbra],
[LongLamp, x,z,y -position, color, intensity],

---

ground plan (floors and ceilings):

floors = [ ];
ceilings = [ ];

variant (a) with individual level y.
[x,z,y , ... m[i] (,[uv's])]], // uv's according to the circulation of the plan line

variant (b) with uniform level [y].
[x,z, ... [y], m[i] (,[uv's])], // uv's according to the circulation of the plan line

x,z coordinates for floor and ceiling. There are several independent areas each
possible. Points x,z of the ground plan lines (clockwise). The lines corner to
the center of gravity must be within the surface! The last point is connected to the first point.
If no uv-values are given, they are calculated internally.

---

walls (structural components, interior walls, thick partitions walls, blockings, platforms):
(without [thickness] only visible from inside!)

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

walls = [ ];

[x0,z0, (y00,y01,) x1,z1, y10,y11
(, [wd (, wall depth)]) or (, [wo, wall opening, ... (, indent))
, m[i] or mm[i] (, [uv's front])],

If the levels on the left (y00,y01,) are omitted, the levels y10,y11 apply on both sides.
several wall openings, each: distance (from the left), width, level lower edge, height
uv's front: [u0,v00,v01, u1,v10,v11] suitable to the dimensions

---

soffits IMPORTANT: define after walls!

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

soffits = [ ];

[wall index, opening index, depth, mm[.] (, 'soffit parts') (, sill thickness, sill overhang)],

parts': l left, r right, t top, b bottom, s sill

---

archways

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

archways = [ ];

variant (a)
[x0,z0, x1,z1, y0,y1, depth, mm[.] (, 'archway parts') (, sill thickness, sill overhang)],
x0,z0, x1,z1, coordinates of the center of the sides

variant (b) IMPORTANT: define after walls!
[wall index, opening index, depth, mm[.] (, sill thickness, sill overhang)],

Parts' for both variants:
f front, o opposite, a curvature, l left, r right, b bottom, s sill

---

round walls (polygon)

roundWallViewDefault = '...';

roundWalls = [ ];

[x,z -center, y0, y1, radius, radius segments, start angle°, angle°, m[.] (, 'e' or 'c')],

'e' external or 'c' centers -view

---

components

shell functions for components (height => diameter)
functions: dH[.] = h => term; def[0,1] => value[0,1]

dHdefault = h => 1; don't change, diameter constant

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

componentRotationDefault = ... ;

components = [ ];

[x,z -center, y0, y1, radial segment, height segment, diameter, mm[.] (, dH[.]) (, rotation°)],
mm[.] sequence: mantle (if not dHdefault const. only color as material), top, bottom

---

mirrors

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

mirrors = [ ];

[rectangle, x,z, y , width, height, y-rotation° (, [ti, tilt°]) (, [color])
[polygon, x,z, y, radius, n.rotation°, y-rotation° (, [ti, tilt°]) (, [color])

IMPORTANT: define after walls!
[WallRectangle, wall index, left, bottom, width, height, wall spacing (, [ti, tilt°]) (, [color])
[WallPolygon, wall index, left, bottom, radius, n.rotation°, wall spacing (, [ti, tilt°]) (, [color])

n.rotation°
integer part of the number is corner number, 3 digits behind the point is rotation of the corners

---

frames (js files in the *frames* subfolder)

frame geometry gFrame[.] defined in frames/ frames.js
(created with ConstructFrameShowroom.html)

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

frames = [ ];

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

variant (b) IMPORTANT: define after walls!
[frameID, wall index, opening index (, indent) , mm[.] (, xScale (, zScale(, yScale))],

---

Objects (3D formats: Gltf, Obj in subfolder objects3D/filename)

objects3D = [ ];

[3D-format, 'filename', x,z, y- scaling, x,z, y- position, x,z, y- rotation°],

For Obj, a .mtl with the same name must exist in the folder.

---

surrounding

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

The file names are freely selectable, the order is important.

Textures for +x,-x,+y,-y,+z,-z, in the subfolder CubeMap. Note: three.js order x,y,z

---

visitor (only 3D-format Gltf, in subfolder objects3D/filename)

visitor = ['file name', x,z, y- scaling, eye heightl, x,z- start position, y- start rotation°];

If you move with the control in the accessible areas, you can only see yourself in the mirror

---

walkable areas

walkableAreas = [ ];

[Rectangle, xMin, zMin, xMax, zMax],
[Circle, x, z, radius],
[Triangle, xa,za, xb,zb, xc,zc], *(clockwise*)

The areas should touch or overlap sufficiently to allow a passage.

M A T E R I A L S

file *materials.js*

define colors, materials, multi materials
write array brackets [ ] , optional data in ( ) - do not write these brackets

---

colors (colour scheme for use in m, use there c[.])

c = [ ]; // array of hexadecimal color values

example:

```
c = [

  0x000000,   // 0 black
  0xffffff,   // 1 white
  0xff0000,   // 2 red
  0x00ff00,   // 3 green
  0x0000ff,   // 4 blue
  0xdedede,   // 5 light gray

];
```

---

materials

m = [ ];

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

Empty - nothing is generated, usable for empty wall openings, also placeholder.
Basic or Phong or Lambert, c[.] (, Side) (, opacity)
Texture, graphic file (, 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 placeholder
  [ Video, 'Raindrops_Videvo.mp4' ],            //   9

];
```

---

multi materials

arrays of indices of the material array m[ ]

1 + openings count materials for walls with openings, index 0 for wall

6 materials for thick walls (... structural components etc.)

6 or 10 materials for archways:
front, arching, opposite, left, right, bottom or sill top, sill front, sill underside, sill left, sill right

3 materials for components:
mantle (only color as material), top, bottom

If there are too few materials in [ ], the last material is still used.

mm = [ ];

example:

```
mm = [

  [ 0 ],                //  0 always material 0 (Empty)
  [ 1 ],                //  1 always material 1
  [ 1, 9, 4 ],          //  2
  [ 0 ],                //  3 placeholder
  [ 4, 6, 5, 9 ],       //  4
  [ 0 ],                //  5
  [ 0 ],                //  6

];
```
