POLITECNICO
MILANO 1863

Loading an External 3D Model

Computer Graphics 2020

Erica Stella (erica.stella@polimi.it)

Let's get rid of the boring cubes!

In real applications the rendered models are complex models, usually represented as 3D triangular meshes with material properties:

- Texture (+uv mapping)
- Diffuse color
- Specular color

Load meshes from external files

Different file formats to represent meshes:

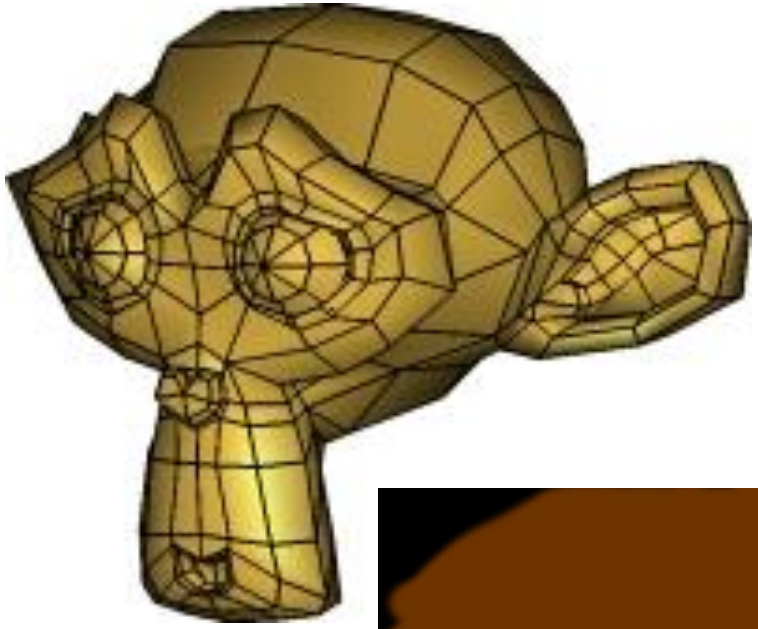
- OFF
- PLY
- STL
- …

How to load them in our javascript code?

- Create a parser for each type of file
  - Not feasible in this course
- Use an external tool to convert them into an easily readable JSON file
- Use an external library to load .obj files

Susan JSON file

```
"meshes": [ {
      "name": "Suzanne"
     ,"materialindex": 0
     ,"primitivetypes": 4
     ,"vertices": [ //list of coordinates of vertices
      0.46875
     ,-0.757813
[...]
"faces": [//indices of the vertices composing faces
[
 0
,1
,2
]
[...]
"texturecoords": [//list of uv coordinates of vertices
[
 0.943088
,0.229138
,0.94343
[...]
```



Susan JSON file

```
"meshes": [ {
     "name": "Suzanne"
    ,"materialindex": 0
    ,"primitivetypes": 4
    ,"vertices": [...]
    ,"faces":[...]
    ,"texturecoords": [...]
    //There might also be normals
    ,"normals": [...]}
```

```
//…or more meshes!!
"meshes": [
        {…},
        {…},//mesh 2
        {…} //mesh 3
        ]
```

```
 "name": "Suzanne"
,"materialindex": 0
,"primitivetypes": 4
,"vertices": [...]
,"faces":[...]
,"texturecoords": [...]
,"normals": [...]}
```

//mesh 1

When more meshes are present in a single JSON file remember **you have to draw each one separately** (possibly with different textures)

Import the JSON model into your javascript file

```
var susanModel;

await utils.get_json(pathToModel,
                     function(loadedModel){susanModel = loadedModel;});


var susanVertices = susanModel.meshes[0].vertices;
var susanIndices = [].concat.apply([], susanModel.meshes[0].faces);
var susanTexCoords = susanModel.meshes[0].texturecoords[0];
```

UV Mapping with the loaded model

```
var texture = gl.createTexture();
// use texture unit 0
gl.activeTexture(gl.TEXTURE0);
// bind to the TEXTURE_2D bind point of texture unit 0
gl.bindTexture(gl.TEXTURE_2D, texture);

var image = new Image();
image.src = baseDir+texturePath;

image.onload= function() {
  gl.bindTexture(gl.TEXTURE_2D, texture);
  gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA,
                gl.RGBA, gl.UNSIGNED_BYTE, image);
  gl.generateMipmap(gl.TEXTURE_2D);
};
```

OBJ files pt1

```
o pig_mechanic_ID240.026
v -0.272493 -0.042941 0.082401 //vertex positions
[...]
vt 0.042342 0.478492 //vertex uv coordinates
[...]
vn 0.1737 0.9398 -0.2943 //vertex normals
[...]
f 916/916/916 914/914/914 996/996/996 //faces in the format vertexIndex/uvIndex/normalsIndex for the
[...]                                  three vertices composing them
```

We load .obj files with the library webgl-obj-loader.min.js

In the html file add:

```
<script type="text/javascript" src="webgl-obj-loader.min.js"></script>
```

OBJ files pt2

- In the javascript file:

```
var objStr = await utils.get_objstr(pathToModel);
var objModel = new OBJ.Mesh(objStr);

var modelVertices = objModel.vertices;
var modelNormals = objModel.normals;
var modelIndices = objModel.indices;
var modelTexCoords = objModel.textures;
```