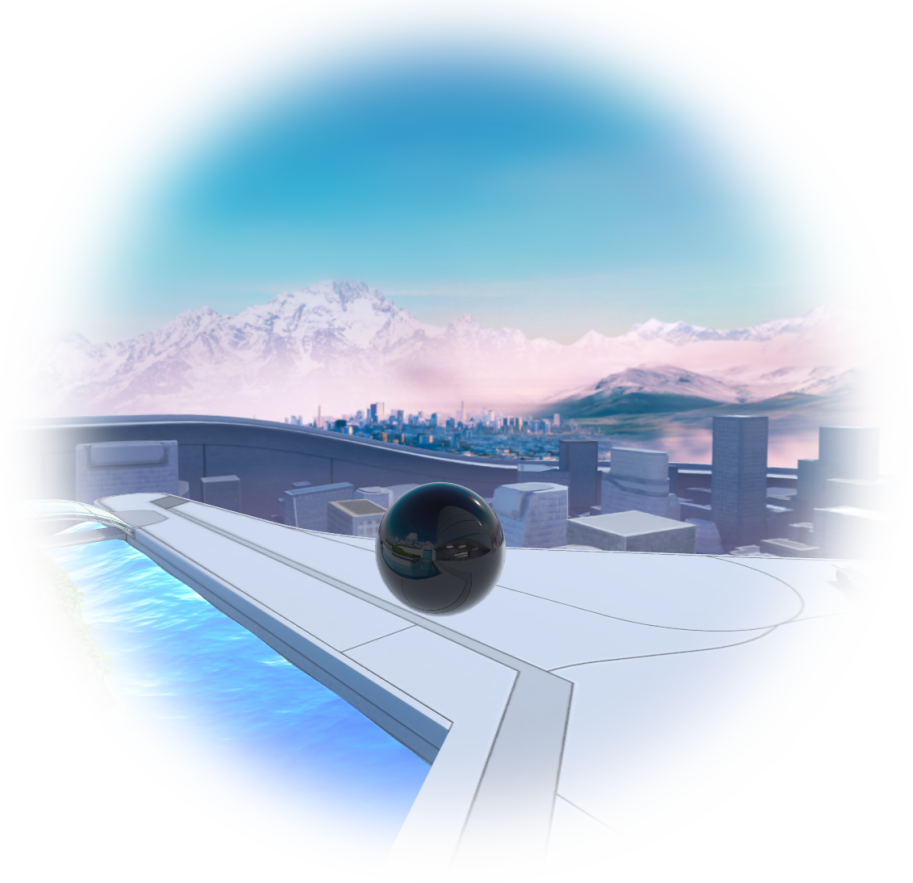

DocXR

SCRIPT REFERENCE GUIDE

Writing DocXR Scripts

DocXR is an experimental web framework with a robust scripting language for building virtual reality experiences. DocXR uses the Three.js library to display 3D computer graphics in a web browser using WebGL.

A DocXR JSON script is embedded in a Javascript module inside the body of an HTML page. A Javascript importmap needs to be inserted before the Javascript module to designate the path to the three.module.js file. There are also other Javascript files that need to be imported before beginning to write a DocXR script. The following code block shows the overall basic structure of the HTML page.

```
<html lang="en">
   <head>
      <title>DocXR</title>
      <meta charset="utf-8">
            <meta name="viewport" content="width=device-width,
              user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">
      <script src="/js/socket.io.js"></script>
   </head>
   <body>
      <script type="importmap">
        {
          "imports": {
                "three":"/js/node/node_modules/three/build/three.module.js"
          }
        }
      </script>
      <script type="module">
        import {bc} from '/js/docXR.js';
        import * as bcshaders from '/js/bcshaders.js';
        import {TableView} from '/js/docXRScript.js';
        import {VoiceChat} from  '/js/audio.js';
        var design = {
         […DOCXR SCRIPT…]
              };
         bc.control.World.newWorld(design, 'demo', (env)=>{});
      </script>
   </body>
</html>
```

Creating An Environment

An environment is created in DocXR by naming the 3D control. All concepts, assets, objects, and handlers for the demo environment will become children of the control.

```
'demo=>Environment':{
   INSERT CONCEPTS, ASSETS, CONTROL3D OBJECTS, HANDLERS
};
```

A typical environment script is arranged with common concepts appearing first, followed by all the assets, Control3D objects and their corresponding handlers. This is purely personal preference since the ordering of these items inside the environment is not required for it to display and function accordingly.

Concepts

Concepts are external JSON components that are reusable across environments. The Github repository includes common concepts for avatars, navigation, and font assets.

Custom concepts can be created for any behavior or object. You will find examples for bar chart, pie chart, dialog box, keyboard, numpad, and scroll in the client concepts directory for reference.

Provide a name for an inserted concept control and a path to the relevant JSON file using the model property. The demo environment below contains an avatar, teleport navigation, and default fonts/icons.

```
'demo=>Environment':{
   'userConcept=>Concept':{model:'concepts/avatar1.json',
    startPosition:{x:0, y:0, z:0},
    startRotation:{x:0, y:0, z:0}},
   'defaults=>Concept':{model:'concepts/defaults.json'},
   'teleport=>Concept':{model:'concepts/teleport.json'},
};
```

Adding Assets

Environment assets are automatically loaded and managed by DocXR. Each type of asset has its own requirements for how they are inserted in the script. Check the DocXR API manual in the help directory to find specific properties for all elements.

Primitives

Primitives like box, circle, cylinder, plane, sphere, and torus can be inserted into an environment using the appropriate geometry. Create a material for the geometry and then use the Mesh control to combine the geometry and material together.

```
'demo=>Environment':{
   'sphereGeo=>SphereGeometry':{radius:0.5, widthSegments:20,
      heightSegments:20},
   'sphereMtl=>StandardMaterial':{transparent:true, color:0xFF0000,
      roughness:0.0, metalness:1},
   'sphereMesh=>Mesh':{object:{scalar:0.304, position:{x:1, y:1.5, z:1}},
      geometry:'env.sphereGeo', material:'env.sphereMtl'},
};
```

Primitive geometry can also be embedded in a mesh object along with its material.

```
'demo=>Environment':{
   'boxMesh=>Mesh':{object:{position:{x:-5, y:0, z:-7}, rotation:{x:0,
      y:0.3409, z:0}},
      'geometry=>BoxGeometry':{x:0.5, y:2.2, z:0.5},
      'material=>StandardMaterial':{transparent:true, opacity:1,
         blending:1, color:0x000060, roughness:0.5, metalness:0.3,
         envMap:'env.hdrCubeMapTex'}},
};
```

3D Models

DocXR accepts Collada, FBX, GLTF, and OBJ 3D formats. An OBJ model needs textures for a material and its geometry. Different controls are used to point to these assets. The Mesh control combines the geometry and material to display the model.

```
'demo=>Environment':{
   'curtainsTex=>BasicTexture':{url:'assets/Curtains.jpg'},
   'curtainsMtl=>BasicMaterial':{map:'env.curtainsTex'},
   'curtainsObj=>Object':{url:'assets/Curtains.obj'},
   'curtainsMesh=>Mesh':{object:{scalar:0.304},
      geometry:'env.curtainsObj',
      material:'env.curtainsMtl'},
};
```

Collada, FBX, and GLTF are self-contained and need only one control for display.

```
'demo=>Environment':{
   'gltf1=>GLTF':{
      data:{clip:'animation_0'},
      url:'assets/BoxAnimated.glb',
      object:{
         position:{x:-5 , y:0.3, z:-4},
         rotation:{x:0, y:0.575, z:0},
         scalar:0.2}},
};
```

The different model mesh formats are Control3D base controls that use Object3D properties to set any position, rotation, or scale values for the 3D model.

Images

DocXR accepts JPG and PNG 2D image formats as BasicTexture controls. The ImageMesh control references the texture for display.

```
'demo=>Environment':{
   'posterTex=>BasicTexture':{url:'assets/ poster.jpg'},
   'posterMesh=>ImageMesh':{
      object:{position:{x:-.75, y:1.6, z:4.77},
      rotation:{x:0, y:3.14159, z:0}},
      map:'env.posterTex',
      originalSize:{x:864, y:1296}},
};
```

Sound

DocXR supports the MP3 audio format for the Audio3D spatial sound control.

```
'demo=>Environment':{
   'clickClip=>Audio3D':{url:'assets/sounds/click.mp3'},
};
```

Video

DocXR supports the M4V file format as its video container and uses the VideoTexture control applied to an OBJ Mesh control for display in an environment. Video controls like a play button can be added using a handler.

```
'demo=>Environment':{
   'movieObj=>Object':{url:'assets/screen_16x9.obj'},
   'movieTex=>VideoTexture':{url:'assets/testMovie.m4v', loop:true},
   'movieMesh=>Mesh':{object:{ scalar:0.304}, geometry:'env.movieObj',
      'material=>BasicMaterial':'env.movieTex'},
};
```

Fonts

DocXR supports web fonts, like Google Fonts, for the Font control. Text controls reference the different font assets to display them in an environment.

```
'demo=>Environment':{
   'Literata=>Font':{url:'assets/fonts/Literata_Book_Regular.json'},
   'KelsonSansR=>Font':{url:'assets/fonts/Kelson_Sans_Regular.json'},
   'KelsonSansB=>Font':{url:'assets/fonts/Kelson_Sans_Bold.json'},
};
```

Lights

DocXR has three types of light controls for insertion in environments. Each light has its own set of properties including color and intensity.

```
'demo=>Environment':{
   'dLight=>DirectionalLight':{color:0xffffff, intensity:1},
   'hLight=>HemisphereLight':{skycolor:0xbbbbbb, groundColor:0xbbbbbb,
      intensity:0.5},
   'pLight=>PointLight':{color:0xffffff, intensity:1, distance:0.5,
      decay:0.5},
};
```

Text

DocXR uses the TextCell control to insert text into an environment. Users can control the width, line space, and word space of the cell container, as well as the font type and size. Materials control the color of cell background and text.

```
'demo=>Environment':{
   'title=>TextCell':{object:{ position:{x:4.4, y:0, z:-0.1},
      align:'left', width:8, lineSpace:0.8, wordSpace:0.12, pad:{top:0.1,
         left:0.1}, fontSize:0.4, text:'Welcome To DocXR!',
         backMtl:'env.titleBodyMtl', fontMtl:'env.titleFontMtl',
         font:'env.defaults.body'
   },
};
```

Textures
There are 4 common types of textures for materials used by meshes in an environment.

BasicTexture
Most common type for textures that points to a JPG or PNG file.

```
'demo=>Environment':{
   'iconDownArrowTex=>BasicTexture':{url:'assets/downArrow.png'},
};
```

SmartTexture
This type of texture switches from the standard JPG or PNG file to a KTX compressed file to optimize asset load times. The KTX file must have the same name as the JPG or PNG file, and the default texture must be set for the environment.

```
'demo=>Environment':{ defaultTexture:'ktx',
   'iconDownArrowTex=>SmartTexture':{url:'assets/downArrow.png'},
};
```

VideoTexture
A special type of texture for attaching videos to meshes.

```
'demo=>Environment':{
   'targetMovie=>VideoTexture':{url:'assets/demoMovie.m4v', loop:true},
};
```

CubeMapTexture
A Cubemap is a collection of six square textures that represent the reflections on an environment. DocXR offers a high dynamic range cube map that represents a broader range of luminance levels than standard digital imaging techniques.

```
'demo=>Environment':{
   'windowTex=>SmartTexture':{url:'assets/WindowGlass.jpg'},
   'cubemapConfroom=>HDRCubeMapTexture':{url:'assets/HDRconf.bin'},
   'windowMtl=>StandardMaterial':{transparent:true, opacity:0.5,
      blending:2, color:0xbbbbbb, roughness:1, metalness:1,
      roughnessMap:'env.windowTex', envMap:'env.cubemapConfroom'},
};
```

Materials
There are 2 common types of materials for meshes in DocXR. Review the DocXR API manual to see all available properties for each material, plus other unique custom materials for reference.

```
'demo=>Environment':{
   'fontMtl=>BasicMaterial':{transparent:true, opacity:1, color:0x000000},
};
```

Basic Material
A material for drawing geometries in a simple shaded (flat or wireframe) way.

Standard Material
A standard physically based (PBR) material that will react correctly under all lighting scenarios to accurately represent real-world materials.

```
'demo=>Environment':{
   'mirrorMtl=>StandardMaterial':{
      color:0xffffff,
      aoMap:'env.mirror_ORM_Tex',
      roughnessMap:'env.mirror_ORM_Tex',
      metalnessMap:'env.mirror_ORM_Tex',
      roughness:1.0,
      metalness:1.0,
      envMap:'env.sunCubeMapTex'},
};
```

Object Transformations

All Control3D base controls can be positioned, rotated, and scaled in an environment by designating values for the X, Y, and Z axises. The scalar transform is a convenience function that allows users to only designate one value for a proportional scale.

Other properties such as renderOrder, opacity, and visibility can also be applied to the 3D Object. Reference the Object3D control in the DocXR API manual to view all available properties.

```
'demo=>Environment':{
   'tableMesh=>Mesh':{ object:{position:{x:-1, y:0, z:0.68}, rotation:{x:0,
      y:0.465025526, z:0}, scale:{x:0.304, y:0.304, z:0.304},
      geometry:'env.tableObj', material:'env.tableMtl'},

   'chairMesh=>Mesh':{object:{position:{x:-1.048, y:0, z:0.685},
      rotation:{x:0, y:0.465025526, z:0}, scalar:0.304},
      geometry:'env.chairObj', material:'env.chairMtl'},

   'sphereMesh=>Mesh':{object:{renderOrder:100, visible:true, opacity:0.5,
      position:{x:1, y:1, z:0.2}},
      'material=>BasicMaterial':{map:'env.sphereTex', side:1,
      transparent:true, opacity:1}, 'geometry=>SphereGeometry':{radius:100,
      widthSegments:64, heightSegments:20}},
};
```

Transform Units

The position transform property uses meters as units. The rotation transform property uses radians for its units. There are online tools available to convert degrees to radians (1° = 0.0174533 radian). Scale units are represented as decimal percentages.

Grouping Objects

The Group control allows users to organize different related objects in the DocXR script. It is useful for positioning a collection of objects, since all children are positioned relative to the parent group position. The example below demonstrates how a button group can be positioned in the environment while maintaining the relative positions and rotations of the child objects.

```
'demo=>Environment':{
   'deleteButton=>Group':{object:{visible:false, position:{x:1, y:1.5,
      z:0}, rotation:{y:1.5708}, scalar:0.2},
      'buttonShape=>Mesh':{object:{rotation:{x:1.5708, y:0, z:0},
         scalar:0.025},'geometry=>CylinderGeometry':{radiusTop:5,
         radiusBottom:5, height:0.5, radialSegments:36, heightSegments:1},
         'material=>StandardMaterial':{transparent:true, opacity:1,
         color:0xFF0000, roughness:0.1, metalness:0.5}},
      'iconMesh=>ImageMesh':{object:{renderOrder:50, position:{x:0, y:0,
         z:0.02}, scalar:0.8}, map:'env.iconCloseTex', transparent:true,
         opacity:1},
   },
};
```

There is also a control called Area that represents a group of objects. Just like the Group control, all Area transformations will affect all child objects.

```
'demo=>Environment':{
   'room=>Area':{object:{position:{x:0, y:0, z:0}, scalar:0.304},
      'tableMesh=>Mesh':{object:{position:{x:0, z:-1.3},
         rotation:{y:-1.5708}}, geometry:'area.tableObj',
         material:'area.tableMtl'},
      'chairMesh=>Mesh':{object:{position:{x:-0.6, z:-0.5},
         rotation:{y:1.047197}}, geometry:'area.chairObj',
         material:'area.chairMtl'},
   };
};
```

Adding Handlers

Handlers are added to Control3D objects to provide intersect event instructions in the environment. Each IntersectAction contains multiple states including onFocused, onUnfocused, onSelected, onUnselected, onGrabbed, and onIntersected. Each state contains a list of rules to execute based on the associated behavior.

A rule contains a path to an object or an object property to effect based on the state. DocXR uses dot notation to traverse the levels of the script. There are also general control terms such as current and parent that can be used to reference levels of the script. Rules are linear and execute sequentially from the first to the last entry.

Handlers are referenced using the handler property inside the Control3D object to point to the location in the script where the handler action resides. A handler can also be embedded within the Control3D object as this example shows.

```
'demo=>Environment':{
   'deleteButton=>Group':{object:{visible:false, position:{x:1, y:1.5,
      z:0}, rotation:{y:1.5708}, scalar:0.2},
      'buttonShape=>Mesh':{object:{rotation:{x:1.5708, y:0, z:0},
         scalar:0.025},'geometry=>CylinderGeometry':{radiusTop:5,
         radiusBottom:5, height:0.5, radialSegments:36, heightSegments:1},
         'material=>StandardMaterial':{transparent:true, opacity:1,
         color:0xFF0000, roughness:0.1, metalness:0.5}},
      'iconMesh=>ImageMesh':{object:{renderOrder:50, position:{x:0, y:0,
         z:0.02}, scalar:0.8}, map:'env.iconCloseTex', transparent:true,
         opacity:1},
      'intersect=>IntersectAction':{
         onFocused:[{'parent.object.scalar.doAnimation':{from:0.2, to:0.25,
            fromTime:0, toTime:100}}],
         onUnfocused:[{'parent.object.scalar.doAnimation':{from:0.25,
            to:0.2, fromTime:0, toTime:100}}],
         onSelected:[
            {'env.mapTokenHandler.active.doRemove':{}},
            {'env.mapTokenHandler.active':['empty']},
            {'current.object.visible':false},
            {'env.swish.doPlayAt':'current'},
         ],
      },
   },
};
```

Conditionals

Conditionals can be used in the various handler states to insert logic for trigger execution. The example below also includes an active variable pointer that is referenced in the conditional to check if the current object is the same as the active object.

```
'demo=>Environment':{
   'colorPickerHandler=>Handler':{'active=>Pointer':{},
      'intersect=>IntersectAction':{
         onFocused:{doIf:{value:'parent.active !== current', onTrue:[
            {'current.object.scalar.doAnimation':{from:0.05, to:0.06,
               fromTime:0, toTime:100}},
         ]}},
         onUnfocused:{doIf:{value:'parent.active !== current', onTrue:[
            {'current.object.scalar.doAnimation':{from:0.06, to:0.05,
               fromTime:0, toTime:100}},
         ]}},
         onSelected:[
            {doIf:{value:'parent.active != current', onTrue:[
               {'parent.unactivate.do':{}},
               {'parent.active':['current']},
               {'parent.active.object.scalar':0.06},
               {'current.iconOutline.object.visible':true},
               {'env.tokensPane.baseColor.value':['current.iconButton
                  .material.color']},
               {'env.tokensPane.hiliteColor.value':['current.iconButton
                  .material.emissive']},
               {'env.tokensPane.redraw.do':{}},
            ], onFalse:[{'parent.unactivate.do':{}}]}},
         ]},
         'unactivate=>Action':{
            onAction:[
               {doIf:{value:'parent.active != empty', onTrue:[
                  {'parent.active.object.scalar.doAnimation':{from:0.06,
                     to:0.05, fromTime:0, toTime:100}},
                  {'parent.active.iconOutline.object.visible':false},
                  {'parent.active':['empty']},
               ]}},
            ],
         },
      },
   },
};
```

Adding Animations
Animations can be added to DocXR scripts using either an inline method or an effect container. The inline method uses the doAnimation control in each handler rule. Keyframes are denoted by using the from and to properties. The animation duration is denoted using the fromTime and toTime properties.

The example below uses inline animation for a button. The button scales larger when a rollover occurs and returns to the original scale on rollout. Clicking the button causes the button to scale up and then down in 100 milliseconds.

```
'demo=>Environment':{
   'exitButtonHandler=>Handler':{
      'intersect=>IntersectAction':{
         onFocused:[
            {'current.object.scalar.doAnimation':{from:0.6, to:0.8,
               fromTime:0, toTime:50}},
            {'env.exitButtonFontMtl.color':0x00c3d8},
         ],
         onUnfocused:[
            {'current.object.scalar.doAnimation':{from:0.8, to:0.6,
               fromTime:0, toTime:50}},
            {'env.exitButtonFontMtl.color':0xffffff},
         ],
         onSelected:[
            {'current.object.scalar.doAnimation':{from:0.8, to:0.6,
               fromTime:0, toTime:50}},
            {'current.object.scalar.doAnimation':{from:0.6, to:0.8,
               fromTime:50, toTime:100}},
         ],
      },
   },
};
```

The example below uses the Effect control as a staging area for the animations. It is referenced in the handler rules along with a doAnimate control. Multiple object animations can use the control, which makes it a more flexible approach when dealing with more complex animations.

```
'demo=>Environment':{
   'effect=>Effect':{},
   'buttonMesh=>Mesh':{'geometry=>CircleGeometry':{radius:.3, segments:50},
      material:['buttonShader'],
      'intersect=>IntersectAction':{
         onFocused:[
            {'env.effect.doAnimate':{value:'current.object.scalar',
               from:0.6, to:0.8, fromTime:0, toTime:50}}
         ],
         onUnfocused:[
            {'env.effect.doAnimate':{value:'current.object.scalar',
               from:0.8, to:0.6, fromTime:0, toTime:50, ease:3}}
         ],
         onSelected:[
            {'env.effect.doAnimate':{value:'current.object.scalar',
               from:0.8, to:0.6, fromTime:0, toTime:50}},
            {'env.effect.doAnimate':{value:'current.object.scalar',
               from:0.6, to:0.8, fromTime:50, toTime:100, ease:3}},
         ],
      },
   },
};
```

Six different animation easing functions can also be inserted using the ease property. This effects the rate, or speed of an animation.

Ease Value	Type
1	easeInQuad
2	easeInCubic
3	easeOutQuad
4	easeOutCubic
5	easeInOutQuad
6	easeInOutCubic

Adding Data Sources
Data sources can be injected into environments using an inline embedded format or a referenced external comma or tab delimited file.

```
'demo=>Environment':{
   'slidesSource=>JSONSource':{data:[
      {slideIndex:'01', slideURL:'assets/slides/slide-01.jpg'},
      {slideIndex:'02', slideURL:'assets/slides/slide-02.jpg'},
      {slideIndex:'03', slideURL:'assets/slides/slide-03.jpg'},
      {slideIndex:'04', slideURL:'assets/slides/slide-04.jpg'},
      {slideIndex:'05', slideURL:'assets/slides/slide-05.jpg'},
      {slideIndex:'06', slideURL:'assets/slides/slide-06.jpg'},
      {slideIndex:'07', slideURL:'assets/slides/slide-07.jpg'},
      {slideIndex:'08', slideURL:'assets/slides/slide-08.jpg'},
      {slideIndex:'09', slideURL:'assets/slides/slide-09.jpg'},
      {slideIndex:'10', slideURL:'assets/slides/slide-10.jpg'},
   ]},
   'slideDataSource':{url:'assets/data/slides.csv'},
};
```

DocXR loads the data into a BasicTable control. A record is built for each row using the referenced data source.

```
'demo=>Environment':{
   'slideDataSource':{url:'assets/data/slides.csv'},
   'slideRecords=>BasicTable':{
      'record=>Record':{
         slideIndex=>StringField':{order:0},
         slideURL=>StringField':{order:1},
      },
      source:'env.slideDataSource'
   },
};
```

DocXR uses the TableView control to display data from the table in a virtual spreadsheet. The TextCell control is then used to set the padding, width and font size for the text field, as well as the background material for the cell, and the font type and color for display.

```
'demo=>Environment':{
   'slideDataSource':{url:'assets/data/slides.csv'},
   'slideRecords=>BasicTable':{
      'record=>Record':{
         slideIndex=>StringField':{order:0},
         slideURL=>StringField':{order:1},
      },
      source:'env.slideDataSource'
   },
   'view=>TableView':{maxFields:2, rowMin:5, rowMax:10,
      table:'env.slideRecords',
      onBuild:[
         {'current.zone.isValue':{
            field:[{'current.field.id.isValue':{
               'slideIndex':{'current.cell=>TextCell':{
               align:'left', width:2, lineSpace:0.5, pad:{top:0.1,
               left:0.1}, fontSize:0.1, text:['current.value'],
               'backMtl=>StandardMaterial':{transparent:true,
               emissiveIntensity:1.0, emissive:0x46c0e5, metalness:1.0,
               color:0x000000, roughness:0.5},
               'fontMtl=>BasicMaterial':{transparent:true, color:0xFFFFFF},
               font:'env.defaults.body',
            }},
            'slideURL':{'current.cell=>TextCell':{
               align:'left', width:1.8, lineSpace:0.5, pad:{top:0.1,
               left:0.1}, fontSize:0.1, text:['current.value'],
               'backMtl=>BasicMaterial':{transparent:true, color:0xFFFFFF},
               'fontMtl=>BasicMaterial':{transparent:true, color:0x000000},
               font:'env.defaults.body',
            }},
         }}],
      }}],
   },
};
```