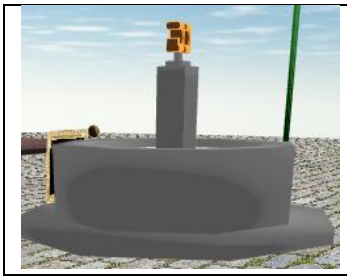
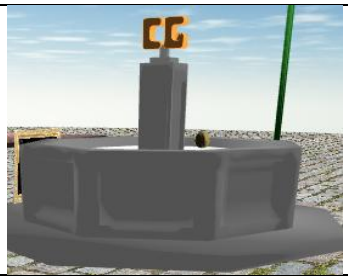
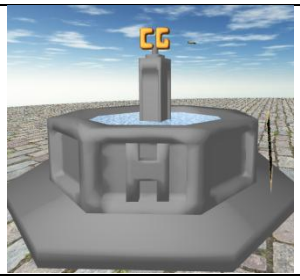
Special Effect: Level of Detail

Introduction

As the second special effect for our movie, we chose “Level of Detail”. In order to fulfill the certain specifications, it was necessary to provide three different detail levels of geometry for a certain object. In our movie we performed this with the well model. The different models have been designed in Wings3D:

		
low detail level	medium detail level	high detail level

These different models appear during the camera flight in respect to the distance of the camera to the well. If the camera is far away, it doesn’t make sense to display the high detail level. On the other hand, if the camera is close to the well, the low detailed level is to less since we are able to see some more detailed geometry structures of the well.

Algorithm Description

Since it is not recommended to setup the different models at render time, we want to preconfigure the models only once at initial time:

This is done by the function `setUpWellLevelOfDetail()`, which loads all level of detail models at startup time just once. After that, we can access the different and complete models at render time arbitrary.

```
{
    let well2 = new MaterialSGNode([
        new RenderSGNode(resources.modelWell2)
    ]);

    well2.ambient = [0.8, 0.8, 0.8, 1];
    well2.diffuse = [0.75164, 0.75164, 0.75164, 1];
    well2.specular = [0.1, 0.1, 0.1, 1];
    well2.shininess = 0.4;

    wellUpsideDown2 = new TransformationSGNode(mat4.create(), [
        new TransformationSGNode(glm.transform({ translate: [0, 0, 0], rotateX : 180, scale: 0.8 }),  [
            well2
        ])
    ]);
}
```

Reference to the .obj file

Global reference which can be accessed at render time

At render time, we have to swap the different level of detail models with respect to the camera distance and the well. Distance calculation is done by `calculateCameraToWellDistance()`:

```
function calculateCameraToWellDistance(){
    xDist = camera.position.x - well.position.x;
    yDist = camera.position.y - well.position.y;
    zDist = camera.position.z - well.position.z;
    return Math.sqrt(Math.pow(xDist, 2) + Math.pow(yDist, 2) + Math.pow(zDist, 2));
}
```

At first this function calculates the distance vector between the camera and the well like:

$$\begin{pmatrix} cameraPosition\ x \\ cameraPosition\ y \\ cameraPosition\ z \end{pmatrix} - \begin{pmatrix} wellPosition\ x \\ wellPosition\ y \\ wellPosition\ z \end{pmatrix} = \text{Distancevector}$$

After this calculation, we finally have to compute the length of this vector with the formula:

$$Cameradistance\ to\ well = \sqrt{(distancevector\ x)^2 + (distancevector\ y)^2 + (distancevector\ z)^2}$$

This calculation is done in every render step, therefore we can decide, which model has to be loaded:

```
//special effect: check well's level of detail
if(cameraToWellDistance >= 5.0 && cameraToWellDistance <= 6.0)
    setWellLevelOfDetail(3);
else if(cameraToWellDistance >= 10.0 && cameraToWellDistance <= 11.0)
    setWellLevelOfDetail(2);
else if(cameraToWellDistance >= 15.0 && cameraToWellDistance <= 16.0)
    setWellLevelOfDetail(1);
```

In order to avoid setting the new model in every render step, even it is not necessary, the check is only performed in certain intervals.

The function `setWellLevelOfDetail()` does the swapping mechanism:

```
//set current level of detail well geometry
function setWellLevelOfDetail(level){
    if(wellLevelToSet != null)
        wellMainNode.remove(wellLevelToSet);

    if(level == 1){
        wellLevelToSet = wellUpsideDown1;
    }else if(level == 2){
        wellLevelToSet = wellUpsideDown2;
    }else if(level == 3) {
        wellLevelToSet = wellUpsideDown3;
    }

    wellMainNode.append(wellLevelToSet);
}
```