glTF-Tutorials

Previous: Cameras | Table of Contents | Next: Morph Targets

A Simple Morph Target

Starting with version 2.0, glTF supports the definition of *morph targets* for meshes. A morph target stores displacements or differences for certain mesh attributes. At runtime, these differences may be added to the original mesh, with different weights, in order to animate parts of the mesh. This is often used in character animations, for example, to encode different facial expressions of a virtual character.

The following is a minimal example that shows a mesh with two morph targets. The new elements will be summarized here, and the broader concept of morph targets and how they are applied at runtime will be explained in the next section.

```
{
  "scene": 0,
  "scenes":[
    {
      "nodes":[
        0
      ]
    }
  ],
  "nodes":[
    {
      "mesh":0
    }
  ],
  "meshes":[
    {
      "primitives":[
        {
          "attributes":{
            "POSITION":1
          },
          "targets":[
            {
              "POSITION":2
            },
            {
              "POSITION":3
            }
          ],
```

```
          "indices":0
        }
      ],
      "weights":[
        0.5,
        0.5
      ]
    }
  ],

  "animations":[
    {
      "samplers":[
        {
          "input":4,
          "interpolation":"LINEAR",
          "output":5
        }
      ],
      "channels":[
        {
          "sampler":0,
          "target":{
            "node":0,
            "path":"weights"
          }
        }
      ]
    }
  ],

  "buffers":[
    {
      "uri":"data:application/gltf-buffer;base64,AAABAAIAAAAAAAAAAAAAAAAAAAAAIA/AAAAAAAA
      "byteLength":116
    },
    {
      "uri":"data:application/gltf-buffer;base64,AAAAAAAAgD8AAABAAABAQAAAgEAAAAAAAAAAAAAA
      "byteLength":60
    }
  ],
  "bufferViews":[
    {
      "buffer":0,
      "byteOffset":0,
      "byteLength":6,
      "target":34963
    },
    {
      "buffer":0,
      "byteOffset":8,
      "byteLength":108,
      "byteStride":12,
```

```
      "target":34962
    },
    {
      "buffer":1,
      "byteOffset":0,
      "byteLength":20
    },
    {
      "buffer":1,
      "byteOffset":20,
      "byteLength":40
    }
  ],
  "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,
        0.5,
        0.0
      ],
      "min":[
        0.0,
        0.0,
        0.0
      ]
    },
    {
      "bufferView":1,
      "byteOffset":36,
      "componentType":5126,
      "count":3,
      "type":"VEC3",
      "max":[
        0.0,
        1.0,
```

```
        0.0
      ],
      "min":[
        -1.0,
        0.0,
        0.0
      ]
    },
    {
      "bufferView":1,
      "byteOffset":72,
      "componentType":5126,
      "count":3,
      "type":"VEC3",
      "max":[
        1.0,
        1.0,
        0.0
      ],
      "min":[
        0.0,
        0.0,
        0.0
      ]
    },
    {
      "bufferView":2,
      "byteOffset":0,
      "componentType":5126,
      "count":5,
      "type":"SCALAR",
      "max":[
        4.0
      ],
      "min":[
        0.0
      ]
    },
    {
      "bufferView":3,
      "byteOffset":0,
      "componentType":5126,
      "count":10,
      "type":"SCALAR",
      "max":[
        1.0
      ],
      "min":[
        0.0
      ]
    }
  ],
```

```
  "asset":{
    "version":"2.0"
  }
}
```

The asset contains an animation that interpolates between the different morph targets for a single triangle. A screenshot of this asset is shown in Image 17a.

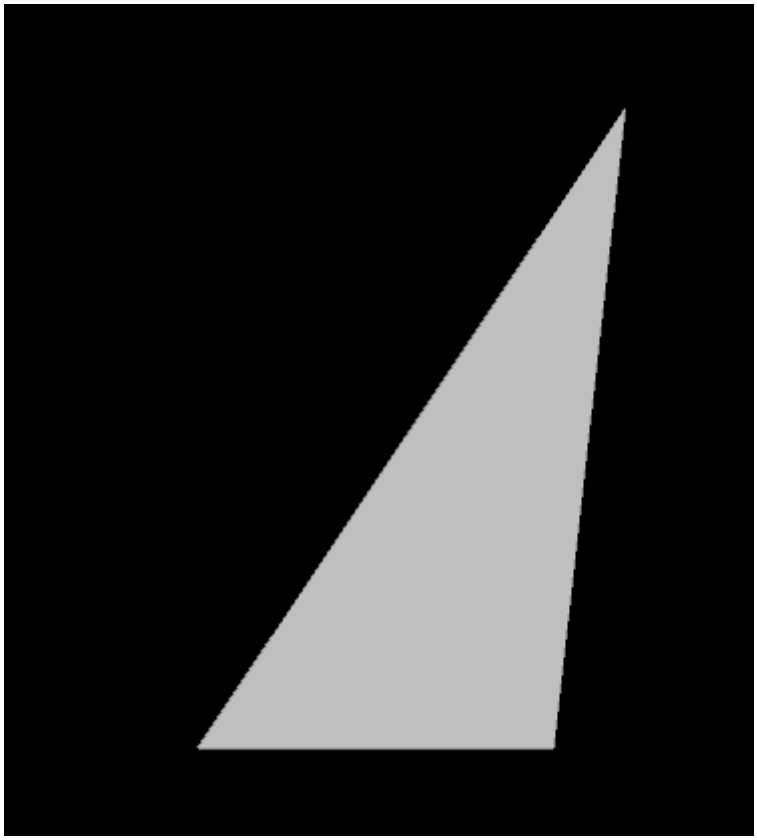

Image 17a: A triangle with two morph targets.

Most of the elements of this asset have already been explained in the previous sections: It contains a `scene` with a single `node` and a single `mesh`. There are two `buffer` objects, one storing the geometry data and one storing the data for the `animation`, and several `bufferView` and `accessor` objects that provide access to this data.

The new elements that have been added in order to define the morph targets are contained in the `mesh` and the `animation`:

```
"meshes":[
  {
    "primitives":[
      {
        "attributes":{
          "POSITION":1
        },
        "targets":[
          {
```

```
            "POSITION":2
          },
          {
            "POSITION":3
          }
        ],
        "indices":0
      }
    ],
    "weights":[
      0.5,
      0.5
    ]
  }
],
```

The `mesh.primitive` contains an array of morph `targets`. Each morph target is a dictionary that maps attribute names to `accessor` objects. In the example, there are two morph targets, both mapping the `"POSITION"` attribute to accessors that contain the morphed vertex positions. The mesh also contains an array of `weights` that defines the contribution of each morph target to the final, rendered mesh. These weights are also the `channel.target` of the `animation` that is contained in the asset:

```
"animations":[
  {
    "samplers":[
      {
        "input":4,
        "interpolation":"LINEAR",
        "output":5
      }
    ],
    "channels":[
      {
        "sampler":0,
        "target":{
          "node":0,
          "path":"weights"
        }
      }
    ]
  }
],
```

This means that the animation will modify the `weights` of the mesh that is referred to by the `target.node`. The result of applying the animation to these weights, and the computation of the final, rendered mesh will be explained in more detail in the next section about Morph Targets.

Previous: Cameras	Table of Contents	Next: Morph Targets