0 – No light

Objects are rendered with their (ambient material) colour with no modulation.

The controls do nothing...

```
<script id="vertex" type="x-shader">
  uniform mat4 uModelViewMatrix;
  uniform mat4 uProjectionMatrix;

  attribute vec3 aVertexPosition;
  attribute vec3 aVertexNormal;

  uniform vec4 uAmbientMaterial;

  varying vec4 vColor;

  void main(void) {
    gl_Position = uProjectionMatrix * uModelViewMatrix * vec4(aVertexPosition, 1.0);

    vColor = uAmbientMaterial;
  }
script>
```

```
<script id="fragment" type="x-shader">
    precision highp float;
    varying vec4 vColor;

    void main() {
      gl_FragColor = vColor;
    }
</script>
```

1 – No light → Ambient Lighting and emissive material

Add ambient light, emissive material and diffuse material uniforms (applied to all pixels) to ***vertex*** shader.

Modulate the ambient material colour with the ambient light, and apply the emissive material colour.

```
<script id="vertex" type="x-shader">
  uniform mat4 uModelViewMatrix;
  uniform mat4 uProjectionMatrix;

  attribute vec3 aVertexPosition;
  attribute vec3 aVertexNormal;

  uniform vec4 uAmbientMaterial;

  uniform vec3 uAmbientLightColour;
  uniform vec4 uEmissiveMaterial;
  varying vec4 vColor;

  void main(void) {
    gl_Position = uProjectionMatrix * uModelViewMatrix * vec4(aVertexPosition, 1.0);

    vColor = uAmbientMaterial * vec4(uAmbientLightColour, 1.0);
    vColor += uEmissiveMaterial;
  }
</script>
```

Get location of ambient light and emissive material uniforms

```
  function initVariableLocations() {
    var uniforms = ["uProjectionMatrix",
                    "uModelViewMatrix",
                    "uAmbientMaterial",
                    "uAmbientLightColour",
                    "uEmissiveMaterial",
    ];
```

Set emissive material colour

```
function drawAll() {
//...
      gl.uniform4fv(shaderProgram.uAmbientMaterial, model.ambientMaterial);
      gl.uniform4fv(shaderProgram.uEmissiveMaterial, model.emissiveMaterial);
```

2 – Ambient → Directional light – vertex shader

Add directional light uniforms and attributes to ***vertex*** shader

```
<script id="vertex" type="x-shader">
  //
  uniform mat3 uNMatrix;
  uniform vec3 uLightPosition;
  uniform vec3 uDirectionalColor;
  varying vec3 vLightWeighting;

  varying vec4 vColor;
```

Determine light weighting and set colour

```
  void main(void) {
    //...

    vec3 transformedNormal = normalize(uNMatrix * aVertexNormal);
    float directionalLightWeighting = max(dot(transformedNormal, uLightPosition), 0.0);
    vLightWeighting = uDirectionalColor * directionalLightWeighting;

    vColor += uAmbientMaterial * vec4(vLightWeighting, 1.0);
  }
```

Get location of direction light uniforms and attributes

```
  function initVariableLocations() {
    var uniforms = ["uProjectionMatrix",
                    "uModelViewMatrix",
                    "uAmbientMaterial",
                    "uAmbientLightColour",
                    "uEmissiveMaterial",
                    "uNMatrix",
                    "uDirectionalColor",
                    "vLightWeighting",
    ];

    var attributes = ["aVertexPosition",
                      "aVertexNormal",
    ];
```

Set light direction uniforms and attributes

```
function drawAll() {
  ...
  gl.uniform3fv(shaderProgram.uAmbientLightColour, ambientLightColour);
  gl.uniform3fv(shaderProgram.uDirectionalColor, directionalLightColour);

  mat4.perspective(45, gl.viewportWidth / gl.viewportHeight, 0.1, 100.0, pMatrix);
  mat4.identity(mvMatrix);
  mat4.translate(mvMatrix, [0.0, -1.0, depth]);
  mat4.multiply(mvMatrix, scene.rotation);

  var lightingDirection = [
      scene.models["light"].rotation[12],
      scene.models["light"].rotation[13],
      scene.models["light"].rotation[14]
  ];
  mat4.multiplyVec3(scene.rotation, lightingDirection, lightingDirection);
  var adjustedLD = vec3.create();
  vec3.normalize(lightingDirection, adjustedLD);
  gl.uniform3fv(shaderProgram. uLightPosition, adjustedLD);

  for (var key in scene.models) {
    ...
    gl.bindBuffer(gl.ARRAY_BUFFER, model.vertexPositionBuffer);
    gl.vertexAttribPointer(shaderProgram.aVertexPosition, model.vertexPositionBuffer.itemSize, gl.FLOAT,
false, 0, 0);

    var normalMatrix = mat3.create();
```

```
    mat4.toInverseMat3(mvMatrix, normalMatrix);
    mat3.transpose(normalMatrix);
    gl.uniformMatrix3fv(shaderProgram.uNMatrix, false, normalMatrix);

    gl.bindBuffer(gl.ARRAY_BUFFER, model.vertexNormalBuffer);
    gl.vertexAttribPointer(shaderProgram.aVertexNormal, model.vertexNormalBuffer.itemSize, gl.FLOAT,
false, 0, 0);
```

Changes to the controls to handle directional light colour

```
function initdatGUI() {
  var gui = new dat.GUI();
  var f2 = gui.addFolder("Control options");
  f2.add(parameters, 'control', ['Light', 'Scene']).onChange(function () {
handleRadioButton(parameters.control) });
  f2.open();
  var f1 = gui.addFolder("Light Colour");
  f1.addColor(parameters, 'ambient').name("Ambient").onChange(function () { setColour(ambientLightColour,
parameters.ambient); });
  f1.addColor(parameters, 'directional').name("Directional").onChange(function () {
setColour(directionalLightColour, parameters.directional);
setColour2(scene.models["light"].emissiveMaterial, directionalLightColour); });
  f1.open();
}
```

3 – Directional – vertex → Directional light – fragment

Move light weighting calculation to ***fragment*** shader for better results

```
<script id="vertex" type="x-shader">
  uniform mat4 uModelViewMatrix;
  uniform mat4 uProjectionMatrix;

  attribute vec3 aVertexPosition;
  attribute vec3 aVertexNormal;

  varying vec3 vTransformedNormal;
  uniform mat3 uNMatrix;

  void main(void) {
    gl_Position = uProjectionMatrix * uModelViewMatrix * vec4(aVertexPosition, 1.0);
    vTransformedNormal = uNMatrix * aVertexNormal;
  }
</script>
```

```
<script id="fragment" type="x-shader">
  precision highp float;
  vec4 vColor;

  uniform vec3 uAmbientLightColour;
  uniform vec4 uAmbientMaterial;

  uniform vec4 uEmissiveMaterial;

  varying vec3 vTransformedNormal;

  uniform vec3 uLightPosition;
  uniform vec3 uDirectionalColor;
  vec3 vLightWeighting;

  void main() {
    vColor = uAmbientMaterial * vec4(uAmbientLightColour, 1.0);
    vColor += uEmissiveMaterial;

    vec3 transformedNormal = normalize(vTransformedNormal);
    float directionalLightWeighting = max(dot(transformedNormal, uLightPosition), 0.0);
    vLightWeighting = uDirectionalColor * directionalLightWeighting;

    vColor += uAmbientMaterial * vec4(vLightWeighting, 1.0);
    gl_FragColor = vColor;
  }
</script>
```

4 – Directional – fragment → Directional light – specular

For specular lighting we need the position of the fragment we're processing. We'll add diffuse material handling while we're at it.

Break the projected position calculation into two and "save" the pre-projection vertex for interpolation.

```
<script id="vertex" type="x-shader">
  uniform mat4 uModelViewMatrix;
  uniform mat4 uProjectionMatrix;

  attribute vec3 aVertexPosition;
  attribute vec3 aVertexNormal;

  varying vec3 vTransformedNormal;
  varying vec3 vPosition;
  uniform mat3 uNMatrix;

  void main(void) {
    vPosition = uModelViewMatrix * vec4(aVertexPosition, 1.0);
    gl_Position = vPosition;
    vTransformedNormal = uNMatrix * aVertexNormal;
  }
</script>
```

Add the calculation for the specular highlight, and the diffuse material

```
<script id="fragment" type="x-shader">
  precision highp float;
  vec4 vColor;

  uniform vec3 uAmbientLightColour;
  uniform vec4 uAmbientMaterial;

  uniform vec4 uEmissiveMaterial;

  varying vec3 vTransformedNormal;

  uniform vec3 uLightPosition;
  uniform vec3 uDirectionalColor;
  vec3 vLightWeighting;

  uniform vec4 uDiffuseMaterial;
  uniform vec4 uSpecularMaterial;
  uniform float uShininess;

  void main() {
    vColor = uAmbientMaterial * vec4(uAmbientLightColour, 1.0);
    vColor += uEmissiveMaterial;

    vec3 transformedNormal = normalize(vTransformedNormal);
    float directionalLightWeighting = max(dot(transformedNormal, uLightPosition), 0.0);
    vLightWeighting = uDirectionalColor * directionalLightWeighting;

    vColor += uDiffuseMaterial * vec4(vLightWeighting, 1.0);

    vec3 eyeDirection = normalize(-vPosition.xyz);
    vec3 reflectionDirection = reflect(-lightDirection, transformedNormal);
    float specularLightWeighting = pow(max(dot(reflectionDirection, eyeDirection), 0.0), uShininess);

    gl_FragColor = vColor;
  }
</script>
```

Get location of direction light uniforms and attributes

```
  function initVariableLocations() {
    var uniforms = ["uProjectionMatrix",
                    ...
                    "uDiffuseMaterial",
                    "uSpecularMaterial",
                    "uShininess",
```

Correct an error

```
function initModel(geometry) {
  ...
  for (var arr = ["ambientMaterial", "emissiveMaterial", "diffuseMaterial", "specularMaterial"], i = 0; i
< arr.length; i++) {
    model[arr[i]] = geometry[arr[i]];
  }
```

Update drawAll()

```
function drawAll() {
  ...
    gl.uniform4fv(shaderProgram.uAmbientMaterial, model.ambientMaterial);
    gl.uniform4fv(shaderProgram.uEmissiveMaterial, model.emissiveMaterial);
    gl.uniform4fv(shaderProgram.uDiffuseMaterial, model.diffuseMaterial);
    gl.uniform4fv(shaderProgram.uSpecularMaterial, model.specularMaterial);
    gl.uniform1f(shaderProgram.uShininess, model.shininess);
```

5 – Directional light – point light

Change the directional light to a point light

```
<script id="fragment" type="x-shader">
  precision highp float;
  vec4 vColor;

  uniform vec3 uAmbientLightColour;
  uniform vec4 uAmbientMaterial;

  uniform vec4 uEmissiveMaterial;

  varying vec3 vTransformedNormal;

  uniform vec3 uLightPosition;
  uniform vec3 uDirectionalColor;
  vec3 vLightWeighting;

  uniform vec4 uDiffuseMaterial;
  uniform vec4 uSpecularMaterial;
  uniform float uShininess;

  void main() {
    vColor = uAmbientMaterial * vec4(uAmbientLightColour, 1.0);
    vColor += uEmissiveMaterial;

    vec3 lightDirection = normalize(uLightPosition - vPosition.xyz);
    vec3 transformedNormal = normalize(vTransformedNormal);
    float directionalLightWeighting = max(dot(transformedNormal, lightDirection), 0.0);
    vLightWeighting = uDirectionalColor * directionalLightWeighting;

    vColor += uDiffuseMaterial * vec4(vLightWeighting, 1.0);

    vec3 eyeDirection = normalize(-vPosition.xyz);
    vec3 reflectionDirection = reflect(-lightDirection, transformedNormal);
    float specularLightWeighting = pow(max(dot(reflectionDirection, eyeDirection), 0.0), uShininess);

    gl_FragColor = vColor;
  }
</script>
```

Update drawAll()

```
function drawAll() {
  ...
  var lightingDirection = [
      scene.models["light"].rotation[12],
      scene.models["light"].rotation[13],
      scene.models["light"].rotation[14]
  ];
  mat4.multiplyVec3(scene.rotation, lightingDirection, lightingDirection);
  var adjustedLD = vec3.create();
  vec3.normalize(lightingDirection, adjustedLD);
  gl.uniform3fv(shaderProgram.uLightPosition, adjustedLD);

  pushMatrix(mvMatrixStack, mvMatrix);
    var model = scene.models["light"];
    mat4.translate(mvMatrix, model.position);
    mat4.multiply(mvMatrix, model.rotation);
    mat4.scale(mvMatrix, model.scale);

    var lightPosition = vec3.create(scene.models["light"].position);
    mat4.multiplyVec3(mvMatrix, lightPosition);
    gl.uniformMatrix4fv(shaderProgram.uLightViewMatrix, false, mvMatrix);
    gl.uniform3fv(shaderProgram.uLightPosition, lightPosition);
  mvMatrix = popMatrix(mvMatrixStack);
```