Interactive Graphics Systems

Pictures: two details of HCI from movie T*he Dark Knight (credits BLIND, Ltd.)*

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Dealing with material processing

Alexandre Valle | DEI | FEUP

Reading material data…

```
// assuming this.reader is an object of CGFXMLreader();
// example of simple javascript object instanciation and assignment
var mat = { id: null, shininess:0,
specular:{r:0,g:0,b:0,a:0}, diffuse:{r:0,g:0,b:0,a:0},
ambient:{r:0,g:0,b:0,a:0}, emission:{r:0,g:0,b:0,a:0} };


// assuming variable mat is assigned and id
// example on setting the specular red component, read from the atribute r
of xml element xmlElem
mat.specular.r = this.reader.getFloat(xmlElem,'r',1);
```

From material data to CGFappearance

```
// assuming this has the scope of a class extending CGFscene and mat is an object
describing a material
var appearence = new CGFappearance(this);
appearence.setShininess(mat.shininess);
appearence.setSpecular(mat.specular.r,mat.specular.g,mat.specular.b,mat.specular.a);
appearence.setDiffuse(mat.diffuse.r,mat.diffuse.g,mat.diffuse.b,mat.diffuse.a);
appearence.setAmbient(mat.ambient.r,mat.ambient.g,mat.ambient.b,mat.ambient.a);
appearence.setEmission(mat.emission.r,mat.emission.g,mat.emission.b,mat.emission.a);

// assuming mat.id is not null
// preserve appearance in an array of CGFappearences (array requires instantiation)
this.appearences[mat.id] = appearence;
```

The main section of MyScene…

```
// assuming this has the scope of a class extending CGFScene and this.root is the class attribute holding the root
component


// set a default appearance

this.setAmbient(0.1, 0.1, 0.1, 1.0);

this.setDiffuse(0.2, 0.4, 0.8, 1.0);

this.setSpecular(0.2, 0.4, 0.8, 1.0);

this.setShininess(10.0);


…

// call the draw of the entire scene graph...

// second argument with null value means no previous material id provided

this.drawComponent(this.root, null,…);
```

drawComponent method…

```
// assuming this has the scope of a class extending CGFScene
// assuming objects of class MyComponent have an attribute m holding the component's transformation matrix
// assuming further non-null validations are further required in the code
MyScene.prototype.drawComponent = function(currNode, prevAppearenceId,…) {
            // the following is an illustration of a logic that selects a material id
            // you will have to modify this logic depending on this year work requirements towards materials!
            var id = (currNode.mat.id !== null ? currNode.mat.id : prevAppearenceId)

            for(var i = 0; i < currNode.children.length ;i++) { // assuming children refers ONLY to child components
                        // recursively visit the next child component, passing resolved material id
                        this.drawComponent(currNode.children[i], id, …);
            }
            // retrieve the CGFappearence based on resolved material id
            var currAppearence = this.appearances[id]
             // set the active material. IF TEXTURES ARE PRESENT, THE APPLY MUST BE PERFORMED AFTER THE currAppearence.setTexture
            currAppearence.apply();
            for(var i = 0; i < currNode.primitives.length ;i++) { // assuming primitives refers to this component's primitives
                        // call each primitive display method
            }
}
```