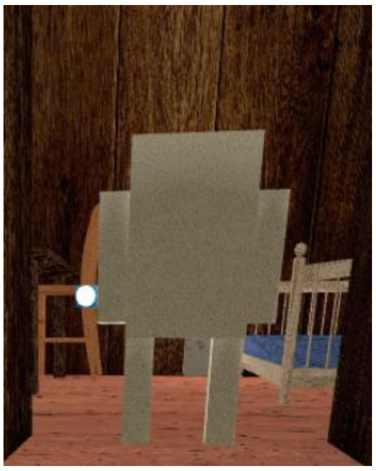


Basic Movie Effects

Composed Model

The Composed Model has several different Parts, each of those has an own transformationMatice, which allows individual movements.
(During the animation, the model is picking up the flashlight, the arms are swinging and the composed Model itself is moving)

```
////Body Transformation
var rightArmTransformationMatrix;
var leftArmTransformationMatrix;
var rightLegTransformationMatrix;
var leftLegTransformationMatrix;
var isFlashlightPickedUp = false;
var figureTransformationNode;
var armRotationX = 90;
var bodyRotationY = 0;
//// Flashlight
var flashLightTransformationNode;
var flashLightRotation


function renderBody(timeInMilliseconds){
  relativeTimeInMilliseconds = timeInMilliseconds%30000;
  if(between(relativeTimeInMilliseconds, 6000,15000)){
    isFlashlightPickedUp = true;
  } else {
    isFlashlightPickedUp = false;
  }

  var animation = getBetweenAnimation(relativeTimeInMilliseconds);
  setFigureTranslation(animation, relativeTimeInMilliseconds);
  setFigureRotation(animation, relativeTimeInMilliseconds);
  setFlashLightTranslation(animation, relativeTimeInMilliseconds);
  swingArm(relativeTimeInMilliseconds);

}
```

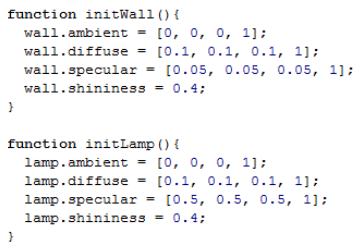
Material

We set MaterialSGNodes as general Variables or load Models from the resources (for example):

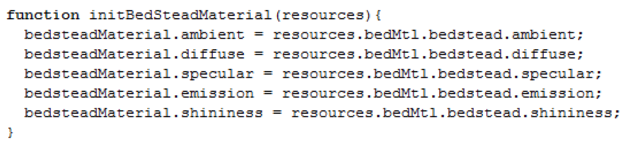
```
////General
var wall = new MaterialSGNode([new RenderSGNode(makeRect())]);
var lamp = new MaterialSGNode([new RenderSGNode(makeLamp())]);
var cuboid = new MaterialSGNode([new RenderSGNode(makeCuboid())]);
```

```
function setMaterials(resources){
    deskMaterial = new MaterialSGNode(new RenderSGNode(resources.table));
    chairMaterial = new MaterialSGNode(new RenderSGNode(resources.chair));
    ceilLampMaterial = new MaterialSGNode(new RenderSGNode(resources.ceilingLamp));
    bedMattressMaterial = new MaterialSGNode(new RenderSGNode(resources.bedMattress));
    bedsteadMaterial = new MaterialSGNode(new RenderSGNode(resources.bedstead));
    toiletMaterial=new MaterialSGNode(new RenderSGNode(resources.toilet));
    sinkMaterial=new MaterialSGNode(new RenderSGNode(resources.sink));
    frezzerMaterial=new MaterialSGNode(new RenderSGNode(resources.frezzer));
    tabMaterial = new MaterialSGNode(new RenderSGNode(resources.tab));
}
```

Afterwards within the init Funktion of the Scene, we initialize the nodes with the corresponding Material. The Material is either set manually or stored within the resources (example):



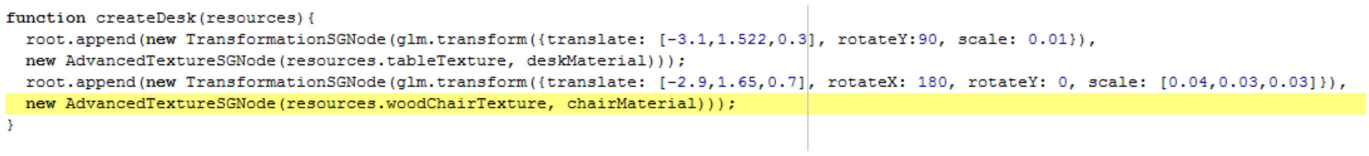
```
function initWall(){
  wall.ambient = [0, 0, 0, 1];
  wall.diffuse = [0.1, 0.1, 0.1, 1];
  wall.specular = [0.05, 0.05, 0.05, 1];
  wall.shininess = 0.4;
}

function initLamp(){
  lamp.ambient = [0, 0, 0, 1];
  lamp.diffuse = [0.1, 0.1, 0.1, 1];
  lamp.specular = [0.5, 0.5, 0.5, 1];
  lamp.shininess = 0.4;
}
```



```
function initBedSteadMaterial(resources){
  bedsteadMaterial.ambient = resources.bedMtl.bedstead.ambient;
  bedsteadMaterial.diffuse = resources.bedMtl.bedstead.diffuse;
  bedsteadMaterial.specular = resources.bedMtl.bedstead.specular;
  bedsteadMaterial.emission = resources.bedMtl.bedstead.emission;
  bedsteadMaterial.shininess = resources.bedMtl.bedstead.shininess;
}
```

Texturing

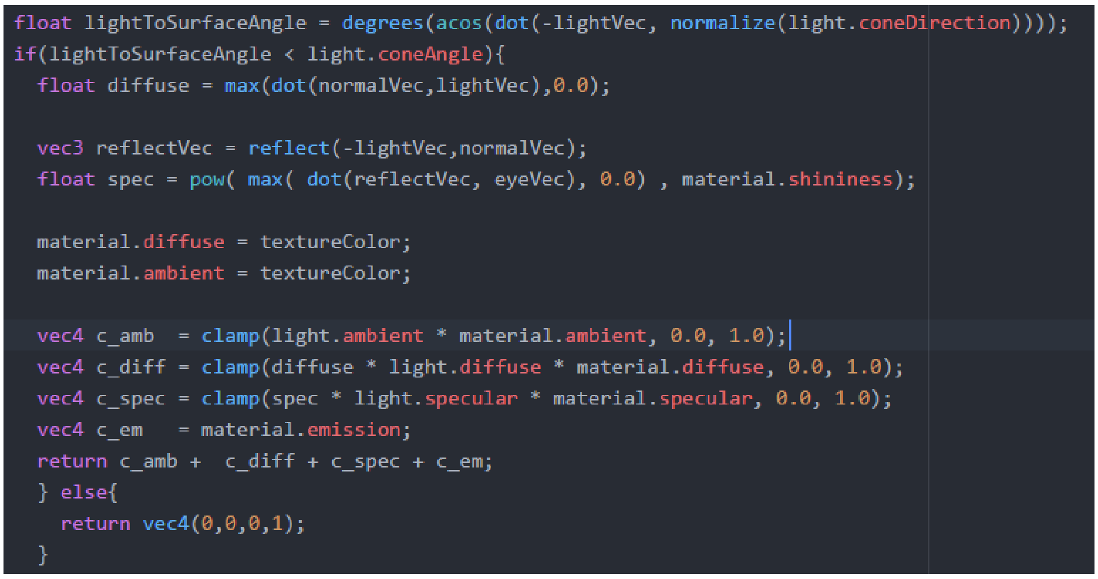
The Textures will be loaded in the load Resource function, afterwards we can apply the Textures to by creating an AdvancedTexturesSGNode.
(all nodes except the water and the kitchen Table have a Texture applied with AdvancedTexturesSGNode)



```
function createDesk(resources){
  root.append(new TransformationSGNode(glm.transform({translate: [-3.1,1.522,0.3], rotateY:90, scale: 0.01}),
  new AdvancedTextureSGNode(resources.tableTexture, deskMaterial)));
  root.append(new TransformationSGNode(glm.transform({translate: [-2.9,1.65,0.7], rotateX: 180, rotateY: 0, scale: [0.04,0.03,0.03]}),
  new AdvancedTextureSGNode(resources.woodChairTexture, chairMaterial)));
}
```

Illumination

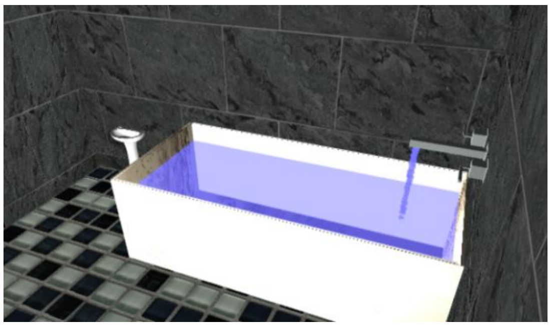
For the spot light we check the angle between the fragment and the Light Source and the direction Vector of the Spot light. If the angle is smaller, that the parameter coneAngle, normal phong shading is applied, otherwise we do apply any shading.



```
float lightToSurfaceAngle = degrees(acos(dot(-lightVec, normalize(light.coneDirection))));
if(lightToSurfaceAngle < light.coneAngle){
  float diffuse = max(dot(normalVec,lightVec),0.0);

  vec3 reflectVec = reflect(-lightVec,normalVec);
  float spec = pow( max( dot(reflectVec, eyeVec), 0.0) , material.shininess);

  material.diffuse = textureColor;
  material.ambient = textureColor;

  vec4 c_amb  = clamp(light.ambient * material.ambient, 0.0, 1.0);
  vec4 c_diff = clamp(diffuse * light.diffuse * material.diffuse, 0.0, 1.0);
  vec4 c_spec = clamp(spec * light.specular * material.specular, 0.0, 1.0);
  vec4 c_em   = material.emission;
  return c_amb +  c_diff + c_spec + c_em;
  } else{
    return vec4(0,0,0,1);
  }
```

Transparency

All water within the movie have a fixed color and transparency. Which is set within the watercolor shader.
(The nodes, which support transparency are part of the bathtub and the particle System).

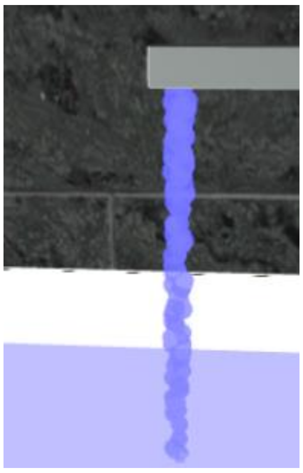


Special Effects

Particle System

Initialize the Particle System:

```
function initParticleSystem(resources){
  particleSystem = new ParticleSystem(resources);
}
```



The Particle System contains a position, which describes the origin of all particles as well as an Array, which contain all generated particles. (We choose to render Spheres instead of triangles since a sphere is more fitting for describing water)

```
//init function for the Particle system
function ParticleSystem (resources){
  //set origin of the Particle Sytem
  this.originX =0.78;
  this.originY=1.38;
  this.originZ=1.7;
  this.particles = new Array(); //storage for particles
  this.active=true;  // is particle system active
```

The particle itself contains a life and a transformationmatrice. Each time a particle moves we reduce the life by one. If the life reaches 0 we will refresh the particle with a new life and the tranformationmatrice will be reset at the origin.

```
//function for rendering the Particle System
ParticleSystem.prototype.transform = function (resources) {
  if (this.active){ //check if Particle System is active, camera close by
    for (var i=0; i<numberOfParticles;i++){// for each particle
     this.particles[i].life=this.particles[i].life-1;   // lower the life by one
     if (this.particles[i].life<0){ //if particle death
       this.particles[i].matrix = glm.transform({translate:[this.originX+Math.random()/100, this.originY, this.originZ+Math.random()/100], scale: 1});
       //refresh matrix at the origin
       this.particles[i].life=Math.floor(Math.random()*100);
       //refresh life with a lifespan between 0 and 1000
     } else {
        mat4.multiply(this.particles[i].matrix,this.particles[i].matrix, glm.transform({translate: [0,0.004,0]}));
        //if paricle is alive perform a transformation
     }
    }
  }
};
```

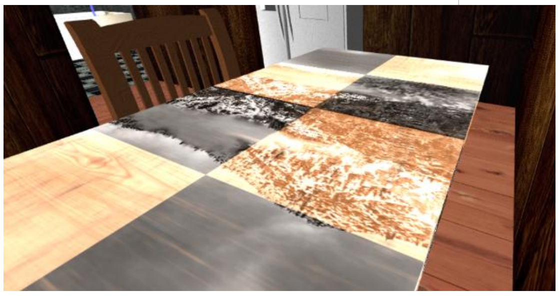
Multitexuring

First of all, the Textures will be initialized with the initMultitexturing(resources) Function. Afterwards during the creation of the Scene Graph the Textures will be used for a new TextureSGNode:

```
class TextureSGNode extends SGNode {
  constructor(texture, texture2, bitmap, children ) {
      super(children);
      this.texture = texture;
      this.texture2 = texture2;
      this.bitmap=bitmap;
      this.textureunit = 1;
      this.textureunit2 = 2;
      this.textureunit3= 3;
  }

  render(context)
  {
    gl.uniform1i(gl.getUniformLocation(context.shader, 'u_tex'), this.textureunit);
    gl.activeTexture(gl.TEXTURE0 + this.textureunit);
    gl.bindTexture(gl.TEXTURE_2D, this.texture);
    gl.uniform1i(gl.getUniformLocation(context.shader, 'u_tex2'), this.textureunit2);
    gl.activeTexture(gl.TEXTURE0 + this.textureunit2);
    gl.bindTexture(gl.TEXTURE_2D, this.texture2);
    gl.uniform1i(gl.getUniformLocation(context.shader, 'u_tex3'), this.textureunit3);//u_tex3 is bitmap
    gl.activeTexture(gl.TEXTURE0 + this.textureunit3);
    gl.bindTexture(gl.TEXTURE_2D, this.bitmap);
    super.render(context);
    gl.activeTexture(gl.TEXTURE0 + this.textureunit);
    gl.activeTexture(gl.TEXTURE0 + this.textureunit2);
    gl.activeTexture(gl.TEXTURE0 + this.textureunit3);
    gl.bindTexture(gl.TEXTURE_2D, null);
  }
}
```

(The Multitexuring shader needs 3 textures u_tex and u_tex2 main textures and u_tex3 as Alpha-Map. It generates the Alpha-Value with float c = texture2D(u_tex3,v_texCoord).r

Afterwards the covering Texture will be multiplied with the Alpha value and than added to the basic texture:
gl_FragColor=texture2D(u_tex,v_texCoord)*c +texture2D(u_tex2,v_texCoord);



Movement

Tab = change between free and animated Camera flight
up/down = moving forward/backwards
Mouse = change camera pitch and yaw

Additional Information

We adopted the given framework, so that Material-Files are now recognized and instantaneous loaded. This was approved by contacting the CG-Staff per E-Mail.