glTF-Tutorials

Previous: Materials | Table of Contents | Next: Textures, Images, Samplers

A Simple Material

The examples of glTF assets that have been given in the previous sections contained a basic scene structure and simple geometric objects. But they did not contain information about the appearance of the objects. When no such information is given, viewers are encouraged to render the objects with a “default” material. And as shown in the screenshot of the minimal glTF file, depending on the light conditions in the scene, this default material causes the object to be rendered with a uniformly white or light gray color.

This section will start with an example of a very simple material and explain the effect of the different material properties.

This is a minimal glTF asset with a simple material:

```
{
  "scene": 0,
  "scenes" : [
    {
      "nodes" : [ 0 ]
    }
  ],

  "nodes" : [
    {
      "mesh" : 0
    }
  ],

  "meshes" : [
    {
      "primitives" : [ {
        "attributes" : {
          "POSITION" : 1
        },
        "indices" : 0,
        "material" : 0
      } ]
    }
  ],

  "buffers" : [
```

```
    {
      "uri" : "data:application/octet-stream;base64,AAABAAIAAAAAAAAAAAAAAAAAAAIA/AAAAA
      "byteLength" : 44
    }
  ],
  "bufferViews" : [
    {
      "buffer" : 0,
      "byteOffset" : 0,
      "byteLength" : 6,
      "target" : 34963
    },
    {
      "buffer" : 0,
      "byteOffset" : 8,
      "byteLength" : 36,
      "target" : 34962
    }
  ],
  "accessors" : [
    {
      "bufferView" : 0,
      "byteOffset" : 0,
      "componentType" : 5123,
      "count" : 3,
      "type" : "SCALAR",
      "max" : [ 2 ],
      "min" : [ 0 ]
    },
    {
      "bufferView" : 1,
      "byteOffset" : 0,
      "componentType" : 5126,
      "count" : 3,
      "type" : "VEC3",
      "max" : [ 1.0, 1.0, 0.0 ],
      "min" : [ 0.0, 0.0, 0.0 ]
    }
  ],

  "materials" : [
    {
      "pbrMetallicRoughness": {
        "baseColorFactor": [ 1.000, 0.766, 0.336, 1.0 ],
        "metallicFactor": 0.5,
        "roughnessFactor": 0.1
      }
    }
  ],
  "asset" : {
    "version" : "2.0"
```

```
    }
  }
```

When rendered, this asset will show the triangle with a new material, as shown in Image 11a.

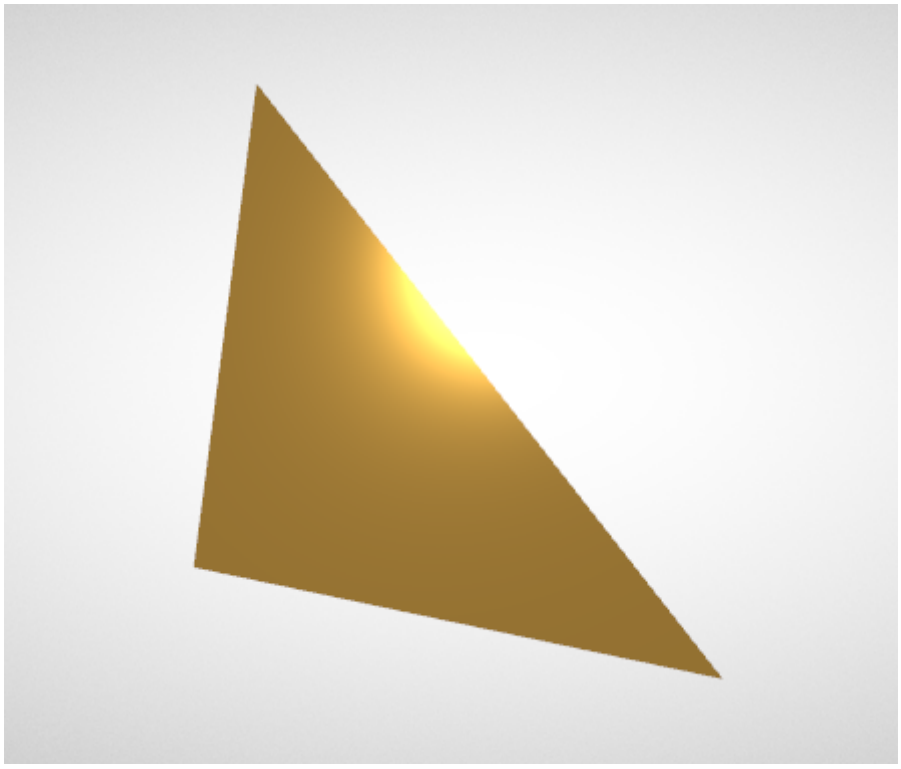

Image 11a: A triangle with a simple material.

Material definition

A new top-level array has been added to the glTF JSON to define this material: The `materials` array contains a single element that defines the material and its properties:

```
  "materials" : [
    {
      "pbrMetallicRoughness": {
        "baseColorFactor": [ 1.000, 0.766, 0.336, 1.0 ],
        "metallicFactor": 0.5,
        "roughnessFactor": 0.1
      }
    }
  ],
```

The actual definition of the `material` here only consists of the `pbrMetallicRoughness` object, which defines the basic properties of a material in the *metallic-roughness-model*. (All other material properties will therefore have default values, which will be explained later.) The `baseColorFactor` contains the red, green, blue, and alpha components of the main color of the material - here, a bright orange color. The `metallicFactor` of 0.5 indicates that the material should have reflection characteristics between that of a metal and a non-metal material. The

`roughnessFactor` causes the material to not be perfectly mirror-like, but instead scatter the reflected light a bit.

Assigning the material to objects

The material is assigned to the triangle, namely to the `mesh.primitive`, by referring to the material using its index:

```
"meshes" : [
  {
    "primitives" : [ {
      "attributes" : {
        "POSITION" : 1
      },
      "indices" : 0,
      "material" : 0
    } ]
  }
```

The next section will give a short introduction to how textures are defined in a glTF asset. The use of textures will then allow the definition of more complex and realistic materials.

Previous: Materials	Table of Contents	Next: Textures, Images, Samplers