浙江大学

课程设计报告

课程名称：计算机图形学

指导教师：吴鸿智

参加成员：3170103294 孔伊宁

3170103551 王 玥

专业类别：计算机科学

所在学院：计算机科学

论文提交日期 2018 年 1 月 17 日

目录

一、绪论

1.1 设计背景

以场景为基础，实现第一视角漫游夏夜晚上的场景。主要灵感来源于网游中的场景，希望通过大作业的机会来实现场景的搭建和基本漫游。

场景选为古代场景的夜晚，在一个山丘中的农家生活。有农耕禾苗，建筑和一些树木组成夏天的夜晚场景。

1.2 主要内容和难点

用原生 webgl 写，在一些基础的实现上需要查询一些资料来理解。相关资料不如 opengl 多，查找困难。

在模型搭建（学习使用 3dmax）和处理（读取 obj 和 mtl 文件）中理解格式和统一格式上耗费时间较多。光照模型的搭建和实现，视图矩阵的变换和实时阴影的应用都是主要难点。

html、GLSL ES、js 和 css 语言需要重新理解学习，控制台调试较为繁复。

二、设计原理

2.1 着色器

webgl 绘制的第一步，就是将顶点信息传输到着色器后在显卡上显示。着色器的语言时候用 GLSL ES 语言，需要用字符串的方式传入。

其中注意到，webgl 没有颜色缓冲区，通过 attribute、uniform 和 varying 变量实现不同值在顶点和片元着色器之间的交换。

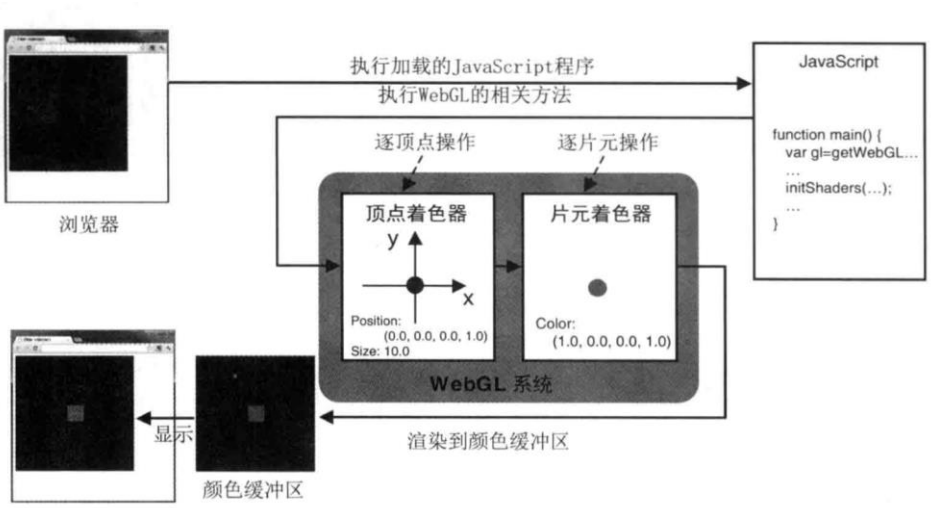

图 1 着色器结构

2.2 视角变换

2.2.1 webgl 坐标

webgl 使用笛卡尔坐标系（右手坐标系），而绘制到网页 canvas 上会转化成以 1 为最大值的坐标。在计算时，需要单位化和正负转换。

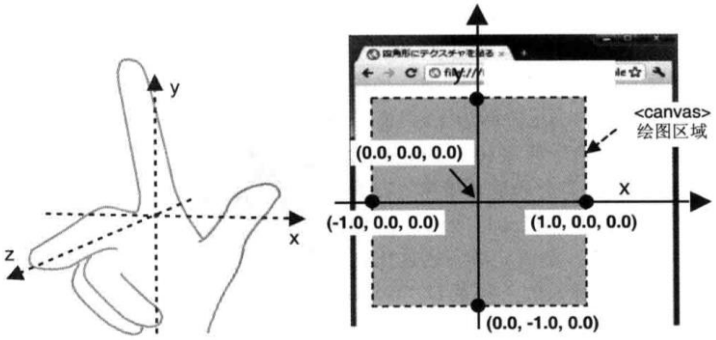

图 2 webgl 坐标和 canvas 坐标

实际在操作时，通过调试来判断矩阵乘除的正负，将比经过计算后得出要方便快捷。

2.2.2 变换矩阵

视角的变换通过键盘事件和鼠标事件改变视图矩阵的值。其中注意到矩阵变换是对坐标左乘，矩阵运算的顺序需要注意。

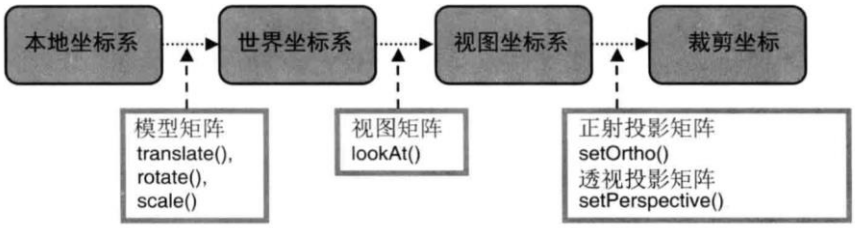

图 3 坐标变换

通过定义全局变量 modelmatrix 来计算，我们将视角矩阵乘除放入到变换后的世界坐标系之前，然后裁剪坐标。

2.3 模型导入

虽然在 webgl 中也可以实现模型自由组合，搭建等操作，但运用建模软件可以更加直观地实现效果。我们使用 3dmax 来进行 obj（Wavefront）的导出和 webgl 对 obj 的导入。为了保证 alpha 通道的有效性，贴图使用 png 格式。

此处注意到 blender 导出的 obj 格式和 3dmax 稍有不同。

obj 文件中写入应用的贴图，包含的顶点，面法向量等信息。mtl 文件中保存着对引用贴图的信息。

obj 文件格式:

顶点数据(Vertex data):

v **几何体顶点** (Geometric vertices)

vt **贴图坐标点** (Texture vertices)

vn **顶点法线** (Vertex normals)

vp **参数空格顶点** (Parameter space vertices)

元素(Elements):

p **点** (Point)

l **线** (Line)

f **面** (Face)

成组(Grouping):

g **组名称** (Group name)

s **光滑组** (Smoothing group)

显示(Display)/**渲染属性**(render attributes):

usemtl **材质名称** (Material name)

mtllib **材质库** (Material library)

mtl 文件格式:

材质颜色光照

环境反射(三选一)

Ka r g b

Ka spectral file.rfl factor

Ka xyz x y z

漫反射:

Kd r g b

Kd spectral file.rfl factor

Kd xyz x y z

镜反射:

Ks r g b

Ks spectral file.rfl factor

Ks xyz x y z

滤光透射率:

Tf r g b

Tf spectral file.rfl factor

Tf xyz x y z

我们通过创建自定义 obj 格式的结构体，导入 obj 格式的信息，并将这些信息转化成

顶点坐标信息传入事先准备好的空的缓冲区对象，最后将解析出的顶点数据写入着色器在 canvas 中显示。

在这里，我们获取模型文件内容与程序异步进行，在浏览器完成对模型文件后调用注册的响应函数事先模型绘制。

在构造新的结构函数时，使用应用 init 方法来创建成员函数。

2.4 光照变化

2.4.1 环境光源

定义全局变量，设置环境光，偏暗色。运用定义的环境光来进行后续光照模型的颜色计算。

2.4.2 漫反射光

定义漫反射光，实现环境光下的漫反射。由于环境光是均匀地设在物体表面，

<环境反射光颜色>=<入射光颜色>×<表面基底色>

<表面的反射光颜色>=<漫反射光颜色>+<环境反射光颜色>

图 4 反射光颜色变换

2.4.3 镜面反射光

定义镜面反射光向量，与模型的法向量做乘积后以 rgb 的形式成镜面反射系数加入顶点的颜色信息中。计算过程与漫反射相似。

注意，此处运用模型矩阵的逆转置矩阵求得变换后的法向量而不能直接引用模型的法向量。

三、设计实现

3.1 场景效果

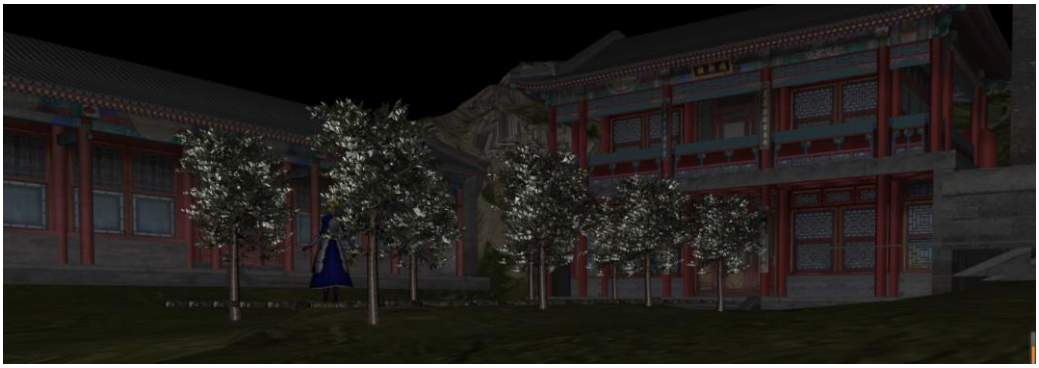

图 5 部分场景

具体可见视频内容。

3.2 视角变换

用方向键和鼠标滚轮实现场景漫游:
W：视角拉近
S：视角放远
A：视角左移
D：视角右移
Q：视角上移
E：视角下移
鼠标滚轮：视角拉近/远

四、结果和展望

4.1 困难

在完成的过程中，在模型的搭建和贴图上花费了较大的时间。应用 3dmax 导出 obj 的格式不可控制，alpha 通道的值未能反映在画布上。这个问题暂时没有得到解决。

由于在一些矩阵变换中参考了他人的代码，自主调用时受到一定的限制，可以尝试重新写一遍转换等辅助函数，来更加灵活地实现。

场景搭建，是贴图平面，漫游时会穿过地面，由于没有物体做碰撞检测，可以尝试使用高度来阻止控制。由于涉及操作键较多，计算复杂，未能实现。

4.2 改进

由于时间紧张，在一些方面完成度有限。比如实时阴影，通过设置两个着色器，分别绘制实际模型和阴影模型（将视角变换到光源位置，判断深度）。天空盒的实现（用包围盒进行映射），增加空间的广度和真实感。高光贴图，通过模型搭建直接实现效果更加丰富和真实感更加强烈的图像效果。

在运行时，虽然已经使用异步加载，load 模型的过程依旧花费较多时间，由于图形较精细，可以考虑通过深度检测，同一个模型准备精细和粗糙版。我们绘制过程通过判断加载完毕后绘制，可先根据视图矩阵判断屏幕范围，进行裁剪后选择性加载文件。在加载过程中的黑屏体验效果不佳，可考虑增加封面，加载进度条的方式，保持画面动态。

动态效果不多，包括粒子系统和雾化效果都和计划的实现相差较多。

4.3 心得

webgl 入门较为不容易，但是由于调用 API 比较底层，且网页直接实现。在运用时不会受到库函数的限制，比 opengl 较为自由。在查找一些资料时也有一定的限制，如果有更多的时间可以做深入的研究会比较好。

附录

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">
    <title>仲夏夜之梦</title>
    <link rel="stylesheet" type="text/css" href="./css/style.css">
</head>

<body>
<center>
    <canvas id="webgl" width="1400" height="800">
    Please use a browser that supports "canvas"
    </canvas>

        <!--顶点 Shader 程序-->
        <script id="vertex-shader" type="x-shader/x-vertex">
        attribute vec3 a_Position;  // 输入三维顶点坐标(建模坐标系)
        attribute vec3 a_Normal;    // 顶点法向(建模坐标系)
        attribute vec2 a_TexCoord;  // 顶点纹理坐标

        varying vec3 v_fN;          // 法向(观察坐标系)
        varying vec3 v_fE;          // 观察向量(观察坐标系)
        varying vec3 v_fL;          // 光照向量(观察坐标系)
        varying vec2 v_TexCoord;    // 输出纹理坐标

        uniform mat4 u_ModelView;       // 模视矩阵
        uniform mat4 u_Projection;      // 投影矩阵
        uniform vec4 u_LightPosition;   // 光源位置(观察坐标系)

        void main(){
            // 将顶点坐标转到观察坐标系下(在观察坐标系计算光照)
            vec3 pos = (u_ModelView * vec4(a_Position, 1.0)).xyz;
            v_fE = normalize(-pos);     // 观察者方向向量
            // 将顶点法向转到观察坐标系下(针对模视变换不含非均匀缩放情况)
            v_fN = normalize((u_ModelView * vec4(a_Normal, 0.0)).xyz);
            if(u_LightPosition.w != 0.0) // 近距离光源
                v_fL = normalize(u_LightPosition.xyz - pos);
            else                        // 远距离光源
```

```
40.                 v_fL = normalize(u_LightPosition.xyz);
41.
42.             // 裁剪坐标系下顶点坐标
43.             gl_Position = u_Projection * vec4(pos, 1.0);
44.
45.             v_TexCoord = a_TexCoord;
46.         }
47.         </script>
48.
49.         <!--片元 Shader 程序-->
50.         <script id="fragment-shader" type="x-shader/x-fragment">
51.         precision mediump float;     // 浮点数精度为中等
52.
53.         varying vec3 v_fN;           // 法向(观察坐标系)
54.         varying vec3 v_fE;           // 观察向量(观察坐标系)
55.         varying vec3 v_fL;           // 光照向量(观察坐标系)
56.         varying vec2 v_TexCoord;     // 输入纹理坐标
57.
58.         // 光源的环境光、漫反射光和镜面光分量
59.         uniform vec3 u_AmbientLight;
60.         uniform vec3 u_DiffuseLight;
61.         uniform vec3 u_SpecularLight;
62.
63.         // 物体材质
64.         uniform vec3 u_Ka;  // 环境光反射系数
65.         uniform vec3 u_Kd;  // 漫反射系数
66.         uniform vec3 u_Ks;  // 镜面反射系数
67.         uniform vec3 u_Ke;  // 发射系数
68.         uniform float u_Ns; // 高光系数
69.         uniform float u_d;  // 不透明度
70.
71.         uniform sampler2D u_Sampler;  // 纹理采样器
72.
73.         void main(){
74.             // 归一化输入的向量
75.             vec3 N = normalize(v_fN);
76.             vec3 E = normalize(v_fE);
77.             vec3 L = normalize(v_fL);    // 光源方向向量(从顶点指向光源)
78.             vec3 H = normalize(L + E);  // 半角向量
79.
80.             // 环境反射分量
81.             vec3 ambient = u_AmbientLight * u_Ka;
82.
83.             // 漫反射分量
```

```
84.             float Fd = max(dot(L, N), 0.0);
85.             vec3 diffuse = Fd * u_DiffuseLight * u_Kd;
86.
87.             // 镜面反射分量
88.             vec3 specular = vec3(0.0, 0.0, 0.0); // 镜面反射分量
89.             if( Fd != 0.0 ) {  // 即 dot(L, N) > 0
90.                 float Fs = pow(max(dot(N, H), 0.0), u_Ns);
91.                 specular = Fs * u_SpecularLight * u_Ks; // 镜面反射分量
92.             }
93.
94.             // 累加光照计算得到颜色，先不考虑镜面反射分量
95.             gl_FragColor.rgb = ambient + diffuse + u_Ke;
96.             // 获取纹理颜色，并与光照颜色相乘(使纹理颜色受光照影响)
97.             // 然后加上镜面反射分量
98.             gl_FragColorgl_FragColor.rgb = gl_FragColor.rgb *
99.                 texture2D(u_Sampler, v_TexCoord).rgb + specular;
100.
101.              // 设置不透明度
102.              gl_FragColor.a = u_d;
103.         }
104.         </script>
105.
106.     <script src="js/common/webgl-utils.js"></script>
107.     <script src="js/common/webgl-debug.js"></script>
108.     <script src="js/common/cuon-utils.js"></script>
109.
110.     <script src="js/common/Controls.js"></script>
111.     <script src="js/common/CanvasMatrix.js"></script>
112.     <script src="js/common/MV.js"></script>
113.
114.     <script src="js/ObjModel.js"></script>
115.     <script src="js/Matrix.js"></script>
116.     <script src="js/initShaders.js"></script>
117.     <script src="js/index.js"></script>
118.     <script src="js/drawscene.js"></script>
119.
120. </center>
121. </body>
122. </html>
```

style.css

```
body {
        font-family: Monospace;
        background-color: #000;
        color: #fff;
        margin: 0px;
        overflow: hidden;
    }
*   {
        margin: 0;
        padding: 0;
      }
      html, body {
        height: 100%;
        width: 100%;
      }
      canvas {
        display: block;
      }
```

ObjModel.js

```
// ObjModel.js

// 请求加载模型文件
function loadOBJ(fileName){
    var objModel = new OBJModel(fileName);
    var request = new XMLHttpRequest();

    request.onreadystatechange = function() {
        if (request.readyState === 4 && request.status !== 404) {
            console.log(fileName + "加载完毕");
            onReadOBJFile(request.responseText, objModel);
        }
        else if(request.status === 404) // 文件找不到
            console.log("obj 文件加载失败：" + fileName);
    }
    // 创建获取文件的请求，最后一个参数表示采用异步方式
    request.open('GET', fileName, true);
    request.send();  // 发送请求
    console.log("开始加载材质文件：" + fileName);
    return objModel;
}
```

```
22.
23. function onReadOBJFile(fileString, objModel){
24.     return objModel.readOBJFile(fileString);
25. }
26.
27. // OBJModel 对象
28. // 构造函数(参数为 Obj 模型文件路径)
29. var OBJModel = function(fileName) {
30.     this.fileName = fileName;
31.     this.mtllib = null;               // 材质库，初始化为空
32.     this.groups = new Array(0);       // group 数组，初始为空
33.     this.vertices = new Array(0);     // 顶点坐标数组，初始为空
34.     this.normals = new Array(0);      // 顶点法向数组，初始为空
35.     this.texcoords = new Array(0);    // 顶点纹理坐标数组，初始为空
36.     this.complete = false;            // obj 文件是否读取完毕
37.     this.ready = false;               // 为 true 则一切就绪，可直接绘制了
38. }
39.
40. // 读取 obj 文件内容
41. OBJModel.prototype.readOBJFile = function(fileString){
42.     console.log("开始读取 OBJ 文件" + this.fileName + "内容");
43.     // 用换行符对文件进行切分，即获取每一行内容，存为数组
44.     var lines = fileString.split('\n');
45.     lines.push(null); // 添加一个 null 数组元素，作为结束标志
46.     var index = 0;    // lines 数组下标
47.
48.     var currentGroup = null; // 当前 group
49.
50.     // 逐行进行解析
51.     var line;  // 存储待解析的一行字符
52.     var sp = new StringParser();  // 创建字符串解析器
53.     var x, y, z, s, t;  // 临时变量
54.     while ((line = lines[index++]) != null) {
55.         sp.init(line);              // 初始化 StringParser
56.         var command = sp.getWord(); // 获取当前行的指令
57.         if(command == null)  continue;  // 当前行为空行
58.
59.         switch(command){
60.         case '#':
61.             continue;  // 跳过注释行
62.         case 'mtllib': // 读取材质文件
63.             var path = this.parseMtllib(sp, this.fileName);
64.             this.mtllib = new MTLLib(path);  // 创建 MTLLib 对象
65.             loadMTL(this.mtllib);    // 加载材质文件到 this.mtllib
```

```
66.             continue; // 继续处理下一行
67.         case 'v':   // 读取顶点
68.             x = sp.getFloat();
69.             y = sp.getFloat();
70.             z = sp.getFloat();
71.             this.vertices.push(vec3(x, y, z));
72.             continue; // 继续处理下一行
73.         case 'vn':   // 读取法向
74.             x = sp.getFloat();
75.             y = sp.getFloat();
76.             z = sp.getFloat();
77.             this.normals.push(vec3(x, y, z));
78.             continue; // 继续处理下一行
79.         case 'vt':   // 读取纹理坐标
80.             s = sp.getFloat();
81.             t = sp.getFloat();
82.             this.texcoords.push(vec2(s, t));
83.             continue; // 继续处理下一行
84.         case 'o':
85.         case 'g':   // 组对象名称行
86.             // 如果前一组对象没有顶点，则将其从 group 数组中弹出
87.             if(currentGroup != null && currentGroup.vertices.length == 0)
88.                 this.groups.pop();
89.             // 读取组对象名称并创建组对象
90.             currentGroup = new Group(sp.getWord());
91.             this.groups.push(currentGroup);
92.             continue; // 继续处理下一行
93.         case 'usemtl': // 读取当前组对象所使用的材质名称
94.             // 如果当前 Group 已有材质，则新建一个 Group 对象
95.             // 该 Group 对象名称和当前 Group 一样
96.             if(currentGroup.mtlName != null){
97.                 currentGroup = new Group(currentGroup.name);
98.                 this.groups.push(currentGroup);
99.             }
100.             currentGroup.mtlName = sp.getWord();
101.             continue; // 继续处理下一行
102.          case 'f': // 面信息
103.             // 解析面信息并添加到 currentGroup 中
104.             this.parseFace(sp, currentGroup);
105.             continue; // 继续处理下一行
106.         }
107.     }
108.     this.complete = true;
109.     console.log("OBJ 文件" + this.fileName + "内容读取完毕");
```

```
110.     return true;
111. }
112. 
113. // 解析面信息并添加到 currentGroup 中
114. OBJModel.prototype.parseFace = function(sp, currentGroup) {
115.     //console.log(currentGroup.name + "面信息");
116.     var vIndices = new Array(0); // 顶点坐标索引数组
117.     var tIndices = new Array(0); // 顶点纹理坐标索引数组
118.     var nIndices = new Array(0); // 顶点法向数组
119.     // 获取面索引信息
120.     for(;;){
121.         var word = sp.getWord();
122.         if(word == null) break;  // 获取 word 为空则此行解析完毕
123.         //console.log(word);
124.         //alert("ok");
125.         var subWords = word.split('/');
126.         if(subWords.length >= 1){
127.             // 顶点坐标(减 1 是因为 obj 文件里顶点编号从 1 开始)
128.             var vi = parseInt(subWords[0]) - 1;
129.             if(isNaN(vi))
130.                 break;
131.             vIndices.push(vi);
132.         }
133.         if(subWords.length >= 2){
134.             // 纹理坐标
135.             if(subWords[1] != ""){
136.                 var ti = parseInt(subWords[1]) - 1;
137.                 tIndices.push(ti);
138.             }
139.         }
140.         if(subWords.length >= 3){
141.             // 法向
142.             if(subWords[2] != ""){
143.                 var ni = parseInt(subWords[2]) - 1;
144.                 nIndices.push(ni);
145.             }
146.         }
147.     }
148. 
149.     // 如果构成 face 的顶点数大于 3，则要拆分成多个三角形
150.     if(vIndices.length > 3){
151.         //console.log("存在面顶点数超过 3 的情况：%d", vIndices.length);
152.         //console.log("%d %d %d %d", vIndices[0], vIndices[1], vIndices[2],
    vIndices[3]);
```

```
153.         var n = vIndices.length - 2; // 3 角形数
154.         var newVIndices = new Array(n * 3);
155.         if(tIndices.length > 0)
156.             var newTIndices = new Array(n * 3);
157.         if(nIndices.length > 0)
158.             var newNIndices = new Array(n * 3);
159.         for(var i = 0; i < n; i++){
160.             newVIndices[i * 3 + 0] = vIndices[0];
161.             newVIndices[i * 3 + 1] = vIndices[i + 1];
162.             newVIndices[i * 3 + 2] = vIndices[i + 2];
163.             if(tIndices.length > 0){
164.                 newTIndices[i * 3 + 0] = tIndices[0];
165.                 newTIndices[i * 3 + 1] = tIndices[i + 1];
166.                 newTIndices[i * 3 + 2] = tIndices[i + 2];
167.             }
168.             if(nIndices.length > 0){
169.                 newNIndices[i * 3 + 0] = nIndices[0];
170.                 newNIndices[i * 3 + 1] = nIndices[i + 1];
171.                 newNIndices[i * 3 + 2] = nIndices[i + 2];
172.             }
173.         }
174.         vIndices = newVIndices;
175.         if(tIndices.length > 0)
176.             tIndices = newTIndices;
177.         if(nIndices.length > 0)
178.             nIndices = newNIndices;
179.     }
180.
181.     // 将顶点数据添加到 Group 对象中
182.     for(var i = 0; i < vIndices.length; i++){
183.         currentGroup.vertices.push(this.vertices[vIndices[i]]);
184.         if(tIndices.length > 0)
185.             currentGroup.texcoords.push(this.texcoords[tIndices[i]]);
186.         if(nIndices.length > 0)
187.             currentGroup.normals.push(this.normals[nIndices[i]]);
188.     }
189. }
190.
191. // 获取 mtllib 文件名
192. OBJModel.prototype.parseMtllib = function(sp, fileName) {
193.     // 获取目录路径
194.     var i = fileName.lastIndexOf("\\");
195.     var dirPath = "";
196.     if(i > 0) dirPath = fileName.substr(0, i+1);
```

```
197.
198.     return dirPath + sp.getRemain(); // 得到完整路径
199. }
200.
201. // 是否一切就绪，可以绘制了？
202. OBJModel.prototype.isAllReady = function(gl) {
203.     // 如果 obj 相关文件信息读取完毕,但绘制准备工作还未做
204.     if(!this.ready && this.isComplete()){
205.         // 为绘制做准备，例如初始化纹理对象、
206.         // 查找各 Group 对象所使用材质、初始化顶点缓存对象
207.         this.onReadComplete(gl);
208.         this.ready = true; // 此时一切就绪了
209.     }
210.     return this.ready;
211. }
212.
213. // 是否相关信息已全部加载完毕
214. OBJModel.prototype.isComplete = function() {
215.
216.     if(this.complete && this.mtllib.isComplete())//error: this.complete is
    false
217.         return true;
218.     else
219.         return false;
220. }
221.
222. // 全部信息读取完成后，为绘制做准备
223. OBJModel.prototype.onReadComplete = function(gl){
224.     console.log(this.fileName + "相关文件全部读取完毕");
225.
226.     // 初始化纹理对象
227.     this.mtllib.initTextures(gl);
228.
229.     console.log("查找 Group 对象材质及初始化顶点缓存对象");
230.     for(var i = 0; i < this.groups.length; i++){
231.         // 根据 group 对象的材质名称查找材质对象
232.         this.groups[i].material =
233.             this.mtllib.findMTL(this.groups[i].mtlName);
234.         // 初始化各缓存对象
235.         this.groups[i].initBuffers(gl);
236.     }
237.
238.     // 释放空间
239.     this.vertices.length = 0;
```

```
240.     this.normals.length = 0;
241.     this.texcoords.length = 0;
242. }
243. 
244. // 绘制 OBJ 模型，后面 3 个参数分别为 shader 中顶点属性变量索引、
245. // uniform 材质变量索引和纹理采样器索引
246. OBJModel.prototype.draw = function(gl, attribIndex, mtlIndex, u_Sampler){
247.     if(attribIndex == null) return;
248.     for(var i = 0; i < this.groups.length; i++){
249.         this.groups[i].draw(gl, attribIndex, mtlIndex, u_Sampler);
250.     }
251. }
252. 
253. // 请求加载模型文件
254. function loadMTL(mtlLib){
255.     var request = new XMLHttpRequest();
256. 
257.     request.onreadystatechange = function() {
258.         if (request.readyState === 4 && request.status !== 404) {
259.             console.log(mtlLib.fileName + "加载完毕");
260.             onReadMTLFile(request.responseText, mtlLib);
261.         }
262.         else if(request.status === 404) // 文件找不到
263.             console.log("mtl 文件加载失败：" + mtlLib.fileName);
264.     }
265.     // 创建获取文件的请求，最后一个参数表示采用异步方式
266.     request.open('GET', mtlLib.fileName, true);
267.     request.send();  // 发送请求
268.     console.log("开始加载材质文件：" + mtlLib.fileName);
269. }
270. 
271. function onReadMTLFile(fileString, mtlLib){
272.     mtlLib.readMTLFile(fileString);
273. }
274. 
275. // Group 对象
276. // 构造函数(参数为 Group 对象名称)
277. var Group = function(name) {
278.     this.name = name;                 // Group 对象名称
279.     this.mtlName = null;              // 使用材质名称
280.     this.material = new Material(null);  // 使用材质，初始化为默认材质
281.     this.vertices = new Array(0);     // 顶点坐标数组，初始为空
282.     this.normals = new Array(0);      // 顶点法向数组，初始为空
283.     this.texcoords = new Array(0);    // 顶点纹理坐标数组，初始为空
```

```
284.
285.     this.vBuffer = null;     // 顶点坐标 buffer 对象
286.     this.tBuffer = null;     // 顶点纹理坐标 buffer 对象
287.     this.nBuffer = null;     // 顶点法向 buffer 对象
288.     this.numVertices = 0;    // 顶点数，初始化为 0
289. }
290.
291. // 绘制 Group 对象，后面 3 个参数分别为 shader 中顶点属性变量索引、
292. // uniform 材质变量索引和纹理采样器索引
293. Group.prototype.draw = function(gl, attribIndex, mtlIndex, u_Sampler){
294.     if(attribIndex == null) return;
295.     // 为 a_Position 提供数据
296.     if(attribIndex.a_Position != -1 && this.vBuffer != null){
297.         gl.bindBuffer(gl.ARRAY_BUFFER, this.vBuffer);
298.         gl.vertexAttribPointer(
299.             attribIndex.a_Position, // 属性变量索引
300.             3,                  // 每个顶点属性的分量个数
301.             gl.FLOAT,           // 数组数据类型
302.             false,              // 是否进行归一化处理
303.             0,    // 在数组中相邻属性成员起始位置间的间隔(以字节为单位)
304.             0     // 第一个属性值在 buffer 中的偏移量
305.         );
306.         gl.enableVertexAttribArray(attribIndex.a_Position);
307.     }
308.
309.     // 为 a_Texcoord 提供数据
310.     if(attribIndex.a_Texcoord != -1 && this.tBuffer != null){
311.         gl.bindBuffer(gl.ARRAY_BUFFER, this.tBuffer);
312.         gl.vertexAttribPointer(
313.             attribIndex.a_Texcoord, // 属性变量索引
314.             2,                  // 每个顶点属性的分量个数
315.             gl.FLOAT,           // 数组数据类型
316.             false,              // 是否进行归一化处理
317.             0,    // 在数组中相邻属性成员起始位置间的间隔(以字节为单位)
318.             0     // 第一个属性值在 buffer 中的偏移量
319.         );
320.         gl.enableVertexAttribArray(attribIndex.a_Texcoord);
321.     }
322.
323.     // 为 a_Normal 提供数据
324.     if(attribIndex.a_Normal != -1 && this.nBuffer != null){
325.         gl.bindBuffer(gl.ARRAY_BUFFER, this.nBuffer);
326.         gl.vertexAttribPointer(
327.             attribIndex.a_Normal,   // 属性变量索引
```

```
328.                3,                  // 每个顶点属性的分量个数
329.                gl.FLOAT,           // 数组数据类型
330.                false,              // 是否进行归一化处理
331.                0,    // 在数组中相邻属性成员起始位置间的间隔(以字节为单位)
332.                0     // 第一个属性值在 buffer 中的偏移量
333.            );
334.            gl.enableVertexAttribArray(attribIndex.a_Normal);
335.        }
336.
337.        // 为 uniform 材质变量传值
338.        if(mtlIndex != null){
339.            if(mtlIndex.u_Kd) // 漫反射系数
340.                gl.uniform3f(mtlIndex.u_Kd, this.material.Kd[0],
341.                    this.material.Kd[1], this.material.Kd[2]);
342.            if(mtlIndex.u_Ks) // 镜面反射系数
343.                gl.uniform3f(mtlIndex.u_Ks, this.material.Ks[0],
344.                    this.material.Ks[1], this.material.Ks[2]);
345.            if(mtlIndex.u_Ka) // 环境反射系数
346.                gl.uniform3f(mtlIndex.u_Ka, this.material.Ka[0],
347.                    this.material.Ka[1], this.material.Ka[2]);
348.            if(mtlIndex.u_Ke) // 发射系数
349.                gl.uniform3f(mtlIndex.u_Ke, this.material.Ke[0],
350.                    this.material.Ke[1], this.material.Ke[2]);
351.            if(mtlIndex.u_Ns) // 高光系数
352.                gl.uniform1f(mtlIndex.u_Ns, this.material.Ns);
353.            if(mtlIndex.u_d)  // 不透明度
354.                gl.uniform1f(mtlIndex.u_d, this.material.d);
355.        }
356.
357.        // 为 u_Sampler 传值
358.        if(u_Sampler != 0 && this.material.texture != null){
359.            gl.activeTexture(gl.TEXTURE0);
360.            // 绑定纹理对象到目标上
361.            gl.bindTexture(gl.TEXTURE_2D, this.material.texture);
362.            // 使用 0 号纹理单元，这里不考虑多纹理单元情况
363.            gl.uniform1i(u_Sampler, 0);
364.        }
365.
366.        gl.drawArrays(gl.TRIANGLES, 0, this.numVertices);
367. }
368.
369. // 初始化顶点 buffer 对象
370. Group.prototype.initBuffers = function(gl){
371.     console.log("开始初始化 Group 对象" + this.name + "的 buffer");
```

```
372.     console.log("顶点数：%d", this.vertices.length);
373.     if(this.vertices.length == 0)
374.         return;
375.     // 初始化顶点坐标 buffer 对象
376.     this.vBuffer = gl.createBuffer();
377.     gl.bindBuffer(gl.ARRAY_BUFFER, this.vBuffer);
378.     gl.bufferData(gl.ARRAY_BUFFER,  // Buffer 类型
379.         flatten(this.vertices),     // 数据来源
380.         gl.STATIC_DRAW  // 表明是一次提供数据，多遍绘制
381.     );
382.     this.numVertices = this.vertices.length; // 顶点数
383.     this.vertices.length = 0; // 释放空间
384.
385.     // 初始化顶点纹理坐标 buffer 对象
386.     if(this.texcoords.length > 0){
387.         this.tBuffer = gl.createBuffer();
388.         gl.bindBuffer(gl.ARRAY_BUFFER, this.tBuffer);
389.         gl.bufferData(gl.ARRAY_BUFFER,  // Buffer 类型
390.             flatten(this.texcoords),    // 数据来源
391.             gl.STATIC_DRAW  // 表明是一次提供数据，多遍绘制
392.         );
393.         this.texcoords.length = 0; // 释放空间
394.     }
395.
396.     // 初始化顶点法向 buffer 对象
397.     if(this.normals.length > 0){
398.         this.nBuffer = gl.createBuffer();
399.         gl.bindBuffer(gl.ARRAY_BUFFER, this.nBuffer);
400.         gl.bufferData(gl.ARRAY_BUFFER,  // Buffer 类型
401.             flatten(this.normals),  // 数据来源
402.             gl.STATIC_DRAW  // 表明是一次提供数据，多遍绘制
403.         );
404.         this.normals.length = 0; // 释放空间
405.     }
406. }
407.
408. // MTLLib 对象
409. // 构造函数(参数为材质文件路径)
410. var MTLLib = function(fileName) {
411.     this.fileName = fileName;
412.     this.materials = new Array(0); // 材质数组，初始为空
413.     this.textures = new Array(0);  // 纹理对象数组，初始为空
414.     this.complete = false;       // 材质文件是否读取完毕
415. }
```

```
416.
417. // 读取 mtl 文件内容
418. MTLLib.prototype.readMTLFile = function(fileString){
419.     console.log("开始读取 MTL 文件" + this.fileName + "内容");
420.     // 用换行符对文件进行切分，即获取每一行内容，存为数组
421.     var lines = fileString.split('\n');
422.     lines.push(null); // 添加一个 null 数组元素，作为结束标志
423.     var index = 0;    // lines 数组下标
424.
425.     // 逐行进行解析
426.     var newMTL = null; // 新材质对象
427.     var line;  // 存储待解析的一行字符
428.     var sp = new StringParser();  // 创建字符串解析器
429.     var r, g, b;    // 临时变量
430.     while ((line = lines[index++]) != null) {
431.         sp.init(line);              // 初始化 StringParser
432.         var command = sp.getWord(); // 获取当前行的指令
433.         if(command == null)  continue;  // 当前行为空行
434.
435.         switch(command){
436.         case '#':
437.             continue;    // 跳过注释
438.         case 'newmtl':   // 新材质名称行
439.             // 获取新材质名称并创建新的材质对象
440.             newMTL = new Material(sp.getWord());
441.             this.materials.push(newMTL);    // 添加到材质数组里
442.             continue; // 继续处理下一行
443.         case 'Kd':   // 漫反射系数
444.             r = sp.getFloat();
445.             g = sp.getFloat();
446.             b = sp.getFloat();
447.             newMTL.Kd = vec3(r, g, b);
448.             continue; // 继续处理下一行
449.         case 'Ks':   // 镜面反射系数
450.             r = sp.getFloat();
451.             g = sp.getFloat();
452.             b = sp.getFloat();
453.             newMTL.Ks = vec3(r, g, b);
454.             continue; // 继续处理下一行
455.         case 'Ka':   // 环境反射系数
456.             r = sp.getFloat();
457.             g = sp.getFloat();
458.             b = sp.getFloat();
459.             newMTL.Ka = vec3(r, g, b);
```

```
460.                 continue; // 继续处理下一行
461.             case 'Ke':   // 发射系数
462.                 r = sp.getFloat();
463.                 g = sp.getFloat();
464.                 b = sp.getFloat();
465.                 newMTL.Ke = vec3(r, g, b);
466.                 continue; // 继续处理下一行
467.             case 'Ns':  // 高光系数
468.                 newMTL.Ns = sp.getFloat();
469.                 continue; // 继续处理下一行
470.             case 'd':   // 不透明度
471.                 newMTL.d = sp.getFloat();
472.                 continue; // 继续处理下一行
473.             case 'map_Kd': // 漫反射贴图
474.                 // 获取漫反射贴图文件名
475.                 var path = this.parseMapKd(sp, this.fileName);
476.                 newMTL.map_Kd = path;
477.                 if(this.findTexture(path) != null)  // 已加载过此纹理图?
478.                     continue;
479.                 var newTex = new Texture(path);
480.                 this.textures.push(newTex);
481.                 loadImg(newTex, path); // 加载纹理图
482.                 continue; // 继续处理下一行
483.             }
484.     }
485.     console.log("材质文件" + this.fileName + "读取完毕");
486.     this.complete = true;
487. }
488.
489. // 加载纹理图
490. function loadImg(newTex, path){
491.     newTex.image = new Image(); // 新建图像对象
492.
493.     // 加载完成后调用此函数
494.     newTex.image.onload = function(){
495.         newTex.complete = true;
496.         console.log("纹理图" + path + "加载完毕");
497.     }
498.
499.     // 开始加载图像
500.     newTex.image.src = path;
501.     console.log("开始加载纹理图" + path);
502. }
503.
```

```
504. // 获取漫反射贴图文件名
505. MTLLib.prototype.parseMapKd = function(sp, fileName) {
506.     // 获取目录路径
507.     var i = fileName.lastIndexOf('\\');
508.     var dirPath = "";
509.     if(i > 0) dirPath = fileName.substr(0, i+1);
510.
511.     return dirPath + sp.getRemain(); // 得到完整路径
512. }
513.
514. // MTLLib 文件是否已全部加载完毕
515. MTLLib.prototype.isComplete = function() {
516.     if(this.complete == false){
517.         return false;
518.     }
519.     for(var i = 0; i < this.textures.length; i++){
520.         if(this.textures[i].complete == false)
521.             return false;
522.     }
523.     return true;
524. }
525.
526. // 根据材质名查找材质对象
527. MTLLib.prototype.findMTL = function(name){
528.     for(var i = 0; i < this.materials.length; i++){
529.         if(this.materials[i].name == name)
530.             return(this.materials[i]);
531.     }
532.
533.     return(new Material(null));
534. }
535.
536. // 根据纹理图像路径名查找 Texture 对象
537. MTLLib.prototype.findTexture = function(name){
538.     for(var i = 0; i < this.textures.length; i++){
539.         if(this.textures[i].pathName == name)
540.             return(this.textures[i]);
541.     }
542.
543.     return null;
544. }
545.
546. // 初始化纹理对象
547. MTLLib.prototype.initTextures = function(gl){
```

```
548.     console.log("初始化纹理对象");
549.     console.log(this.textures.length);
550.     for(var i = 0; i < this.textures.length; i++){
551.         this.textures[i].initTexture();
552.     }
553.     for(var i = 0; i < this.materials.length; i++){
554.         if(this.materials[i].map_Kd == null)
555.             continue;
556.         this.materials[i].texture =
557.             this.findTexture(this.materials[i].map_Kd).texture;
558.     }
559. }
560.
561. // Texture 对象
562. var Texture = function(pathName){
563.     this.pathName = pathName;    // 纹理图路径名
564.     this.image = null;           // image 对象
565.     this.complete = false;       // 纹理图是否加载完毕
566.     this.texture = null;         // WebGL 纹理对象
567. }
568.
569. // 加载纹理图
570. Texture.prototype.initTexture = function(){
571.     if(this.image == null || this.complete == false)
572.         return;
573.
574.     this.texture = gl.createTexture(); // 创建纹理对象
575.     console.log(this.texture);
576.
577.     // 绑定纹理对象到目标上
578.     gl.bindTexture(gl.TEXTURE_2D, this.texture);
579.
580.     // 对纹理图像进行 y 轴反转
581.     gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, 1);
582.
583.     // 配置纹理图像
584.     gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, thi
    s.image);
585.     // 自动生成各级分辨率的纹理图
586.     gl.generateMipmap(gl.TEXTURE_2D);
587.
588.     // 设置插值参数
589.     gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR_MIPMAP
    _LINEAR);
```

```
590. }
591.
592.
593. // Material 对象
594. // 构造函数
595. var Material = function(materialName){
596.     this.name = materialName;        // 材质名称
597.     this.Kd = vec3(1.0, 1.0, 1.0);   // 漫反射系数
598.     this.Ks = vec3(0.0, 0.0, 0.0);   // 镜面反射系数
599.     this.Ka = vec3(0.2, 0.2, 0.2);   // 环境反射系数
600.     this.Ke = vec3(0.0, 0.0, 0.0);   // 发射系数
601.     this.Ns = 10;   // 高光系数
602.     this.d = 1.0;   // 不透明度，默认完全不透明
603.     this.map_Kd = null; // 漫反射贴图
604.     this.texture = null;    // 纹理对象，初始为空
605. }
606.
607. //------------------------------------------------------------------------
    -----
608. // 字符串解析器对象
609. // 构造函数
610. var StringParser = function(str) {
611.     this.str;   // 待解析字符串
612.     this.index; // 字符串中处理到的位置
613.     this.init(str);
614. }
615.
616. // 初始化 StringParser 对象
617. StringParser.prototype.init = function(str){
618.     this.str = str;
619.     this.index = 0;
620. }
621.
622. // 从当前位置开始，跳过分隔符
623. StringParser.prototype.skipDelimiters = function(){
624.     var i = this.index;
625.     for(len = this.str.length; i < len; i++){
626.         var c = this.str.charAt(i);
627.         // 跳过 TAB、空格、以及左右括号
628.         if (c == '\t'|| c == ' ' || c == '(' || c == ')' || c == '"')
629.             continue;
630.         break;
631.     }
632.     this.index = i;
```

```
633. }
634.
635. // 获取下一个 word
636. StringParser.prototype.getWord = function(){
637.     this.skipDelimiters(); // 跳过分隔符
638.     // 获取 Word 的长度
639.     var n = getWordLength(this.str, this.index);
640.     if (n == 0) return null; // 没有待处理的 word 了
641.     var word = this.str.substr(this.index, n);
642.     this.index += (n + 1);
643.
644.     return word;
645. }
646.
647. // 获取当前行剩余的字符(除去末尾的空字符)
648. StringParser.prototype.getRemain = function(){
649.     this.skipDelimiters(); // 跳过分隔符
650.
651.     // 去掉末尾的空字符
652.     var endIndex;
653.     for(endIndex = this.str.length - 1; endIndex > this.index; endIndex-
    -){
654.         var c = this.str.charAt(endIndex);
655.         // 跳过 TAB、空格、以及左右括号
656.         if (c == '\t'|| c == ' ' || c == '(' || c == ')' || c == '"')
657.             continue;
658.         break;
659.     }
660.
661.     if(endIndex == this.index)
662.         return null;
663.
664.     var word = this.str.substr(this.index, endIndex);
665.     this.index += (endIndex + 1);
666.
667.     return word;
668. }
669.
670. // 获取一个 Float
671. StringParser.prototype.getFloat = function() {
672.     return parseFloat(this.getWord());
673. }
674.
675. // 获取 Word 的长度
```

```
676. function getWordLength(str, start) {
677.     var i = start;
678.     for(len = str.length; i < len; i++){
679.         var c = str.charAt(i);
680.         if (c == '\t'|| c == ' ' || c == '(' || c == ')' || c == '"')
681.             break;
682.     }
683.     return i - start;
684. }
685.
686. // 顶点属性 Index 对象
687. // 存放 shader 中对应顶点 attribute 变量的索引
688. // 构造函数
689. var AttribIndex = function(){
690.     this.a_Position = -1;    // 顶点坐标索引
691.     this.a_Normal   = -1;    // 顶点法向索引
692.     this.a_Texcoord = -1;    // 顶点纹理坐标索引
693. }
694.
695. // 如果 shader 中没有相关变量则传-1
696. AttribIndex.prototype.init = function(a_Position, a_Normal, a_Texcoord){
697.     this.a_Position = a_Position;   // 顶点坐标索引
698.     this.a_Texcoord = a_Texcoord;   // 顶点法向索引
699.     this.a_Normal = a_Normal;       // 顶点纹理坐标索引
700. }
701.
702. // Material Index 对象
703. // 存放 shader 中对应 uniform 材质变量的索引
704. // 构造函数
705. var MTLIndex = function(){
706.     this.u_Kd = 0;   // 漫反射系数索引
707.     this.u_Ks = 0;   // 镜面反射系数索引
708.     this.u_Ka = 0;   // 环境反射系数索引
709.     this.u_Ke = 0;   // 发射系数索引
710.     this.u_Ns = 0;   // 高光系数索引
711.     this.u_d  = 0;   // 不透明度索引
712. }
713.
714. // 如果 shader 中没有相关变量则传 0
715. MTLIndex.prototype.init = function(u_Kd, u_Ks, u_Ka, u_Ke, u_Ns, u_d){
716.     this.u_Kd = u_Kd;    // 漫反射系数索引
717.     this.u_Ks = u_Ks;    // 镜面反射系数索引
718.     this.u_Ka = u_Ka;    // 环境反射系数索引
719.     this.u_Ke = u_Ke;    // 发射系数索引
```

```
720.     this.u_Ns = u_Ns;     // 高光系数索引
721.     this.u_d  = u_d;      // 不透明度索引
722. }
```

Matrix.js

```
1.  /*This file contains the matrix calculation and creation forms*/
2.  /**
3.   * This is a class treating 4x4 matrix.
4.   * This class contains the function that is equivalent to OpenGL matrix stac
    k.
5.   * The matrix after conversion is calculated by multiplying a conversion mat
    rix from the right.
6.   * The matrix is replaced by the calculated result.
7.   */
8.
9.  /**
10.  * Constructor of Matrix4
11.  * If opt_src is specified, new matrix is initialized by opt_src.
12.  * Otherwise, new matrix is initialized by identity matrix.
13.  * @param opt_src source matrix(option)
14.  */
15. var Matrix4 = function(opt_src) {
16.   var i, s, d;
17.   if (opt_src && typeof opt_src === 'object' && opt_src.hasOwnProperty('elem
    ents')) {
18.     s = opt_src.elements;
19.     d = new Float32Array(16);
20.     for (i = 0; i < 16; ++i) {
21.       d[i] = s[i];
22.     }
23.     this.elements = d;
24.   } else {
25.     this.elements = new Float32Array([1,0,0,0, 0,1,0,0, 0,0,1,0, 0,0,0,1]);

26.   }
27. };
28.
29. /**
30.  * Set the identity matrix.
31.  * @return this
32.  */
33. Matrix4.prototype.setIdentity = function() {
```

```
34.   var e = this.elements;
35.   e[0] = 1;   e[4] = 0;   e[8]  = 0;   e[12] = 0;
36.   e[1] = 0;   e[5] = 1;   e[9]  = 0;   e[13] = 0;
37.   e[2] = 0;   e[6] = 0;   e[10] = 1;   e[14] = 0;
38.   e[3] = 0;   e[7] = 0;   e[11] = 0;   e[15] = 1;
39.   return this;
40. };
41.
42. /**
43.  * Copy matrix.
44.  * @param src source matrix
45.  * @return this
46.  */
47. Matrix4.prototype.set = function(src) {
48.   var i, s, d;
49.
50.   s = src.elements;
51.   d = this.elements;
52.
53.   if (s === d) {
54.     return;
55.   }
56.
57.   for (i = 0; i < 16; ++i) {
58.     d[i] = s[i];
59.   }
60.
61.   return this;
62. };
63.
64. /**
65.  * Multiply the matrix from the right.
66.  * @param other The multiply matrix
67.  * @return this
68.  */
69. Matrix4.prototype.concat = function(other) {
70.   var i, e, a, b, ai0, ai1, ai2, ai3;
71.
72.   // Calculate e = a * b
73.   e = this.elements;
74.   a = this.elements;
75.   b = other.elements;
76.
77.   // If e equals b, copy b to temporary matrix.
```

```
78.   if (e === b) {
79.     b = new Float32Array(16);
80.     for (i = 0; i < 16; ++i) {
81.       b[i] = e[i];
82.     }
83.   }
84.
85.   for (i = 0; i < 4; i++) {
86.     ai0=a[i];  ai1=a[i+4];  ai2=a[i+8];  ai3=a[i+12];
87.     e[i]    = ai0 * b[0]  + ai1 * b[1]  + ai2 * b[2]  + ai3 * b[3];
88.     e[i+4]  = ai0 * b[4]  + ai1 * b[5]  + ai2 * b[6]  + ai3 * b[7];
89.     e[i+8]  = ai0 * b[8]  + ai1 * b[9]  + ai2 * b[10] + ai3 * b[11];
90.     e[i+12] = ai0 * b[12] + ai1 * b[13] + ai2 * b[14] + ai3 * b[15];
91.   }
92.
93.   return this;
94. };
95. Matrix4.prototype.multiply = Matrix4.prototype.concat;
96.
97. /**
98.  * Multiply the three-dimensional vector.
99.  * @param pos  The multiply vector
100.  * @return The result of multiplication(Float32Array)
101.  */
102. Matrix4.prototype.multiplyVector3 = function(pos) {
103.   var e = this.elements;
104.   var p = pos.elements;
105.   var v = new Vector3();
106.   var result = v.elements;
107.
108.   result[0] = p[0] * e[0] + p[1] * e[4] + p[2] * e[ 8] + e[11];
109.   result[1] = p[0] * e[1] + p[1] * e[5] + p[2] * e[ 9] + e[12];
110.   result[2] = p[0] * e[2] + p[1] * e[6] + p[2] * e[10] + e[13];
111.
112.   return v;
113. };
114.
115. /**
116.  * Multiply the four-dimensional vector.
117.  * @param pos  The multiply vector
118.  * @return The result of multiplication(Float32Array)
119.  */
120. Matrix4.prototype.multiplyVector4 = function(pos) {
121.   var e = this.elements;
```

```
122.   var p = pos.elements;
123.   var v = new Vector4();
124.   var result = v.elements;
125.
126.   result[0] = p[0] * e[0] + p[1] * e[4] + p[2] * e[ 8] + p[3] * e[12];
127.   result[1] = p[0] * e[1] + p[1] * e[5] + p[2] * e[ 9] + p[3] * e[13];
128.   result[2] = p[0] * e[2] + p[1] * e[6] + p[2] * e[10] + p[3] * e[14];
129.   result[3] = p[0] * e[3] + p[1] * e[7] + p[2] * e[11] + p[3] * e[15];
130.
131.   return v;
132. };
133.
134. /**
135.  * Transpose the matrix.
136.  * @return this
137.  */
138. Matrix4.prototype.transpose = function() {
139.   var e, t;
140.
141.   e = this.elements;
142.
143.   t = e[ 1];  e[ 1] = e[ 4];  e[ 4] = t;
144.   t = e[ 2];  e[ 2] = e[ 8];  e[ 8] = t;
145.   t = e[ 3];  e[ 3] = e[12];  e[12] = t;
146.   t = e[ 6];  e[ 6] = e[ 9];  e[ 9] = t;
147.   t = e[ 7];  e[ 7] = e[13];  e[13] = t;
148.   t = e[11];  e[11] = e[14];  e[14] = t;
149.
150.   return this;
151. };
152.
153. /**
154.  * Calculate the inverse matrix of specified matrix, and set to this.
155.  * @param other The source matrix
156.  * @return this
157.  */
158. Matrix4.prototype.setInverseOf = function(other) {
159.   var i, s, d, inv, det;
160.
161.   s = other.elements;
162.   d = this.elements;
163.   inv = new Float32Array(16);
164.
165.   inv[0]  =   s[5]*s[10]*s[15] - s[5] *s[11]*s[14] - s[9] *s[6]*s[15]
```

```
            + s[9]*s[7] *s[14] + s[13]*s[6] *s[11] - s[13]*s[7]*s[10];
  inv[4]  = - s[4]*s[10]*s[15] + s[4] *s[11]*s[14] + s[8] *s[6]*s[15]
            - s[8]*s[7] *s[14] - s[12]*s[6] *s[11] + s[12]*s[7]*s[10];
  inv[8]  =   s[4]*s[9] *s[15] - s[4] *s[11]*s[13] - s[8] *s[5]*s[15]
            + s[8]*s[7] *s[13] + s[12]*s[5] *s[11] - s[12]*s[7]*s[9];
  inv[12] = - s[4]*s[9] *s[14] + s[4] *s[10]*s[13] + s[8] *s[5]*s[14]
            - s[8]*s[6] *s[13] - s[12]*s[5] *s[10] + s[12]*s[6]*s[9];

  inv[1]  = - s[1]*s[10]*s[15] + s[1] *s[11]*s[14] + s[9] *s[2]*s[15]
            - s[9]*s[3] *s[14] - s[13]*s[2] *s[11] + s[13]*s[3]*s[10];
  inv[5]  =   s[0]*s[10]*s[15] - s[0] *s[11]*s[14] - s[8] *s[2]*s[15]
            + s[8]*s[3] *s[14] + s[12]*s[2] *s[11] - s[12]*s[3]*s[10];
  inv[9]  = - s[0]*s[9] *s[15] + s[0] *s[11]*s[13] + s[8] *s[1]*s[15]
            - s[8]*s[3] *s[13] - s[12]*s[1] *s[11] + s[12]*s[3]*s[9];
  inv[13] =   s[0]*s[9] *s[14] - s[0] *s[10]*s[13] - s[8] *s[1]*s[14]
            + s[8]*s[2] *s[13] + s[12]*s[1] *s[10] - s[12]*s[2]*s[9];

  inv[2]  =   s[1]*s[6]*s[15] - s[1] *s[7]*s[14] - s[5] *s[2]*s[15]
            + s[5]*s[3]*s[14] + s[13]*s[2]*s[7]  - s[13]*s[3]*s[6];
  inv[6]  = - s[0]*s[6]*s[15] + s[0] *s[7]*s[14] + s[4] *s[2]*s[15]
            - s[4]*s[3]*s[14] - s[12]*s[2]*s[7]  + s[12]*s[3]*s[6];
  inv[10] =   s[0]*s[5]*s[15] - s[0] *s[7]*s[13] - s[4] *s[1]*s[15]
            + s[4]*s[3]*s[13] + s[12]*s[1]*s[7]  - s[12]*s[3]*s[5];
  inv[14] = - s[0]*s[5]*s[14] + s[0] *s[6]*s[13] + s[4] *s[1]*s[14]
            - s[4]*s[2]*s[13] - s[12]*s[1]*s[6]  + s[12]*s[2]*s[5];

  inv[3]  = - s[1]*s[6]*s[11] + s[1]*s[7]*s[10] + s[5]*s[2]*s[11]
            - s[5]*s[3]*s[10] - s[9]*s[2]*s[7]  + s[9]*s[3]*s[6];
  inv[7]  =   s[0]*s[6]*s[11] - s[0]*s[7]*s[10] - s[4]*s[2]*s[11]
            + s[4]*s[3]*s[10] + s[8]*s[2]*s[7]  - s[8]*s[3]*s[6];
  inv[11] = - s[0]*s[5]*s[11] + s[0]*s[7]*s[9]  + s[4]*s[1]*s[11]
            - s[4]*s[3]*s[9]  - s[8]*s[1]*s[7]  + s[8]*s[3]*s[5];
  inv[15] =   s[0]*s[5]*s[10] - s[0]*s[6]*s[9]  - s[4]*s[1]*s[10]
            + s[4]*s[2]*s[9]  + s[8]*s[1]*s[6]  - s[8]*s[2]*s[5];

  det = s[0]*inv[0] + s[1]*inv[4] + s[2]*inv[8] + s[3]*inv[12];
  if (det === 0) {
    return this;
  }

  det = 1 / det;
  for (i = 0; i < 16; i++) {
    d[i] = inv[i] * det;
  }
```

```
210.
211.   return this;
212. };
213.
214. /**
215.  * Calculate the inverse matrix of this, and set to this.
216.  * @return this
217.  */
218. Matrix4.prototype.invert = function() {
219.   return this.setInverseOf(this);
220. };
221.
222. /**
223.  * Set the orthographic projection matrix.
224.  * @param left The coordinate of the left of clipping plane.
225.  * @param right The coordinate of the right of clipping plane.
226.  * @param bottom The coordinate of the bottom of clipping plane.
227.  * @param top The coordinate of the top top clipping plane.
228.  * @param near The distances to the nearer depth clipping plane. This value
    is minus if the plane is to be behind the viewer.
229.  * @param far The distances to the farther depth clipping plane. This value
    is minus if the plane is to be behind the viewer.
230.  * @return this
231.  */
232. Matrix4.prototype.setOrtho = function(left, right, bottom, top, near, far)
   {
233.   var e, rw, rh, rd;
234.
235.   if (left === right || bottom === top || near === far) {
236.     throw 'null frustum';
237.   }
238.
239.   rw = 1 / (right - left);
240.   rh = 1 / (top - bottom);
241.   rd = 1 / (far - near);
242.
243.   e = this.elements;
244.
245.   e[0]  = 2 * rw;
246.   e[1]  = 0;
247.   e[2]  = 0;
248.   e[3]  = 0;
249.
250.   e[4]  = 0;
```

```
251.   e[5]  = 2 * rh;
252.   e[6]  = 0;
253.   e[7]  = 0;
254.
255.   e[8]  = 0;
256.   e[9]  = 0;
257.   e[10] = -2 * rd;
258.   e[11] = 0;
259.
260.   e[12] = -(right + left) * rw;
261.   e[13] = -(top + bottom) * rh;
262.   e[14] = -(far + near) * rd;
263.   e[15] = 1;
264.
265.   return this;
266. };
267.
268. /**
269.  * Multiply the orthographic projection matrix from the right.
270.  * @param left The coordinate of the left of clipping plane.
271.  * @param right The coordinate of the right of clipping plane.
272.  * @param bottom The coordinate of the bottom of clipping plane.
273.  * @param top The coordinate of the top top clipping plane.
274.  * @param near The distances to the nearer depth clipping plane. This value
    is minus if the plane is to be behind the viewer.
275.  * @param far The distances to the farther depth clipping plane. This value
    is minus if the plane is to be behind the viewer.
276.  * @return this
277.  */
278. Matrix4.prototype.ortho = function(left, right, bottom, top, near, far) {
279.   return this.concat(new Matrix4().setOrtho(left, right, bottom, top, near,
    far));
280. };
281.
282. /**
283.  * Set the perspective projection matrix.
284.  * @param left The coordinate of the left of clipping plane.
285.  * @param right The coordinate of the right of clipping plane.
286.  * @param bottom The coordinate of the bottom of clipping plane.
287.  * @param top The coordinate of the top top clipping plane.
288.  * @param near The distances to the nearer depth clipping plane. This value
    must be plus value.
289.  * @param far The distances to the farther depth clipping plane. This value
    must be plus value.
```

```
290.  * @return this
291.  */
292. Matrix4.prototype.setFrustum = function(left, right, bottom, top, near, far
    ) {
293.   var e, rw, rh, rd;
294.
295.   if (left === right || top === bottom || near === far) {
296.     throw 'null frustum';
297.   }
298.   if (near <= 0) {
299.     throw 'near <= 0';
300.   }
301.   if (far <= 0) {
302.     throw 'far <= 0';
303.   }
304.
305.   rw = 1 / (right - left);
306.   rh = 1 / (top - bottom);
307.   rd = 1 / (far - near);
308.
309.   e = this.elements;
310.
311.   e[ 0] = 2 * near * rw;
312.   e[ 1] = 0;
313.   e[ 2] = 0;
314.   e[ 3] = 0;
315.
316.   e[ 4] = 0;
317.   e[ 5] = 2 * near * rh;
318.   e[ 6] = 0;
319.   e[ 7] = 0;
320.
321.   e[ 8] = (right + left) * rw;
322.   e[ 9] = (top + bottom) * rh;
323.   e[10] = -(far + near) * rd;
324.   e[11] = -1;
325.
326.   e[12] = 0;
327.   e[13] = 0;
328.   e[14] = -2 * near * far * rd;
329.   e[15] = 0;
330.
331.   return this;
332. };
```

```
333.
334. /**
335.  * Multiply the perspective projection matrix from the right.
336.  * @param left The coordinate of the left of clipping plane.
337.  * @param right The coordinate of the right of clipping plane.
338.  * @param bottom The coordinate of the bottom of clipping plane.
339.  * @param top The coordinate of the top top clipping plane.
340.  * @param near The distances to the nearer depth clipping plane. This value
    must be plus value.
341.  * @param far The distances to the farther depth clipping plane. This value
    must be plus value.
342.  * @return this
343.  */
344. Matrix4.prototype.frustum = function(left, right, bottom, top, near, far) {

345.   return this.concat(new Matrix4().setFrustum(left, right, bottom, top, nea
   r, far));
346. };
347.
348. /**
349.  * Set the perspective projection matrix by fovy and aspect.
350.  * @param fovy The angle between the upper and lower sides of the frustum.
351.  * @param aspect The aspect ratio of the frustum. (width/height)
352.  * @param near The distances to the nearer depth clipping plane. This value
    must be plus value.
353.  * @param far The distances to the farther depth clipping plane. This value
    must be plus value.
354.  * @return this
355.  */
356. Matrix4.prototype.setPerspective = function(fovy, aspect, near, far) {
357.   var e, rd, s, ct;
358.
359.   if (near === far || aspect === 0) {
360.     throw 'null frustum';
361.   }
362.   if (near <= 0) {
363.     throw 'near <= 0';
364.   }
365.   if (far <= 0) {
366.     throw 'far <= 0';
367.   }
368.
369.   fovy = Math.PI * fovy / 180 / 2;
370.   s = Math.sin(fovy);
```

```
371.   if (s === 0) {
372.     throw 'null frustum';
373.   }
374.
375.   rd = 1 / (far - near);
376.   ct = Math.cos(fovy) / s;
377.
378.   e = this.elements;
379.
380.   e[0]  = ct / aspect;
381.   e[1]  = 0;
382.   e[2]  = 0;
383.   e[3]  = 0;
384.
385.   e[4]  = 0;
386.   e[5]  = ct;
387.   e[6]  = 0;
388.   e[7]  = 0;
389.
390.   e[8]  = 0;
391.   e[9]  = 0;
392.   e[10] = -(far + near) * rd;
393.   e[11] = -1;
394.
395.   e[12] = 0;
396.   e[13] = 0;
397.   e[14] = -2 * near * far * rd;
398.   e[15] = 0;
399.
400.   return this;
401. };
402.
403. /**
404.  * Multiply the perspective projection matrix from the right.
405.  * @param fovy The angle between the upper and lower sides of the frustum.
406.  * @param aspect The aspect ratio of the frustum. (width/height)
407.  * @param near The distances to the nearer depth clipping plane. This value
     must be plus value.
408.  * @param far The distances to the farther depth clipping plane. This value
     must be plus value.
409.  * @return this
410.  */
411. Matrix4.prototype.perspective = function(fovy, aspect, near, far) {
```

```
412.   return this.concat(new Matrix4().setPerspective(fovy, aspect, near, far))
    ;
413. };
414.
415. /**
416.  * Set the matrix for scaling.
417.  * @param x The scale factor along the X axis
418.  * @param y The scale factor along the Y axis
419.  * @param z The scale factor along the Z axis
420.  * @return this
421.  */
422. Matrix4.prototype.setScale = function(x, y, z) {
423.   var e = this.elements;
424.   e[0] = x;  e[4] = 0;  e[8]  = 0;  e[12] = 0;
425.   e[1] = 0;  e[5] = y;  e[9]  = 0;  e[13] = 0;
426.   e[2] = 0;  e[6] = 0;  e[10] = z;  e[14] = 0;
427.   e[3] = 0;  e[7] = 0;  e[11] = 0;  e[15] = 1;
428.   return this;
429. };
430.
431. /**
432.  * Multiply the matrix for scaling from the right.
433.  * @param x The scale factor along the X axis
434.  * @param y The scale factor along the Y axis
435.  * @param z The scale factor along the Z axis
436.  * @return this
437.  */
438. Matrix4.prototype.scale = function(x, y, z) {
439.   var e = this.elements;
440.   e[0] *= x;  e[4] *= y;  e[8]  *= z;
441.   e[1] *= x;  e[5] *= y;  e[9]  *= z;
442.   e[2] *= x;  e[6] *= y;  e[10] *= z;
443.   e[3] *= x;  e[7] *= y;  e[11] *= z;
444.   return this;
445. };
446.
447. /**
448.  * Set the matrix for translation.
449.  * @param x The X value of a translation.
450.  * @param y The Y value of a translation.
451.  * @param z The Z value of a translation.
452.  * @return this
453.  */
454. Matrix4.prototype.setTranslate = function(x, y, z) {
```

```
455.   var e = this.elements;
456.   e[0] = 1;  e[4] = 0;  e[8]  = 0;  e[12] = x;
457.   e[1] = 0;  e[5] = 1;  e[9]  = 0;  e[13] = y;
458.   e[2] = 0;  e[6] = 0;  e[10] = 1;  e[14] = z;
459.   e[3] = 0;  e[7] = 0;  e[11] = 0;  e[15] = 1;
460.   return this;
461. };
462. 
463. /**
464.  * Multiply the matrix for translation from the right.
465.  * @param x The X value of a translation.
466.  * @param y The Y value of a translation.
467.  * @param z The Z value of a translation.
468.  * @return this
469.  */
470. Matrix4.prototype.translate = function(x, y, z) {
471.   var e = this.elements;
472.   e[12] += e[0] * x + e[4] * y + e[8]  * z;
473.   e[13] += e[1] * x + e[5] * y + e[9]  * z;
474.   e[14] += e[2] * x + e[6] * y + e[10] * z;
475.   e[15] += e[3] * x + e[7] * y + e[11] * z;
476.   return this;
477. };
478. 
479. /**
480.  * Set the matrix for rotation.
481.  * The vector of rotation axis may not be normalized.
482.  * @param angle The angle of rotation (degrees)
483.  * @param x The X coordinate of vector of rotation axis.
484.  * @param y The Y coordinate of vector of rotation axis.
485.  * @param z The Z coordinate of vector of rotation axis.
486.  * @return this
487.  */
488. Matrix4.prototype.setRotate = function(angle, x, y, z) {
489.   var e, s, c, len, rlen, nc, xy, yz, zx, xs, ys, zs;
490. 
491.   angle = Math.PI * angle / 180;
492.   e = this.elements;
493. 
494.   s = Math.sin(angle);
495.   c = Math.cos(angle);
496. 
497.   if (0 !== x && 0 === y && 0 === z) {
498.     // Rotation around X axis
```

```
    if (x < 0) {
      s = -s;
    }
    e[0] = 1;  e[4] = 0;  e[ 8] = 0;  e[12] = 0;
    e[1] = 0;  e[5] = c;  e[ 9] =-s;  e[13] = 0;
    e[2] = 0;  e[6] = s;  e[10] = c;  e[14] = 0;
    e[3] = 0;  e[7] = 0;  e[11] = 0;  e[15] = 1;
  } else if (0 === x && 0 !== y && 0 === z) {
    // Rotation around Y axis
    if (y < 0) {
      s = -s;
    }
    e[0] = c;  e[4] = 0;  e[ 8] = s;  e[12] = 0;
    e[1] = 0;  e[5] = 1;  e[ 9] = 0;  e[13] = 0;
    e[2] =-s;  e[6] = 0;  e[10] = c;  e[14] = 0;
    e[3] = 0;  e[7] = 0;  e[11] = 0;  e[15] = 1;
  } else if (0 === x && 0 === y && 0 !== z) {
    // Rotation around Z axis
    if (z < 0) {
      s = -s;
    }
    e[0] = c;  e[4] =-s;  e[ 8] = 0;  e[12] = 0;
    e[1] = s;  e[5] = c;  e[ 9] = 0;  e[13] = 0;
    e[2] = 0;  e[6] = 0;  e[10] = 1;  e[14] = 0;
    e[3] = 0;  e[7] = 0;  e[11] = 0;  e[15] = 1;
  } else {
    // Rotation around another axis
    len = Math.sqrt(x*x + y*y + z*z);
    if (len !== 1) {
      rlen = 1 / len;
      x *= rlen;
      y *= rlen;
      z *= rlen;
    }
    nc = 1 - c;
    xy = x * y;
    yz = y * z;
    zx = z * x;
    xs = x * s;
    ys = y * s;
    zs = z * s;

    e[ 0] = x*x*nc +  c;
    e[ 1] = xy *nc + zs;
```

```
543.     e[ 2] = zx *nc - ys;
544.     e[ 3] = 0;
545.
546.     e[ 4] = xy *nc - zs;
547.     e[ 5] = y*y*nc +  c;
548.     e[ 6] = yz *nc + xs;
549.     e[ 7] = 0;
550.
551.     e[ 8] = zx *nc + ys;
552.     e[ 9] = yz *nc - xs;
553.     e[10] = z*z*nc +  c;
554.     e[11] = 0;
555.
556.     e[12] = 0;
557.     e[13] = 0;
558.     e[14] = 0;
559.     e[15] = 1;
560.   }
561.
562.   return this;
563. };
564.
565. /**
566.  * Multiply the matrix for rotation from the right.
567.  * The vector of rotation axis may not be normalized.
568.  * @param angle The angle of rotation (degrees)
569.  * @param x The X coordinate of vector of rotation axis.
570.  * @param y The Y coordinate of vector of rotation axis.
571.  * @param z The Z coordinate of vector of rotation axis.
572.  * @return this
573.  */
574. Matrix4.prototype.rotate = function(angle, x, y, z) {
575.   return this.concat(new Matrix4().setRotate(angle, x, y, z));
576. };
577.
578. /**
579.  * Set the viewing matrix.
580.  * @param eyeX, eyeY, eyeZ The position of the eye point.
581.  * @param centerX, centerY, centerZ The position of the reference point.
582.  * @param upX, upY, upZ The direction of the up vector.
583.  * @return this
584.  */
585. Matrix4.prototype.setLookAt = function(eyeX, eyeY, eyeZ, centerX, centerY,
    centerZ, upX, upY, upZ) {
```

```
586.   var e, fx, fy, fz, rlf, sx, sy, sz, rls, ux, uy, uz;
587.
588.   fx = centerX - eyeX;
589.   fy = centerY - eyeY;
590.   fz = centerZ - eyeZ;
591.
592.   // Normalize f.
593.   rlf = 1 / Math.sqrt(fx*fx + fy*fy + fz*fz);
594.   fx *= rlf;
595.   fy *= rlf;
596.   fz *= rlf;
597.
598.   // Calculate cross product of f and up.
599.   sx = fy * upZ - fz * upY;
600.   sy = fz * upX - fx * upZ;
601.   sz = fx * upY - fy * upX;
602.
603.   // Normalize s.
604.   rls = 1 / Math.sqrt(sx*sx + sy*sy + sz*sz);
605.   sx *= rls;
606.   sy *= rls;
607.   sz *= rls;
608.
609.   // Calculate cross product of s and f.
610.   ux = sy * fz - sz * fy;
611.   uy = sz * fx - sx * fz;
612.   uz = sx * fy - sy * fx;
613.
614.   // Set to this.
615.   e = this.elements;
616.   e[0] = sx;
617.   e[1] = ux;
618.   e[2] = -fx;
619.   e[3] = 0;
620.
621.   e[4] = sy;
622.   e[5] = uy;
623.   e[6] = -fy;
624.   e[7] = 0;
625.
626.   e[8] = sz;
627.   e[9] = uz;
628.   e[10] = -fz;
629.   e[11] = 0;
```

```
630.
631.   e[12] = 0;
632.   e[13] = 0;
633.   e[14] = 0;
634.   e[15] = 1;
635.
636.   // Translate.
637.   return this.translate(-eyeX, -eyeY, -eyeZ);
638. };
639.
640. /**
641.  * Multiply the viewing matrix from the right.
642.  * @param eyeX, eyeY, eyeZ The position of the eye point.
643.  * @param centerX, centerY, centerZ The position of the reference point.
644.  * @param upX, upY, upZ The direction of the up vector.
645.  * @return this
646.  */
647. Matrix4.prototype.lookAt = function(eyeX, eyeY, eyeZ, centerX, centerY, cen
    terZ, upX, upY, upZ) {
648.   return this.concat(new Matrix4().setLookAt(eyeX, eyeY, eyeZ, centerX, cen
    terY, centerZ, upX, upY, upZ));
649. };
650.
651. /**
652.  * Multiply the matrix for project vertex to plane from the right.
653.  * @param plane The array[A, B, C, D] of the equation of plane "Ax + By + C
    z + D = 0".
654.  * @param light The array which stored coordinates of the light. if light[3
    ]=0, treated as parallel light.
655.  * @return this
656.  */
657. Matrix4.prototype.dropShadow = function(plane, light) {
658.   var mat = new Matrix4();
659.   var e = mat.elements;
660.
661.   var dot = plane[0] * light[0] + plane[1] * light[1] + plane[2] * light[2]
     + plane[3] * light[3];
662.
663.   e[ 0] = dot - light[0] * plane[0];
664.   e[ 1] =     - light[1] * plane[0];
665.   e[ 2] =     - light[2] * plane[0];
666.   e[ 3] =     - light[3] * plane[0];
667.
668.   e[ 4] =     - light[0] * plane[1];
```

```
669.   e[ 5] = dot - light[1] * plane[1];
670.   e[ 6] =     - light[2] * plane[1];
671.   e[ 7] =     - light[3] * plane[1];
672.
673.   e[ 8] =     - light[0] * plane[2];
674.   e[ 9] =     - light[1] * plane[2];
675.   e[10] = dot - light[2] * plane[2];
676.   e[11] =     - light[3] * plane[2];
677.
678.   e[12] =     - light[0] * plane[3];
679.   e[13] =     - light[1] * plane[3];
680.   e[14] =     - light[2] * plane[3];
681.   e[15] = dot - light[3] * plane[3];
682.
683.   return this.concat(mat);
684. }
685.
686. /**
687.  * Multiply the matrix for project vertex to plane from the right.(Projecte
    d by parallel light.)
688.  * @param normX, normY, normZ The normal vector of the plane.(Not necessary
     to be normalized.)
689.  * @param planeX, planeY, planeZ The coordinate of arbitrary points on a pl
    ane.
690.  * @param lightX, lightY, lightZ The vector of the direction of light.(Not
    necessary to be normalized.)
691.  * @return this
692.  */
693. Matrix4.prototype.dropShadowDirectionally = function(normX, normY, normZ, p
    laneX, planeY, planeZ, lightX, lightY, lightZ) {
694.   var a = planeX * normX + planeY * normY + planeZ * normZ;
695.   return this.dropShadow([normX, normY, normZ, -
    a], [lightX, lightY, lightZ, 0]);
696. };
697.
698. /**
699.  * Constructor of Vector3
700.  * If opt_src is specified, new vector is initialized by opt_src.
701.  * @param opt_src source vector(option)
702.  */
703. var Vector3 = function(opt_src) {
704.   var v = new Float32Array(3);
705.   if (opt_src && typeof opt_src === 'object') {
706.     v[0] = opt_src[0]; v[1] = opt_src[1]; v[2] = opt_src[2];
```

```
707.   }
708.   this.elements = v;
709. }
710.
711. /**
712.   * Normalize.
713.   * @return this
714.   */
715. Vector3.prototype.normalize = function() {
716.   var v = this.elements;
717.   var c = v[0], d = v[1], e = v[2], g = Math.sqrt(c*c+d*d+e*e);
718.   if(g){
719.     if(g == 1)
720.         return this;
721.    } else {
722.      v[0] = 0; v[1] = 0; v[2] = 0;
723.      return this;
724.    }
725.    g = 1/g;
726.    v[0] = c*g; v[1] = d*g; v[2] = e*g;
727.    return this;
728. };
729.
730. /**
731.  * Constructor of Vector4
732.  * If opt_src is specified, new vector is initialized by opt_src.
733.  * @param opt_src source vector(option)
734.  */
735. var Vector4 = function(opt_src) {
736.   var v = new Float32Array(4);
737.   if (opt_src && typeof opt_src === 'object') {
738.     v[0] = opt_src[0]; v[1] = opt_src[1]; v[2] = opt_src[2]; v[3] = opt_src
    [3];
739.   }
740.   this.elements = v;
741. }
```

index.js

```
1.  var canvas;
2.  var gl;
3.  var program;
4.
```

```
5.  //相机位移
6.  var dy = 20.0;
7.  var dz = -80.0;
8.  var dx = 0.0;
9.  //相机镜头位移
10. var currentAngle = [0.0, 0.0];
11.
12. //场景缩放
13. var scaleRatio = 1.0;
14.
15. //鼠标控制相关参数
16. var dragging = false;
17. var lastX = -1,
18.   lastY = -1;
19. //投影矩阵
20. var matProj;
21. //模视矩阵
22. //var modelMatrix_each = new Matrix4();
23. //modelMatrix_each.type = 'mat4';
24.
25. // shader 中变量的索引
26. var attribIndex = new AttribIndex(); // shader 中 attribute 变量索引
27. var mtlIndex = new MTLIndex(); // shader 中材质变量索引
28. var u_ModelView; // shader 中 uniform 变量"u_ModelView"的索引
29. var u_Sampler; // shader 中 uniform 变量"u_Sampler"的索引
30.
31. //光源定义
32. var lightPosition = vec4(0, 0, 20, 1); // 光源位置(观察.o 坐标系)
33. var ambientLight = vec3(0.3, 0.3, 0.3); // 环境光
34. var diffuseLight = vec3(0.5, 0.5, 0.5); // 漫反射光
35. var specularLight = vec3(0.8, 0.8, 0.8); // 镜面反射光
36.
37. window.onload = function main() {
38.   canvas = document.getElementById('webgl');
39.   gl = WebGLUtils.setupWebGL(canvas);
40.   if (!gl) {
41.     console.log('Failed to get the rendering context for WebGL');
42.     return;
43.   }
44.
45.   /*设置 WebGL 相关属性*/
46.   // 清颜色缓存和深度缓存
47.   gl.clearColor(0.0, 0.0, 0.0, 1.0);
48.   gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
```

```
49.   gl.enable(gl.DEPTH_TEST);
50.   gl.enable(gl.CULL_FACE); //开启面剔除，默认剔除背面
51.   // 设置视口，占满整个 canvas
52.   gl.viewport(0, 0, canvas.width, canvas.height);
53.   // 设置投影矩阵：透视投影，根据视口宽高比指定视域体
54.   matProj = perspective(45.0, // 垂直方向视角
55.     canvas.width / canvas.height, // 视域体宽高比
56.     1.0, // 相机到近裁剪面距离
57.     1000.0); // 相机到远裁剪面距离
58.
59.   program = initShaders(gl, "vertex-shader", "fragment-shader");
60.   gl.useProgram(program); // 启用该 shader 程序对象
61.
62.   initMouseEvent();
63.   initEventHandlers(canvas, currentAngle);
64.   renderall();
65. }
66.
67. function renderall() {
68.   // 进行绘制
69.   var tick = function() {
70.     if (!prepare())
71.       {
72.         requestAnimFrame(tick);
73.       }
74.     else {
75.         gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
76.         drawscene();
77.     }
78.   }
79.   tick();
80. }
81. //-----画模型于指定坐标-------------------------------------------------------
    -------------
82. function loadmodel(obj, position) {
83.
84.   // 获取 shader attribute 变量的位置
85.   var a_Position = gl.getAttribLocation(program, "a_Position");
86.   var a_Normal = gl.getAttribLocation(program, "a_Normal");
87.   var a_TexCoord = gl.getAttribLocation(program, "a_TexCoord");
88.   if (a_Position < 0 || a_Normal < 0 || a_TexCoord < 0) { // getAttribLocati
    on 获取失败则返回-1
89.     alert("获取 attribute 变量失败！");
90.     return;
```

```
91.  }
92.
93.  // 初始化 attribIndex
94.  // 注意顺序不要错，分别为顶点坐标、法向和纹理坐标的索引
95.  // 如果 shader 中没有相关变量则传-1
96.  attribIndex.init(a_Position, a_Normal, a_TexCoord);
97.
98.  u_ModelView = gl.getUniformLocation(program, "u_ModelView");
99.  var u_Projection = gl.getUniformLocation(program, "u_Projection");
100.  var u_LightPosition = gl.getUniformLocation(program, "u_LightPosition");

101.  if (!u_ModelView || !u_Projection || !u_LightPosition) {
102.    alert("获取 uniform 变量失败！")
103.    return;
104.  }
105.
106.  gl.uniformMatrix4fv(u_Projection, false, flatten(matProj));
107.  gl.uniform4fv(u_LightPosition, flatten(lightPosition));
108.
109.  var u_AmbientLight = gl.getUniformLocation(program, "u_AmbientLight");
110.  var u_DiffuseLight = gl.getUniformLocation(program, "u_DiffuseLight");
111.  var u_SpecularLight = gl.getUniformLocation(program, "u_SpecularLight");

112.  if (!u_AmbientLight || !u_DiffuseLight || !u_SpecularLight) {
113.    alert("获取光相关 uniform 变量失败！")
114.    return;
115.  }
116.
117.  // 给光源的三种光颜色传值
118.  gl.uniform3fv(u_AmbientLight, flatten(ambientLight));
119.  gl.uniform3fv(u_DiffuseLight, flatten(diffuseLight));
120.  gl.uniform3fv(u_SpecularLight, flatten(specularLight));
121.
122.  // 获取 shader 中 uniform 材质变量索引
123.  var u_Kd = gl.getUniformLocation(program, "u_Kd");
124.  var u_Ks = gl.getUniformLocation(program, "u_Ks");
125.  var u_Ka = gl.getUniformLocation(program, "u_Ka");
126.  var u_Ke = gl.getUniformLocation(program, "u_Ke");
127.  var u_Ns = gl.getUniformLocation(program, "u_Ns");
128.  var u_d = gl.getUniformLocation(program, "u_d");
129.  if (!u_Kd || !u_Ks || !u_Ka || !u_Ke || !u_Ns || !u_d) {
130.    alert("获取 uniform 材质变量失败！")
131.    return;
132.  }
```

```
133.
134.   // 初始化 mtlIndex，参数顺序不要错！
135.   // 依次为漫反射系数索引、镜面反射系数索引、环境反射系数索引、
136.   // 发射系数索引、高光系数、不透明度
137.   // 如果 shader 中没有相关变量则传 0
138.   mtlIndex.init(u_Kd, u_Ks, u_Ka, u_Ke, u_Ns, u_d);
139.
140.   // 获取名称为"u_Sampler"的 shader uniform 变量位置
141.   u_Sampler = gl.getUniformLocation(program, "u_Sampler");
142.   if (!u_Sampler) {
143.     alert("获取 uniform 变量 u_Sampler 失败！")
144.     return;
145.   }
146.
147.   render(obj, position);
148. };
149.
150.
151. // 绘制函数
152. function render(obj, position) {
153.   // 创建变换矩阵
154.   matModelView =
155.   mult(translate(dx, dy, dz),
156.     mult(rotateX(currentAngle[0]),
157.       mult(rotateY(currentAngle[1]),
158.           mult(scale(scaleRatio,scaleRatio,scaleRatio),
159.              translate(position[0], position[1], position[2])
160.         ))))
161.   ;
162.
163.   // 传值给 shader 中的 u_ModelView
164.   gl.uniformMatrix4fv(u_ModelView, false, flatten(matModelView));
165.
166.   // 绘制 obj 模型
167.   obj.draw(gl, attribIndex, mtlIndex, u_Sampler);
168. }
169.
170.
171. var g_matrixStack = []; // Array for storing a matrix
172. function pushMatrix(m) { // Store the specified matrix to the array
173.   var m2 = new Matrix4(m);
174.   g_matrixStack.push(m2);
175. }
176.
```

```
177. function popMatrix() { // Retrieve the matrix from the array
178.   return g_matrixStack.pop();
179. }
180.
181. //注册鼠标事件
182. function initMouseEvent() {
183.   canvas.addEventListener('DOMMouseScroll', wheelHandler, false);
184.   canvas.addEventListener('mousewheel', wheelHandler, false);
185. }
186.
187. function wheelHandler(ev){
188.   if(ev.wheelDelta> 0){
189.       scaleRatio += ev.wheelDelta/120/20;
190.   }
191.   else if(scaleRatio + ev.wheelDelta/120/20 > 0){
192.     scaleRatio += ev.wheelDelta/120/20;
193.   }
194.   requestAnimFrame(renderall()); // 请求重绘
195. }
196.
197. function initEventHandlers(canvas, currentAngle) {
198.   var dragging = false;
199.   var lastX = -1,
200.     lastY = -1;
201.   canvas.onmousedown = function(ev) {
202.     var x = ev.clientX,
203.       y = ev.clientY;
204.     var rect = ev.target.getBoundingClientRect();
205.     if (rect.left <= x && x < rect.right && rect.top <= y && y < rect.botto
    m) {
206.       lastX = x;
207.       lastY = y;
208.       dragging = true;
209.     }
210.   };
211.
212.   canvas.onmouseup = function(ev) {
213.     dragging = false;
214.   };
215.
216.   canvas.onmousemove = function(ev) { // Mouse is moved
217.     var x = ev.clientX,
218.       y = ev.clientY;
219.     if (dragging) {
```

```
220.       var factor = 60 / canvas.height; // The rotation ratio
221.       var dx = factor * (x - lastX);
222.       var dy = factor * (y - lastY);
223.       // Limit x-axis rotation angle to -90 to 90 degrees
224.       currentAngle[0] = Math.max(Math.min(currentAngle[0] + dy, 90.0), -
    90.0);
225.       currentAngle[1] = currentAngle[1] + dx;
226.     }
227.     lastX = x, lastY = y;
228.     requestAnimFrame(renderall()); // 请求重绘
229.   };
230. }
231.
232. // 按键响应
233. // 用于控制视角
234. window.onkeydown = function() {
235.   switch (event.keyCode) {
236.     case 65: // 按键 A left
237.       dx += 1.0;
238.       break;
239.     case 87: // 按键 W straightword
240.       dz += 1.0;
241.       break;
242.     case 68: // 按键 D right
243.       dx += -1.0;
244.       break;
245.     case 83: // 按键 S backword
246.       dz += -1.0;
247.       break;
248.     case 81: //按键 Q up
249.       dy += -1.0;
250.       break;
251.     case 69: //按键 E down
252.       dy += 1.0;
253.       break;
254.     default:
255.       return;
256.   }
257.   requestAnimFrame(renderall()); // 请求重绘
258. }
```

drawscene.js

```
1.  //drawscene.js
2.  /*Load all the model and define their locations*/
3.  var tree = loadOBJ(".\\model\\tree\\tree.obj");
4.  var tree2 = loadOBJ(".\\model\\tree2\\tree2.obj");
5.  var Saber = loadOBJ(".\\model\\Saber\\Saber.obj");
6.  var mountain = loadOBJ(".\\model\\mountain\\mountain.obj");
7.  var building2 = loadOBJ(".\\model\\building2\\building2.obj");
8.  var building3 = loadOBJ(".\\model\\building3\\building3.obj");
9.  var qingke = loadOBJ(".\\model\\qingKe0\\qingKe0.obj");
10. var lan = loadOBJ(".\\model\\lan\\lan.obj");
11. var road0 = loadOBJ(".\\model\\road0\\road0.obj");
12. var ren = loadOBJ(".\\model\\ren\\ren.obj");
13. var objects = [tree,tree2,Saber,mountain,building2,building3,
14.                qingke,lan,road0,ren];
15. /*Load all the model and define their locations*/
16. function drawscene(){
17.     //plants           x      z      y
18.     loadmodel(tree, [-60.0, -53.0, 130.0]);//the small one
19.     //plants           x      z      y
20.     loadmodel(tree, [-60.0, -53.0, 130.0]);//the small one
21.     loadmodel(tree, [-60.0, -53.0, 170.0]);//the small one
22.     loadmodel(tree, [50.0, -53.0, 130.0]);//the small one
23.     loadmodel(tree, [50.0, -53.0, 170.0]);//the small one
24.     //buildings
25.     loadmodel(mountain, [0.0, 0.0, 0.0]);
26.     loadmodel(building2, [0.0, -53.0, 65.0]);  //the longer one
27.     loadmodel(building2, [0.0, -53.0, -65.0]);  //the longer one
28.     loadmodel(building3, [-60.0, -56.0, 0.0]);
29.
30.      //房子前面的路
31.     for(var i=-30+1;i<=30.0+1;i+=20.0)
32.       for(var j=-8.0+1.2;j<=8.0+1.2;j+=16)
33.       loadmodel(tree2, [i, -56.0, j]);//the large green
34.     for(var i=-30+1;i<=30.0+1;i+=10.0)
35.       for(var j=-6.0+1.2;j<=6.0+1.2;j+=3){
36.         loadmodel(road0,[i,-54.8,j]);
37.       }
38.     loadmodel(Saber, [29.0, -55.0, 0.0]);
39.
40.     //房子右边的篱笆和草和稻草人
41.     loadmodel(lan, [-60.0, -53.0, -155.0]);//栏杆
42.     loadmodel(lan, [-20.0, -53.0, -155.0]);
43.     loadmodel(lan, [-45.0, -53.0, -155.0]);
44.     loadmodel(lan, [20.0, -53.0, -155.0]);
```

```
    loadmodel(lan, [45.0, -53.0, -155.0]);
    loadmodel(lan, [60.0, -53.0, -155.0]);
      for(var i=0;i<28+28;i+=4)
        for(var j=0;j<70;j+=7)
         for(var z=0;z<=0.5;z+=0.2)
            for(var c=0;c<=0.5;c+=0.3)
             loadmodel(qingke,[i-75, -47.0, j-220]);
    //稻草人
    loadmodel(ren, [-20.0, -50.0, -170.0]);
    loadmodel(ren, [0.0, -50.0, -180.0]);
};
//加载所有模型后,绘制
function prepare() {
  var len=objects.length;
  for (var i=0 ;i<len ;i++ )
  {
    if(!objects[i].isAllReady(gl))
      return 0;
  }
  return 1;
};
```

initShaders.js

```
//  initShaders.js
/*
initalize Shaders and attach all the parameters
@parameter:gl
@parameter:vertexShaderId
@parameter:fragmentShaderId
*/

function initShaders( gl, vertexShaderId, fragmentShaderId )
{
    var vertShdr;
    var fragShdr;

    var vertElem = document.getElementById( vertexShaderId );
    if ( !vertElem ) {
        alert( "Unable to load vertex shader " + vertexShaderId );
        return -1;
    }
    else {
```

```
20.         vertShdr = gl.createShader( gl.VERTEX_SHADER );
21.         gl.shaderSource( vertShdr, vertElem.text );
22.         gl.compileShader( vertShdr );
23.         if ( !gl.getShaderParameter(vertShdr, gl.COMPILE_STATUS) ) {
24.             var msg = "Vertex shader failed to compile.  The error log is:"
25.             + "<pre>" + gl.getShaderInfoLog( vertShdr ) + "</pre>";
26.             alert( msg );
27.             return -1;
28.         }
29.     }
30.
31.     var fragElem = document.getElementById( fragmentShaderId );
32.     if ( !fragElem ) {
33.         alert( "Unable to load vertex shader " + fragmentShaderId );
34.         return -1;
35.     }
36.     else {
37.         fragShdr = gl.createShader( gl.FRAGMENT_SHADER );
38.         gl.shaderSource( fragShdr, fragElem.text );
39.         gl.compileShader( fragShdr );
40.         if ( !gl.getShaderParameter(fragShdr, gl.COMPILE_STATUS) ) {
41.             var msg = "Fragment shader failed to compile.  The error log is
    :"
42.             + "<pre>" + gl.getShaderInfoLog( fragShdr ) + "</pre>";
43.             alert( msg );
44.             return -1;
45.         }
46.     }
47.
48.     var program = gl.createProgram();
49.     gl.attachShader( program, vertShdr );
50.     gl.attachShader( program, fragShdr );
51.     gl.linkProgram( program );
52.
53.     if ( !gl.getProgramParameter(program, gl.LINK_STATUS) ) {
54.         var msg = "Shader program failed to link.  The error log is:"
55.             + "<pre>" + gl.getProgramInfoLog( program ) + "</pre>";
56.         alert( msg );
57.         return -1;
58.     }
59.
60.     return program;
61. }
```

借鉴

webgl-utils.js

```
/*
 * Copyright 2010, Google Inc.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are
 * met:
 *
 *     * Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *     * Redistributions in binary form must reproduce the above
 * copyright notice, this list of conditions and the following disclaimer
 * in the documentation and/or other materials provided with the
 * distribution.
 *     * Neither the name of Google Inc. nor the names of its
 * contributors may be used to endorse or promote products derived from
 * this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */


/**
 * @fileoverview This file contains functions every webgl program will need
 * a version of one way or another.
 *
 * Instead of setting up a context manually it is recommended to
 * use. This will check for success or failure. On failure it
 * will attempt to present an approriate message to the user.
 *
```

```
41.  *       gl = WebGLUtils.setupWebGL(canvas);
42.  *
43.  * For animated WebGL apps use of setTimeout or setInterval are
44.  * discouraged. It is recommended you structure your rendering
45.  * loop like this.
46.  *
47.  *       function render() {
48.  *         window.requestAnimFrame(render, canvas);
49.  *
50.  *         // do rendering
51.  *         ...
52.  *       }
53.  *       render();
54.  *
55.  * This will call your rendering function up to the refresh rate
56.  * of your display but will stop rendering if your app is not
57.  * visible.
58.  */
59.
60. WebGLUtils = function() {
61.
62. /**
63.  * Creates the HTLM for a failure message
64.  * @param {string} canvasContainerId id of container of th
65.  *        canvas.
66.  * @return {string} The html.
67.  */
68. var makeFailHTML = function(msg) {
69.   return '' +
70.     '<table style="background-
    color: #8CE; width: 100%; height: 100%;"><tr>' +
71.     '<td align="center">' +
72.     '<div style="display: table-cell; vertical-align: middle;">' +
73.     '<div style="">' + msg + '</div>' +
74.     '</div>' +
75.     '</td></tr></table>';
76. };
77.
78. /**
79.  * Mesasge for getting a webgl browser
80.  * @type {string}
81.  */
82. var GET_A_WEBGL_BROWSER = '' +
83.   'This page requires a browser that supports WebGL.<br/>' +
```

```
84.   '<a href="http://get.webgl.org">Click here to upgrade your browser.</a>';
85.
86. /**
87.  * Mesasge for need better hardware
88.  * @type {string}
89.  */
90. var OTHER_PROBLEM = '' +
91.   "It doesn't appear your computer can support WebGL.<br/>" +
92.   '<a href="http://get.webgl.org/troubleshooting/">Click here for more infor
    mation.</a>';
93.
94. /**
95.  * Creates a webgl context. If creation fails it will
96.  * change the contents of the container of the <canvas>
97.  * tag to an error message with the correct links for WebGL.
98.  * @param {Element} canvas. The canvas element to create a
99.  *     context from.
100.  * @param {WebGLContextCreationAttirbutes} opt_attribs Any
101.  *     creation attributes you want to pass in.
102.  * @return {WebGLRenderingContext} The created context.
103.  */
104. var setupWebGL = function(canvas, opt_attribs) {
105.   function showLink(str) {
106.     var container = canvas.parentNode;
107.     if (container) {
108.       container.innerHTML = makeFailHTML(str);
109.     }
110.   };
111.
112.   if (!window.WebGLRenderingContext) {
113.     showLink(GET_A_WEBGL_BROWSER);
114.     return null;
115.   }
116.
117.   var context = create3DContext(canvas, opt_attribs);
118.   if (!context) {
119.     showLink(OTHER_PROBLEM);
120.   }
121.   return context;
122. };
123.
124. /**
125.  * Creates a webgl context.
```

```
126.  * @param {!Canvas} canvas The canvas tag to get context
127.  *     from. If one is not passed in one will be created.
128.  * @return {!WebGLContext} The created context.
129.  */
130. var create3DContext = function(canvas, opt_attribs) {
131.   var names = ["webgl", "experimental-webgl", "webkit-3d", "moz-webgl"];
132.   var context = null;
133.   for (var ii = 0; ii < names.length; ++ii) {
134.     try {
135.       context = canvas.getContext(names[ii], opt_attribs);
136.     } catch(e) {}
137.     if (context) {
138.       break;
139.     }
140.   }
141.   return context;
142. }
143.
144. return {
145.   create3DContext: create3DContext,
146.   setupWebGL: setupWebGL
147. };
148. }();
149.
150. /**
151.  * Provides requestAnimationFrame in a cross browser way.
152.  */
153. window.requestAnimFrame = (function() {
154.   return window.requestAnimationFrame ||
155.          window.webkitRequestAnimationFrame ||
156.          window.mozRequestAnimationFrame ||
157.          window.oRequestAnimationFrame ||
158.          window.msRequestAnimationFrame ||
159.          function(/* function FrameRequestCallback */ callback, /* DOMEleme
    nt Element */ element) {
160.            window.setTimeout(callback, 1000/60);
161.          };
162. })();
```

webgl-debug.js

```
1.  //Copyright (c) 2009 The Chromium Authors. All rights reserved.
2.  //Use of this source code is governed by a BSD-style license that can be
```

```
3.  //found in the LICENSE file.
4.
5.  // Various functions for helping debug WebGL apps.
6.
7.  WebGLDebugUtils = function() {
8.
9.  /**
10.  * Wrapped logging function.
11.  * @param {string} msg Message to log.
12.  */
13. var log = function(msg) {
14.   if (window.console && window.console.log) {
15.     window.console.log(msg);
16.   }
17. };
18.
19. /**
20.  * Which arguements are enums.
21.  * @type {!Object.<number, string>}
22.  */
23. var glValidEnumContexts = {
24.
25.   // Generic setters and getters
26.
27.   'enable': { 0:true },
28.   'disable': { 0:true },
29.   'getParameter': { 0:true },
30.
31.   // Rendering
32.
33.   'drawArrays': { 0:true },
34.   'drawElements': { 0:true, 2:true },
35.
36.   // Shaders
37.
38.   'createShader': { 0:true },
39.   'getShaderParameter': { 1:true },
40.   'getProgramParameter': { 1:true },
41.
42.   // Vertex attributes
43.
44.   'getVertexAttrib': { 1:true },
45.   'vertexAttribPointer': { 2:true },
46.
```

```
47.   // Textures
48.
49.   'bindTexture': { 0:true },
50.   'activeTexture': { 0:true },
51.   'getTexParameter': { 0:true, 1:true },
52.   'texParameterf': { 0:true, 1:true },
53.   'texParameteri': { 0:true, 1:true, 2:true },
54.   'texImage2D': { 0:true, 2:true, 6:true, 7:true },
55.   'texSubImage2D': { 0:true, 6:true, 7:true },
56.   'copyTexImage2D': { 0:true, 2:true },
57.   'copyTexSubImage2D': { 0:true },
58.   'generateMipmap': { 0:true },
59.
60.   // Buffer objects
61.
62.   'bindBuffer': { 0:true },
63.   'bufferData': { 0:true, 2:true },
64.   'bufferSubData': { 0:true },
65.   'getBufferParameter': { 0:true, 1:true },
66.
67.   // Renderbuffers and framebuffers
68.
69.   'pixelStorei': { 0:true, 1:true },
70.   'readPixels': { 4:true, 5:true },
71.   'bindRenderbuffer': { 0:true },
72.   'bindFramebuffer': { 0:true },
73.   'checkFramebufferStatus': { 0:true },
74.   'framebufferRenderbuffer': { 0:true, 1:true, 2:true },
75.   'framebufferTexture2D': { 0:true, 1:true, 2:true },
76.   'getFramebufferAttachmentParameter': { 0:true, 1:true, 2:true },
77.   'getRenderbufferParameter': { 0:true, 1:true },
78.   'renderbufferStorage': { 0:true, 1:true },
79.
80.   // Frame buffer operations (clear, blend, depth test, stencil)
81.
82.   'clear': { 0:true },
83.   'depthFunc': { 0:true },
84.   'blendFunc': { 0:true, 1:true },
85.   'blendFuncSeparate': { 0:true, 1:true, 2:true, 3:true },
86.   'blendEquation': { 0:true },
87.   'blendEquationSeparate': { 0:true, 1:true },
88.   'stencilFunc': { 0:true },
89.   'stencilFuncSeparate': { 0:true, 1:true },
90.   'stencilMaskSeparate': { 0:true },
```

```
  'stencilOp': { 0:true, 1:true, 2:true },
  'stencilOpSeparate': { 0:true, 1:true, 2:true, 3:true },

  // Culling

  'cullFace': { 0:true },
  'frontFace': { 0:true },
};

/**
 * Map of numbers to names.
 * @type {Object}
 */
var glEnums = null;

/**
 * Initializes this module. Safe to call more than once.
 * @param {!WebGLRenderingContext} ctx A WebGL context. If
 *    you have more than one context it doesn't matter which one
 *    you pass in, it is only used to pull out constants.
 */
function init(ctx) {
  if (glEnums == null) {
    glEnums = { };
    for (var propertyName in ctx) {
      if (typeof ctx[propertyName] == 'number') {
        glEnums[ctx[propertyName]] = propertyName;
      }
    }
  }
}

/**
 * Checks the utils have been initialized.
 */
function checkInit() {
  if (glEnums == null) {
    throw 'WebGLDebugUtils.init(ctx) not called';
  }
}

/**
 * Returns true or false if value matches any WebGL enum
 * @param {*} value Value to check if it might be an enum.
```

```
135.  * @return {boolean} True if value matches one of the WebGL defined enums
136.  */
137. function mightBeEnum(value) {
138.   checkInit();
139.   return (glEnums[value] !== undefined);
140. }
141.
142. /**
143.  * Gets an string version of an WebGL enum.
144.  *
145.  * Example:
146.  *   var str = WebGLDebugUtil.glEnumToString(ctx.getError());
147.  *
148.  * @param {number} value Value to return an enum for
149.  * @return {string} The string version of the enum.
150.  */
151. function glEnumToString(value) {
152.   checkInit();
153.   var name = glEnums[value];
154.   return (name !== undefined) ? name :
155.       ("*UNKNOWN WebGL ENUM (0x" + value.toString(16) + ")");
156. }
157.
158. /**
159.  * Returns the string version of a WebGL argument.
160.  * Attempts to convert enum arguments to strings.
161.  * @param {string} functionName the name of the WebGL function.
162.  * @param {number} argumentIndx the index of the argument.
163.  * @param {*} value The value of the argument.
164.  * @return {string} The value as a string.
165.  */
166. function glFunctionArgToString(functionName, argumentIndex, value) {
167.   var funcInfo = glValidEnumContexts[functionName];
168.   if (funcInfo !== undefined) {
169.     if (funcInfo[argumentIndex]) {
170.       return glEnumToString(value);
171.     }
172.   }
173.   return value.toString();
174. }
175.
176. /**
177.  * Given a WebGL context returns a wrapped context that calls
178.  * gl.getError after every command and calls a function if the
```

```
179.  * result is not gl.NO_ERROR.
180.  *
181.  * @param {!WebGLRenderingContext} ctx The webgl context to
182.  *        wrap.
183.  * @param {!function(err, funcName, args): void} opt_onErrorFunc
184.  *        The function to call when gl.getError returns an
185.  *        error. If not specified the default function calls
186.  *        console.log with a message.
187.  */
188. function makeDebugContext(ctx, opt_onErrorFunc) {
189.   init(ctx);
190.   opt_onErrorFunc = opt_onErrorFunc || function(err, functionName, args) {

191.         // apparently we can't do args.join(",");
192.         var argStr = "";
193.         for (var ii = 0; ii < args.length; ++ii) {
194.           argStr += ((ii == 0) ? '' : ', ') +
195.               glFunctionArgToString(functionName, ii, args[ii]);
196.         }
197.         log("WebGL error "+ glEnumToString(err) + " in "+ functionName +
198.             "(" + argStr + ")");
199.       };
200.
201.   // Holds booleans for each GL error so after we get the error ourselves
202.   // we can still return it to the client app.
203.   var glErrorShadow = { };
204.
205.   // Makes a function that calls a WebGL function and then calls getError.

206.   function makeErrorWrapper(ctx, functionName) {
207.     return function() {
208.       var result = ctx[functionName].apply(ctx, arguments);
209.       var err = ctx.getError();
210.       if (err != 0) {
211.         glErrorShadow[err] = true;
212.         opt_onErrorFunc(err, functionName, arguments);
213.       }
214.       return result;
215.     };
216.   }
217.
218.   // Make a an object that has a copy of every property of the WebGL contex
    t
219.   // but wraps all functions.
```

```
220.   var wrapper = {};
221.   for (var propertyName in ctx) {
222.     if (typeof ctx[propertyName] == 'function') {
223.         wrapper[propertyName] = makeErrorWrapper(ctx, propertyName);
224.       } else {
225.         wrapper[propertyName] = ctx[propertyName];
226.       }
227.   }
228.
229.   // Override the getError function with one that returns our saved results
    .
230.   wrapper.getError = function() {
231.     for (var err in glErrorShadow) {
232.       if (glErrorShadow[err]) {
233.         glErrorShadow[err] = false;
234.         return err;
235.       }
236.     }
237.     return ctx.NO_ERROR;
238.   };
239.
240.   return wrapper;
241. }
242.
243. function resetToInitialState(ctx) {
244.   var numAttribs = ctx.getParameter(ctx.MAX_VERTEX_ATTRIBS);
245.   var tmp = ctx.createBuffer();
246.   ctx.bindBuffer(ctx.ARRAY_BUFFER, tmp);
247.   for (var ii = 0; ii < numAttribs; ++ii) {
248.     ctx.disableVertexAttribArray(ii);
249.     ctx.vertexAttribPointer(ii, 4, ctx.FLOAT, false, 0, 0);
250.     ctx.vertexAttrib1f(ii, 0);
251.   }
252.   ctx.deleteBuffer(tmp);
253.
254.   var numTextureUnits = ctx.getParameter(ctx.MAX_TEXTURE_IMAGE_UNITS);
255.   for (var ii = 0; ii < numTextureUnits; ++ii) {
256.     ctx.activeTexture(ctx.TEXTURE0 + ii);
257.     ctx.bindTexture(ctx.TEXTURE_CUBE_MAP, null);
258.     ctx.bindTexture(ctx.TEXTURE_2D, null);
259.   }
260.
261.   ctx.activeTexture(ctx.TEXTURE0);
262.   ctx.useProgram(null);
```

```
263.   ctx.bindBuffer(ctx.ARRAY_BUFFER, null);
264.   ctx.bindBuffer(ctx.ELEMENT_ARRAY_BUFFER, null);
265.   ctx.bindFramebuffer(ctx.FRAMEBUFFER, null);
266.   ctx.bindRenderbuffer(ctx.RENDERBUFFER, null);
267.   ctx.disable(ctx.BLEND);
268.   ctx.disable(ctx.CULL_FACE);
269.   ctx.disable(ctx.DEPTH_TEST);
270.   ctx.disable(ctx.DITHER);
271.   ctx.disable(ctx.SCISSOR_TEST);
272.   ctx.blendColor(0, 0, 0, 0);
273.   ctx.blendEquation(ctx.FUNC_ADD);
274.   ctx.blendFunc(ctx.ONE, ctx.ZERO);
275.   ctx.clearColor(0, 0, 0, 0);
276.   ctx.clearDepth(1);
277.   ctx.clearStencil(-1);
278.   ctx.colorMask(true, true, true, true);
279.   ctx.cullFace(ctx.BACK);
280.   ctx.depthFunc(ctx.LESS);
281.   ctx.depthMask(true);
282.   ctx.depthRange(0, 1);
283.   ctx.frontFace(ctx.CCW);
284.   ctx.hint(ctx.GENERATE_MIPMAP_HINT, ctx.DONT_CARE);
285.   ctx.lineWidth(1);
286.   ctx.pixelStorei(ctx.PACK_ALIGNMENT, 4);
287.   ctx.pixelStorei(ctx.UNPACK_ALIGNMENT, 4);
288.   ctx.pixelStorei(ctx.UNPACK_FLIP_Y_WEBGL, false);
289.   ctx.pixelStorei(ctx.UNPACK_PREMULTIPLY_ALPHA_WEBGL, false);
290.   // TODO: Delete this IF.
291.   if (ctx.UNPACK_COLORSPACE_CONVERSION_WEBGL) {
292.     ctx.pixelStorei(ctx.UNPACK_COLORSPACE_CONVERSION_WEBGL, ctx.BROWSER_DEF
    AULT_WEBGL);
293.   }
294.   ctx.polygonOffset(0, 0);
295.   ctx.sampleCoverage(1, false);
296.   ctx.scissor(0, 0, ctx.canvas.width, ctx.canvas.height);
297.   ctx.stencilFunc(ctx.ALWAYS, 0, 0xFFFFFFFF);
298.   ctx.stencilMask(0xFFFFFFFF);
299.   ctx.stencilOp(ctx.KEEP, ctx.KEEP, ctx.KEEP);
300.   ctx.viewport(0, 0, ctx.canvas.clientWidth, ctx.canvas.clientHeight);
301.   ctx.clear(ctx.COLOR_BUFFER_BIT | ctx.DEPTH_BUFFER_BIT | ctx.STENCIL_BUFFE
    R_BIT);
302.
303.   // TODO: This should NOT be needed but Firefox fails with 'hint'
304.   while(ctx.getError());
```

```
}

function makeLostContextSimulatingContext(ctx) {
  var wrapper_ = {};
  var contextId_ = 1;
  var contextLost_ = false;
  var resourceId_ = 0;
  var resourceDb_ = [];
  var onLost_ = undefined;
  var onRestored_ = undefined;
  var nextOnRestored_ = undefined;

  // Holds booleans for each GL error so can simulate errors.
  var glErrorShadow_ = { };

  function isWebGLObject(obj) {
    //return false;
    return (obj instanceof WebGLBuffer ||
            obj instanceof WebGLFramebuffer ||
            obj instanceof WebGLProgram ||
            obj instanceof WebGLRenderbuffer ||
            obj instanceof WebGLShader ||
            obj instanceof WebGLTexture);
  }

  function checkResources(args) {
    for (var ii = 0; ii < args.length; ++ii) {
      var arg = args[ii];
      if (isWebGLObject(arg)) {
        return arg.__webglDebugContextLostId__ == contextId_;
      }
    }
    return true;
  }

  function clearErrors() {
    var k = Object.keys(glErrorShadow_);
    for (var ii = 0; ii < k.length; ++ii) {
      delete glErrorShdow_[k];
    }
  }

  // Makes a function that simulates WebGL when out of context.
  function makeLostContextWrapper(ctx, functionName) {
```

```
    var f = ctx[functionName];
    return function() {
      // Only call the functions if the context is not lost.
      if (!contextLost_) {
        if (!checkResources(arguments)) {
          glErrorShadow_[ctx.INVALID_OPERATION] = true;
          return;
        }
        var result = f.apply(ctx, arguments);
        return result;
      }
    };
  }

  for (var propertyName in ctx) {
    if (typeof ctx[propertyName] == 'function') {
       wrapper_[propertyName] = makeLostContextWrapper(ctx, propertyName);

     } else {
       wrapper_[propertyName] = ctx[propertyName];
     }
  }

  function makeWebGLContextEvent(statusMessage) {
    return {statusMessage: statusMessage};
  }

  function freeResources() {
    for (var ii = 0; ii < resourceDb_.length; ++ii) {
      var resource = resourceDb_[ii];
      if (resource instanceof WebGLBuffer) {
        ctx.deleteBuffer(resource);
      } else if (resource instanceof WebctxFramebuffer) {
        ctx.deleteFramebuffer(resource);
      } else if (resource instanceof WebctxProgram) {
        ctx.deleteProgram(resource);
      } else if (resource instanceof WebctxRenderbuffer) {
        ctx.deleteRenderbuffer(resource);
      } else if (resource instanceof WebctxShader) {
        ctx.deleteShader(resource);
      } else if (resource instanceof WebctxTexture) {
        ctx.deleteTexture(resource);
      }
    }
```

```
392.   }
393.
394.   wrapper_.loseContext = function() {
395.     if (!contextLost_) {
396.       contextLost_ = true;
397.       ++contextId_;
398.       while (ctx.getError());
399.       clearErrors();
400.       glErrorShadow_[ctx.CONTEXT_LOST_WEBGL] = true;
401.       setTimeout(function() {
402.           if (onLost_) {
403.             onLost_(makeWebGLContextEvent("context lost"));
404.           }
405.         }, 0);
406.     }
407.   };
408.
409.   wrapper_.restoreContext = function() {
410.     if (contextLost_) {
411.       if (onRestored_) {
412.         setTimeout(function() {
413.             freeResources();
414.             resetToInitialState(ctx);
415.             contextLost_ = false;
416.             if (onRestored_) {
417.               var callback = onRestored_;
418.               onRestored_ = nextOnRestored_;
419.               nextOnRestored_ = undefined;
420.               callback(makeWebGLContextEvent("context restored"));
421.             }
422.           }, 0);
423.       } else {
424.         throw "You can not restore the context without a listener"
425.       }
426.     }
427.   };
428.
429.   // Wrap a few functions specially.
430.   wrapper_.getError = function() {
431.     if (!contextLost_) {
432.       var err;
433.       while (err = ctx.getError()) {
434.         glErrorShadow_[err] = true;
435.       }
```

```
436.     }
437.     for (var err in glErrorShadow_) {
438.       if (glErrorShadow_[err]) {
439.         delete glErrorShadow_[err];
440.         return err;
441.       }
442.     }
443.     return ctx.NO_ERROR;
444.   };
445.
446.   var creationFunctions = [
447.     "createBuffer",
448.     "createFramebuffer",
449.     "createProgram",
450.     "createRenderbuffer",
451.     "createShader",
452.     "createTexture"
453.   ];
454.   for (var ii = 0; ii < creationFunctions.length; ++ii) {
455.     var functionName = creationFunctions[ii];
456.     wrapper_[functionName] = function(f) {
457.       return function() {
458.         if (contextLost_) {
459.           return null;
460.         }
461.         var obj = f.apply(ctx, arguments);
462.         obj.__webglDebugContextLostId__ = contextId_;
463.         resourceDb_.push(obj);
464.         return obj;
465.       };
466.     }(ctx[functionName]);
467.   }
468.
469.   var functionsThatShouldReturnNull = [
470.     "getActiveAttrib",
471.     "getActiveUniform",
472.     "getBufferParameter",
473.     "getContextAttributes",
474.     "getAttachedShaders",
475.     "getFramebufferAttachmentParameter",
476.     "getParameter",
477.     "getProgramParameter",
478.     "getProgramInfoLog",
479.     "getRenderbufferParameter",
```

```
    "getShaderParameter",
    "getShaderInfoLog",
    "getShaderSource",
    "getTexParameter",
    "getUniform",
    "getUniformLocation",
    "getVertexAttrib"
  ];
  for (var ii = 0; ii < functionsThatShouldReturnNull.length; ++ii) {
    var functionName = functionsThatShouldReturnNull[ii];
    wrapper_[functionName] = function(f) {
      return function() {
        if (contextLost_) {
          return null;
        }
        return f.apply(ctx, arguments);
      }
    }(wrapper_[functionName]);
  }

  var isFunctions = [
    "isBuffer",
    "isEnabled",
    "isFramebuffer",
    "isProgram",
    "isRenderbuffer",
    "isShader",
    "isTexture"
  ];
  for (var ii = 0; ii < isFunctions.length; ++ii) {
    var functionName = isFunctions[ii];
    wrapper_[functionName] = function(f) {
      return function() {
        if (contextLost_) {
          return false;
        }
        return f.apply(ctx, arguments);
      }
    }(wrapper_[functionName]);
  }

  wrapper_.checkFramebufferStatus = function(f) {
    return function() {
      if (contextLost_) {
```

```
524.         return ctx.FRAMEBUFFER_UNSUPPORTED;
525.       }
526.       return f.apply(ctx, arguments);
527.     };
528.   }(wrapper_.checkFramebufferStatus);
529.
530.   wrapper_.getAttribLocation = function(f) {
531.     return function() {
532.       if (contextLost_) {
533.         return -1;
534.       }
535.       return f.apply(ctx, arguments);
536.     };
537.   }(wrapper_.getAttribLocation);
538.
539.   wrapper_.getVertexAttribOffset = function(f) {
540.     return function() {
541.       if (contextLost_) {
542.         return 0;
543.       }
544.       return f.apply(ctx, arguments);
545.     };
546.   }(wrapper_.getVertexAttribOffset);
547.
548.   wrapper_.isContextLost = function() {
549.     return contextLost_;
550.   };
551.
552.   function wrapEvent(listener) {
553.     if (typeof(listener) == "function") {
554.       return listener;
555.     } else {
556.       return function(info) {
557.         listener.handleEvent(info);
558.       }
559.     }
560.   }
561.
562.   wrapper_.registerOnContextLostListener = function(listener) {
563.     onLost_ = wrapEvent(listener);
564.   };
565.
566.   wrapper_.registerOnContextRestoredListener = function(listener) {
567.     if (contextLost_) {
```

```
568.       nextOnRestored_ = wrapEvent(listener);
569.     } else {
570.       onRestored_ = wrapEvent(listener);
571.     }
572.   }
573.
574.   return wrapper_;
575. }
576.
577. return {
578.   /**
579.    * Initializes this module. Safe to call more than once.
580.    * @param {!WebGLRenderingContext} ctx A WebGL context. If
581.    *    you have more than one context it doesn't matter which one
582.    *    you pass in, it is only used to pull out constants.
583.    */
584.   'init': init,
585.
586.   /**
587.    * Returns true or false if value matches any WebGL enum
588.    * @param {*} value Value to check if it might be an enum.
589.    * @return {boolean} True if value matches one of the WebGL defined enums

590.    */
591.   'mightBeEnum': mightBeEnum,
592.
593.   /**
594.    * Gets an string version of an WebGL enum.
595.    *
596.    * Example:
597.    *   WebGLDebugUtil.init(ctx);
598.    *   var str = WebGLDebugUtil.glEnumToString(ctx.getError());
599.    *
600.    * @param {number} value Value to return an enum for
601.    * @return {string} The string version of the enum.
602.    */
603.   'glEnumToString': glEnumToString,
604.
605.   /**
606.    * Converts the argument of a WebGL function to a string.
607.    * Attempts to convert enum arguments to strings.
608.    *
609.    * Example:
610.    *   WebGLDebugUtil.init(ctx);
```

```
611.     *   var str = WebGLDebugUtil.glFunctionArgToString('bindTexture', 0, gl.
    TEXTURE_2D);
612.     *
613.     * would return 'TEXTURE_2D'
614.     *
615.     * @param {string} functionName the name of the WebGL function.
616.     * @param {number} argumentIndx the index of the argument.
617.     * @param {*} value The value of the argument.
618.     * @return {string} The value as a string.
619.     */
620.    'glFunctionArgToString': glFunctionArgToString,
621.
622.    /**
623.     * Given a WebGL context returns a wrapped context that calls
624.     * gl.getError after every command and calls a function if the
625.     * result is not NO_ERROR.
626.     *
627.     * You can supply your own function if you want. For example, if you'd li
    ke
628.     * an exception thrown on any GL error you could do this
629.     *
630.     *    function throwOnGLError(err, funcName, args) {
631.     *      throw WebGLDebugUtils.glEnumToString(err) + " was caused by call
    to" +
632.     *            funcName;
633.     *    };
634.     *
635.     *    ctx = WebGLDebugUtils.makeDebugContext(
636.     *        canvas.getContext("webgl"), throwOnGLError);
637.     *
638.     * @param {!WebGLRenderingContext} ctx The webgl context to wrap.
639.     * @param {!function(err, funcName, args): void} opt_onErrorFunc The func
    tion
640.     *     to call when gl.getError returns an error. If not specified the de
    fault
641.     *     function calls console.log with a message.
642.     */
643.    'makeDebugContext': makeDebugContext,
644.
645.    /**
646.     * Given a WebGL context returns a wrapped context that adds 4
647.     * functions.
648.     *
649.     * ctx.loseContext:
```

```
650.     *   simulates a lost context event.
651.     *
652.     * ctx.restoreContext:
653.     *   simulates the context being restored.
654.     *
655.     * ctx.registerOnContextLostListener(listener):
656.     *   lets you register a listener for context lost. Use instead
657.     *   of addEventListener('webglcontextlostevent', listener);
658.     *
659.     * ctx.registerOnContextRestoredListener(listener):
660.     *   lets you register a listener for context restored. Use
661.     *   instead of addEventListener('webglcontextrestored',
662.     *   listener);
663.     *
664.     * @param {!WebGLRenderingContext} ctx The webgl context to wrap.
665.     */
666.    'makeLostContextSimulatingContext': makeLostContextSimulatingContext,
667.
668.    /**
669.     * Resets a context to the initial state.
670.     * @param {!WebGLRenderingContext} ctx The webgl context to
671.     *     reset.
672.     */
673.    'resetToInitialState': resetToInitialState
674. };
675.
676. }();
```

cuon-utils.js

```
1.  // cuon-utils.js (c) 2012 kanda and matsuda
2.  /**
3.   * Create a program object and make current
4.   * @param gl GL context
5.   * @param vshader a vertex shader program (string)
6.   * @param fshader a fragment shader program (string)
7.   * @return true, if the program object was created and successfully made cur
    rent
8.   */
9.  function initShaders(gl, vshader, fshader) {
10.   var program = createProgram(gl, vshader, fshader);
11.   if (!program) {
12.     console.log('Failed to create program');
```

```
    return false;
  }

  gl.useProgram(program);
  gl.program = program;

  return true;
}

/**
 * Create the linked program object
 * @param gl GL context
 * @param vshader a vertex shader program (string)
 * @param fshader a fragment shader program (string)
 * @return created program object, or null if the creation has failed
 */
function createProgram(gl, vshader, fshader) {
  // Create shader object
  var vertexShader = loadShader(gl, gl.VERTEX_SHADER, vshader);
  var fragmentShader = loadShader(gl, gl.FRAGMENT_SHADER, fshader);
  if (!vertexShader || !fragmentShader) {
    return null;
  }

  // Create a program object
  var program = gl.createProgram();
  if (!program) {
    return null;
  }

  // Attach the shader objects
  gl.attachShader(program, vertexShader);
  gl.attachShader(program, fragmentShader);

  // Link the program object
  gl.linkProgram(program);

  // Check the result of linking
  var linked = gl.getProgramParameter(program, gl.LINK_STATUS);
  if (!linked) {
    var error = gl.getProgramInfoLog(program);
    console.log('Failed to link program: ' + error);
    gl.deleteProgram(program);
    gl.deleteShader(fragmentShader);
```

```
57.     gl.deleteShader(vertexShader);
58.     return null;
59.   }
60.   return program;
61. }
62.
63. /**
64.  * Create a shader object
65.  * @param gl GL context
66.  * @param type the type of the shader object to be created
67.  * @param source shader program (string)
68.  * @return created shader object, or null if the creation has failed.
69.  */
70. function loadShader(gl, type, source) {
71.   // Create shader object
72.   var shader = gl.createShader(type);
73.   if (shader == null) {
74.     console.log('unable to create shader');
75.     return null;
76.   }
77.
78.   // Set the shader program
79.   gl.shaderSource(shader, source);
80.
81.   // Compile the shader
82.   gl.compileShader(shader);
83.
84.   // Check the result of compilation
85.   var compiled = gl.getShaderParameter(shader, gl.COMPILE_STATUS);
86.   if (!compiled) {
87.     var error = gl.getShaderInfoLog(shader);
88.     console.log('Failed to compile shader: ' + error);
89.     gl.deleteShader(shader);
90.     return null;
91.   }
92.
93.   return shader;
94. }
95.
96. /**
97.  * Initialize and get the rendering for WebGL
98.  * @param canvas <cavnas> element
99.  * @param opt_debug flag to initialize the context for debugging
100.  * @return the rendering context for WebGL
```

```
 */
function getWebGLContext(canvas, opt_debug) {
  // Get the rendering context for WebGL
  var gl = WebGLUtils.setupWebGL(canvas);
  if (!gl) return null;

  // if opt_debug is explicitly false, create the context for debugging
  if (arguments.length < 2 || opt_debug) {
    gl = WebGLDebugUtils.makeDebugContext(gl);
  }

  return gl;
}
```

Controls.js

```
var gl, canvas,  pi180 = 180/Math.PI,
  transl = -3, rTouch, fiTouch, idTouch0,
  xRot = yRot = zRot =  xOffs = yOffs =  drag = 0;
function startTouch(evt) {
  var evList = evt.touches;
  if(evList.length == 1){
    xOffs = evList[0].pageX;  yOffs = evList[0].pageY;
    drag = 1;}
  else if(evList.length == 2){
    idTouch0 = evList[0].identifier;
    var dx = evList[1].pageX - evList[0].pageX;
    var dy = evList[1].pageY - evList[0].pageY;
    rTouch = Math.sqrt(dx*dx + dy*dy);
    fiTouch = Math.atan2(dy, dx);
    drag = 2;}
  evt.preventDefault();
}
function continueTouch(evt) {
  if(drag == 1){
    var x = evt.touches[0].pageX,  y = evt.touches[0].pageY;
    yRot = x - xOffs;  xRot = y - yOffs;
    xOffs = x;  yOffs = y;
    drawScene();}
  else if(drag == 2){
    var dx = evt.touches[1].pageX - evt.touches[0].pageX;
    var dy = evt.touches[1].pageY - evt.touches[0].pageY;
    var r = Math.sqrt(dx*dx + dy*dy);
```

```
    var fi;
    if( idTouch0 == evt.touches[0].identifier ) fi = Math.atan2(dy, dx);
    else fi = Math.atan2(-dy, -dx);
    transl *= rTouch / r;
    zRot = pi180*(fiTouch - fi);
    rTouch = r;  fiTouch = fi;
    drawScene();
  }
}
function stopTouch() {
  drag = 0;
}
function mymousedown( ev ){
  drag  = 1;
  xOffs = ev.clientX;  yOffs = ev.clientY;
}
function mymouseup( ev ){
  drag  = 0;
}
function mymousemove( ev ){
  if ( drag == 0 ) return;
  if ( ev.shiftKey ) {
    transl *= 1 + (ev.clientY - yOffs)/1000;
    zRot = (xOffs - ev.clientX)*.3; }
  else {
    yRot = - xOffs + ev.clientX;  xRot = - yOffs + ev.clientY; }
  xOffs = ev.clientX;   yOffs = ev.clientY;
  drawScene();
}
function wheelHandler(ev) {
  var del = 1.1;
  if (ev.shiftKey) del = 1.01;
  var ds = ((ev.detail || ev.wheelDelta) > 0) ? del : (1 / del);
  transl *= ds;
  ev.preventDefault();
  drawScene();
}
function getShader ( gl, id ){
   var shaderScript = document.getElementById ( id );
   var str = "";
   var k = shaderScript.firstChild;
   while ( k ){
     if ( k.nodeType == 3 ) str += k.textContent;
     k = k.nextSibling;
```

```
72.    }
73.    var shader;
74.    if ( shaderScript.type == "x-shader/x-fragment" )
75.            shader = gl.createShader ( gl.FRAGMENT_SHADER );
76.    else if ( shaderScript.type == "x-shader/x-vertex" )
77.            shader = gl.createShader(gl.VERTEX_SHADER);
78.    else return null;
79.    gl.shaderSource(shader, str);
80.    gl.compileShader(shader);
81.    if (gl.getShaderParameter(shader, gl.COMPILE_STATUS) == 0)
82.       alert(id + "\n" + gl.getShaderInfoLog(shader));
83.    return shader;
84. }
85. function initGL(){
86.    canvas = document.getElementById("canvas");
87.    if (!window.WebGLRenderingContext){
88.      alert("Your browser does not support WebGL. See http://get.webgl.org");
89.      return;}
90.    try { gl = canvas.getContext("experimental-webgl");
91.    } catch(e) {}
92.    if ( !gl ) {alert("Can't get WebGL"); return;}
93.    canvas.addEventListener('DOMMouseScroll', wheelHandler, false);
94.    canvas.addEventListener('mousewheel', wheelHandler, false);
95.    canvas.addEventListener('mousedown', mymousedown, false);
96.    canvas.addEventListener('mouseup', mymouseup, false);
97.    canvas.addEventListener('mousemove', mymousemove, false);
98.    canvas.addEventListener('touchstart', startTouch, false);
99.    canvas.addEventListener('touchmove', continueTouch, false);
100.    canvas.addEventListener('touchend', stopTouch, false);
101. }
102. function initGL2(){
103.    canvas = document.getElementById("canvas");
104.    if (!window.WebGLRenderingContext){
105.      alert("Your browser does not support WebGL. See http://get.webgl.org")
   ;
106.      return;}
107.    try { gl = canvas.getContext("webgl2");
108.    } catch(e) {}
109.    if ( !gl ) {alert("Can't get WebGL"); return;}
110.    canvas.addEventListener('DOMMouseScroll', wheelHandler, false);
111.    canvas.addEventListener('mousewheel', wheelHandler, false);
112.    canvas.addEventListener('mousedown', mymousedown, false);
113.    canvas.addEventListener('mouseup', mymouseup, false);
```

```
    canvas.addEventListener('mousemove', mymousemove, false);
    canvas.addEventListener('touchstart', startTouch, false);
    canvas.addEventListener('touchmove', continueTouch, false);
    canvas.addEventListener('touchend', stopTouch, false);
}
```

CanvasMatrix.js

```
CanvasMatrix4 = function(m) {
    if (typeof m == 'object') {
        if ("length" in m && m.length >= 16) {
            this.load(m[0], m[1], m[2], m[3], m[4], m[5], m[6], m[7], m[8], m[9], m[10], m[11], m[12], m[13], m[14], m[15]);
            return
        } else if (m instanceof CanvasMatrix4) {
            this.load(m);
            return
        }
    }
    this.makeIdentity()
};
CanvasMatrix4.prototype.load = function() {
    if (arguments.length == 1 && typeof arguments[0] == 'object') {
        var matrix = arguments[0];
        if ("length" in matrix && matrix.length == 16) {
            this.m11 = matrix[0];
            this.m12 = matrix[1];
            this.m13 = matrix[2];
            this.m14 = matrix[3];
            this.m21 = matrix[4];
            this.m22 = matrix[5];
            this.m23 = matrix[6];
            this.m24 = matrix[7];
            this.m31 = matrix[8];
            this.m32 = matrix[9];
            this.m33 = matrix[10];
            this.m34 = matrix[11];
            this.m41 = matrix[12];
            this.m42 = matrix[13];
            this.m43 = matrix[14];
            this.m44 = matrix[15];
            return
        }
```

```
35.         if (arguments[0] instanceof CanvasMatrix4) {
36.             this.m11 = matrix.m11;
37.             this.m12 = matrix.m12;
38.             this.m13 = matrix.m13;
39.             this.m14 = matrix.m14;
40.             this.m21 = matrix.m21;
41.             this.m22 = matrix.m22;
42.             this.m23 = matrix.m23;
43.             this.m24 = matrix.m24;
44.             this.m31 = matrix.m31;
45.             this.m32 = matrix.m32;
46.             this.m33 = matrix.m33;
47.             this.m34 = matrix.m34;
48.             this.m41 = matrix.m41;
49.             this.m42 = matrix.m42;
50.             this.m43 = matrix.m43;
51.             this.m44 = matrix.m44;
52.             return
53.         }
54.     }
55.     this.makeIdentity()
56. };
57. CanvasMatrix4.prototype.getAsArray = function() {
58.     return [this.m11, this.m12, this.m13, this.m14, this.m21, this.m22, this
    .m23, this.m24, this.m31, this.m32, this.m33, this.m34, this.m41, this.m42,
    this.m43, this.m44]
59. };
60. CanvasMatrix4.prototype.getAsWebGLFloatArray = function() {
61.     return new WebGLFloatArray(this.getAsArray())
62. };
63. CanvasMatrix4.prototype.makeIdentity = function() {
64.     this.m11 = 1;
65.     this.m12 = 0;
66.     this.m13 = 0;
67.     this.m14 = 0;
68.     this.m21 = 0;
69.     this.m22 = 1;
70.     this.m23 = 0;
71.     this.m24 = 0;
72.     this.m31 = 0;
73.     this.m32 = 0;
74.     this.m33 = 1;
75.     this.m34 = 0;
76.     this.m41 = 0;
```

```
77.     this.m42 = 0;
78.     this.m43 = 0;
79.     this.m44 = 1
80. };
81. CanvasMatrix4.prototype.transpose = function() {
82.     var tmp = this.m12;
83.     this.m12 = this.m21;
84.     this.m21 = tmp;
85.     tmp = this.m13;
86.     this.m13 = this.m31;
87.     this.m31 = tmp;
88.     tmp = this.m14;
89.     this.m14 = this.m41;
90.     this.m41 = tmp;
91.     tmp = this.m23;
92.     this.m23 = this.m32;
93.     this.m32 = tmp;
94.     tmp = this.m24;
95.     this.m24 = this.m42;
96.     this.m42 = tmp;
97.     tmp = this.m34;
98.     this.m34 = this.m43;
99.     this.m43 = tmp
100. };
101. CanvasMatrix4.prototype.invert = function() {
102.     var det = this._determinant4x4();
103.     if (Math.abs(det) < 1e-8) return null;
104.     this._makeAdjoint();
105.     this.m11 /= det;
106.     this.m12 /= det;
107.     this.m13 /= det;
108.     this.m14 /= det;
109.     this.m21 /= det;
110.     this.m22 /= det;
111.     this.m23 /= det;
112.     this.m24 /= det;
113.     this.m31 /= det;
114.     this.m32 /= det;
115.     this.m33 /= det;
116.     this.m34 /= det;
117.     this.m41 /= det;
118.     this.m42 /= det;
119.     this.m43 /= det;
120.     this.m44 /= det
```

```
121. };
122. CanvasMatrix4.prototype.translate = function(x, y, z) {
123.     if (x == undefined) x = 0;
124.     if (y == undefined) y = 0;
125.     if (z == undefined) z = 0;
126.     var matrix = new CanvasMatrix4();
127.     matrix.m41 = x;
128.     matrix.m42 = y;
129.     matrix.m43 = z;
130.     this.multRight(matrix)
131. };
132. CanvasMatrix4.prototype.scale = function(x, y, z) {
133.     if (x == undefined) x = 1;
134.     if (z == undefined) {
135.         if (y == undefined) {
136.             y = x;
137.             z = x
138.         } else z = 1
139.     } else if (y == undefined) y = x;
140.     var matrix = new CanvasMatrix4();
141.     matrix.m11 = x;
142.     matrix.m22 = y;
143.     matrix.m33 = z;
144.     this.multRight(matrix)
145. };
146. CanvasMatrix4.prototype.rotate = function(angle, x, y, z) {
147.     angle = angle / 180 * Math.PI;
148.     angle /= 2;
149.     var sinA = Math.sin(angle);
150.     var cosA = Math.cos(angle);
151.     var sinA2 = sinA * sinA;
152.     var length = Math.sqrt(x * x + y * y + z * z);
153.     if (length == 0) {
154.         x = 0;
155.         y = 0;
156.         z = 1
157.     } else if (length != 1) {
158.         x /= length;
159.         y /= length;
160.         z /= length
161.     }
162.     var mat = new CanvasMatrix4();
163.     if (x == 1 && y == 0 && z == 0) {
164.         mat.m11 = 1;
```

```
165.         mat.m12 = 0;
166.         mat.m13 = 0;
167.         mat.m21 = 0;
168.         mat.m22 = 1 - 2 * sinA2;
169.         mat.m23 = 2 * sinA * cosA;
170.         mat.m31 = 0;
171.         mat.m32 = -2 * sinA * cosA;
172.         mat.m33 = 1 - 2 * sinA2;
173.         mat.m14 = mat.m24 = mat.m34 = 0;
174.         mat.m41 = mat.m42 = mat.m43 = 0;
175.         mat.m44 = 1
176.     } else if (x == 0 && y == 1 && z == 0) {
177.         mat.m11 = 1 - 2 * sinA2;
178.         mat.m12 = 0;
179.         mat.m13 = -2 * sinA * cosA;
180.         mat.m21 = 0;
181.         mat.m22 = 1;
182.         mat.m23 = 0;
183.         mat.m31 = 2 * sinA * cosA;
184.         mat.m32 = 0;
185.         mat.m33 = 1 - 2 * sinA2;
186.         mat.m14 = mat.m24 = mat.m34 = 0;
187.         mat.m41 = mat.m42 = mat.m43 = 0;
188.         mat.m44 = 1
189.     } else if (x == 0 && y == 0 && z == 1) {
190.         mat.m11 = 1 - 2 * sinA2;
191.         mat.m12 = 2 * sinA * cosA;
192.         mat.m13 = 0;
193.         mat.m21 = -2 * sinA * cosA;
194.         mat.m22 = 1 - 2 * sinA2;
195.         mat.m23 = 0;
196.         mat.m31 = 0;
197.         mat.m32 = 0;
198.         mat.m33 = 1;
199.         mat.m14 = mat.m24 = mat.m34 = 0;
200.         mat.m41 = mat.m42 = mat.m43 = 0;
201.         mat.m44 = 1
202.     } else {
203.         var x2 = x * x;
204.         var y2 = y * y;
205.         var z2 = z * z;
206.         mat.m11 = 1 - 2 * (y2 + z2) * sinA2;
207.         mat.m12 = 2 * (x * y * sinA2 + z * sinA * cosA);
208.         mat.m13 = 2 * (x * z * sinA2 - y * sinA * cosA);
```

```
209.         mat.m21 = 2 * (y * x * sinA2 - z * sinA * cosA);
210.         mat.m22 = 1 - 2 * (z2 + x2) * sinA2;
211.         mat.m23 = 2 * (y * z * sinA2 + x * sinA * cosA);
212.         mat.m31 = 2 * (z * x * sinA2 + y * sinA * cosA);
213.         mat.m32 = 2 * (z * y * sinA2 - x * sinA * cosA);
214.         mat.m33 = 1 - 2 * (x2 + y2) * sinA2;
215.         mat.m14 = mat.m24 = mat.m34 = 0;
216.         mat.m41 = mat.m42 = mat.m43 = 0;
217.         mat.m44 = 1
218.     }
219.     this.multRight(mat)
220. };
221. CanvasMatrix4.prototype.multRight = function(mat) {
222.     var m11 = (this.m11 * mat.m11 + this.m12 * mat.m21 + this.m13 * mat.m31
     + this.m14 * mat.m41);
223.     var m12 = (this.m11 * mat.m12 + this.m12 * mat.m22 + this.m13 * mat.m32
     + this.m14 * mat.m42);
224.     var m13 = (this.m11 * mat.m13 + this.m12 * mat.m23 + this.m13 * mat.m33
     + this.m14 * mat.m43);
225.     var m14 = (this.m11 * mat.m14 + this.m12 * mat.m24 + this.m13 * mat.m34
     + this.m14 * mat.m44);
226.     var m21 = (this.m21 * mat.m11 + this.m22 * mat.m21 + this.m23 * mat.m31
     + this.m24 * mat.m41);
227.     var m22 = (this.m21 * mat.m12 + this.m22 * mat.m22 + this.m23 * mat.m32
     + this.m24 * mat.m42);
228.     var m23 = (this.m21 * mat.m13 + this.m22 * mat.m23 + this.m23 * mat.m33
     + this.m24 * mat.m43);
229.     var m24 = (this.m21 * mat.m14 + this.m22 * mat.m24 + this.m23 * mat.m34
     + this.m24 * mat.m44);
230.     var m31 = (this.m31 * mat.m11 + this.m32 * mat.m21 + this.m33 * mat.m31
     + this.m34 * mat.m41);
231.     var m32 = (this.m31 * mat.m12 + this.m32 * mat.m22 + this.m33 * mat.m32
     + this.m34 * mat.m42);
232.     var m33 = (this.m31 * mat.m13 + this.m32 * mat.m23 + this.m33 * mat.m33
     + this.m34 * mat.m43);
233.     var m34 = (this.m31 * mat.m14 + this.m32 * mat.m24 + this.m33 * mat.m34
     + this.m34 * mat.m44);
234.     var m41 = (this.m41 * mat.m11 + this.m42 * mat.m21 + this.m43 * mat.m31
     + this.m44 * mat.m41);
235.     var m42 = (this.m41 * mat.m12 + this.m42 * mat.m22 + this.m43 * mat.m32
     + this.m44 * mat.m42);
236.     var m43 = (this.m41 * mat.m13 + this.m42 * mat.m23 + this.m43 * mat.m33
     + this.m44 * mat.m43);
```

```
237.     var m44 = (this.m41 * mat.m14 + this.m42 * mat.m24 + this.m43 * mat.m34
    + this.m44 * mat.m44);
238.     this.m11 = m11;
239.     this.m12 = m12;
240.     this.m13 = m13;
241.     this.m14 = m14;
242.     this.m21 = m21;
243.     this.m22 = m22;
244.     this.m23 = m23;
245.     this.m24 = m24;
246.     this.m31 = m31;
247.     this.m32 = m32;
248.     this.m33 = m33;
249.     this.m34 = m34;
250.     this.m41 = m41;
251.     this.m42 = m42;
252.     this.m43 = m43;
253.     this.m44 = m44
254. };
255. CanvasMatrix4.prototype.multLeft = function(mat) {
256.     var m11 = (mat.m11 * this.m11 + mat.m12 * this.m21 + mat.m13 * this.m31
    + mat.m14 * this.m41);
257.     var m12 = (mat.m11 * this.m12 + mat.m12 * this.m22 + mat.m13 * this.m32
    + mat.m14 * this.m42);
258.     var m13 = (mat.m11 * this.m13 + mat.m12 * this.m23 + mat.m13 * this.m33
    + mat.m14 * this.m43);
259.     var m14 = (mat.m11 * this.m14 + mat.m12 * this.m24 + mat.m13 * this.m34
    + mat.m14 * this.m44);
260.     var m21 = (mat.m21 * this.m11 + mat.m22 * this.m21 + mat.m23 * this.m31
    + mat.m24 * this.m41);
261.     var m22 = (mat.m21 * this.m12 + mat.m22 * this.m22 + mat.m23 * this.m32
    + mat.m24 * this.m42);
262.     var m23 = (mat.m21 * this.m13 + mat.m22 * this.m23 + mat.m23 * this.m33
    + mat.m24 * this.m43);
263.     var m24 = (mat.m21 * this.m14 + mat.m22 * this.m24 + mat.m23 * this.m34
    + mat.m24 * this.m44);
264.     var m31 = (mat.m31 * this.m11 + mat.m32 * this.m21 + mat.m33 * this.m31
    + mat.m34 * this.m41);
265.     var m32 = (mat.m31 * this.m12 + mat.m32 * this.m22 + mat.m33 * this.m32
    + mat.m34 * this.m42);
266.     var m33 = (mat.m31 * this.m13 + mat.m32 * this.m23 + mat.m33 * this.m33
    + mat.m34 * this.m43);
267.     var m34 = (mat.m31 * this.m14 + mat.m32 * this.m24 + mat.m33 * this.m34
    + mat.m34 * this.m44);
```

```
268.     var m41 = (mat.m41 * this.m11 + mat.m42 * this.m21 + mat.m43 * this.m31
     + mat.m44 * this.m41);
269.     var m42 = (mat.m41 * this.m12 + mat.m42 * this.m22 + mat.m43 * this.m32
     + mat.m44 * this.m42);
270.     var m43 = (mat.m41 * this.m13 + mat.m42 * this.m23 + mat.m43 * this.m33
     + mat.m44 * this.m43);
271.     var m44 = (mat.m41 * this.m14 + mat.m42 * this.m24 + mat.m43 * this.m34
     + mat.m44 * this.m44);
272.     this.m11 = m11;
273.     this.m12 = m12;
274.     this.m13 = m13;
275.     this.m14 = m14;
276.     this.m21 = m21;
277.     this.m22 = m22;
278.     this.m23 = m23;
279.     this.m24 = m24;
280.     this.m31 = m31;
281.     this.m32 = m32;
282.     this.m33 = m33;
283.     this.m34 = m34;
284.     this.m41 = m41;
285.     this.m42 = m42;
286.     this.m43 = m43;
287.     this.m44 = m44
288. };
289. CanvasMatrix4.prototype.ortho = function(left, right, bottom, top, near, fa
    r) {
290.     var tx = (left + right) / (left - right);
291.     var ty = (top + bottom) / (top - bottom);
292.     var tz = (far + near) / (far - near);
293.     var matrix = new CanvasMatrix4();
294.     matrix.m11 = 2 / (left - right);
295.     matrix.m12 = 0;
296.     matrix.m13 = 0;
297.     matrix.m14 = 0;
298.     matrix.m21 = 0;
299.     matrix.m22 = 2 / (top - bottom);
300.     matrix.m23 = 0;
301.     matrix.m24 = 0;
302.     matrix.m31 = 0;
303.     matrix.m32 = 0;
304.     matrix.m33 = -2 / (far - near);
305.     matrix.m34 = 0;
306.     matrix.m41 = tx;
```

```
307.     matrix.m42 = ty;
308.     matrix.m43 = tz;
309.     matrix.m44 = 1;
310.     this.multRight(matrix)
311. };
312. CanvasMatrix4.prototype.frustum = function(left, right, bottom, top, near,
    far) {
313.     var matrix = new CanvasMatrix4();
314.     var A = (right + left) / (right - left);
315.     var B = (top + bottom) / (top - bottom);
316.     var C = -(far + near) / (far - near);
317.     var D = -(2 * far * near) / (far - near);
318.     matrix.m11 = (2 * near) / (right - left);
319.     matrix.m12 = 0;
320.     matrix.m13 = 0;
321.     matrix.m14 = 0;
322.     matrix.m21 = 0;
323.     matrix.m22 = 2 * near / (top - bottom);
324.     matrix.m23 = 0;
325.     matrix.m24 = 0;
326.     matrix.m31 = A;
327.     matrix.m32 = B;
328.     matrix.m33 = C;
329.     matrix.m34 = -1;
330.     matrix.m41 = 0;
331.     matrix.m42 = 0;
332.     matrix.m43 = D;
333.     matrix.m44 = 0;
334.     this.multRight(matrix)
335. };
336. CanvasMatrix4.prototype.perspective = function(fovy, aspect, zNear, zFar) {

337.     var top = Math.tan(fovy * Math.PI / 360) * zNear;
338.     var bottom = -top;
339.     var left = aspect * bottom;
340.     var right = aspect * top;
341.     this.frustum(left, right, bottom, top, zNear, zFar)
342. };
343. CanvasMatrix4.prototype.lookat = function(eyex, eyey, eyez, centerx, center
    y, centerz, upx, upy, upz) {
344.     var matrix = new CanvasMatrix4();
345.     var zx = eyex - centerx;
346.     var zy = eyey - centery;
347.     var zz = eyez - centerz;
```

```
348.     var mag = Math.sqrt(zx * zx + zy * zy + zz * zz);
349.     if (mag) {
350.         zx /= mag;
351.         zy /= mag;
352.         zz /= mag
353.     }
354.     var yx = upx;
355.     var yy = upy;
356.     var yz = upz;
357.     xx = yy * zz - yz * zy;
358.     xy = -yx * zz + yz * zx;
359.     xz = yx * zy - yy * zx;
360.     yx = zy * xz - zz * xy;
361.     yy = -zx * xz + zz * xx;
362.     yx = zx * xy - zy * xx;
363.     mag = Math.sqrt(xx * xx + xy * xy + xz * xz);
364.     if (mag) {
365.         xx /= mag;
366.         xy /= mag;
367.         xz /= mag
368.     }
369.     mag = Math.sqrt(yx * yx + yy * yy + yz * yz);
370.     if (mag) {
371.         yx /= mag;
372.         yy /= mag;
373.         yz /= mag
374.     }
375.     matrix.m11 = xx;
376.     matrix.m12 = xy;
377.     matrix.m13 = xz;
378.     matrix.m14 = 0;
379.     matrix.m21 = yx;
380.     matrix.m22 = yy;
381.     matrix.m23 = yz;
382.     matrix.m24 = 0;
383.     matrix.m31 = zx;
384.     matrix.m32 = zy;
385.     matrix.m33 = zz;
386.     matrix.m34 = 0;
387.     matrix.m41 = 0;
388.     matrix.m42 = 0;
389.     matrix.m43 = 0;
390.     matrix.m44 = 1;
391.     matrix.translate(-eyex, -eyey, -eyez);
```

```
392.     this.multRight(matrix)
393. };
394. CanvasMatrix4.prototype._determinant2x2 = function(a, b, c, d) {
395.     return a * d - b * c
396. };
397. CanvasMatrix4.prototype._determinant3x3 = function(a1, a2, a3, b1, b2, b3,
    c1, c2, c3) {
398.     return a1 * this._determinant2x2(b2, b3, c2, c3) - b1 * this._determina
    nt2x2(a2, a3, c2, c3) + c1 * this._determinant2x2(a2, a3, b2, b3)
399. };
400. CanvasMatrix4.prototype._determinant4x4 = function() {
401.     var a1 = this.m11;
402.     var b1 = this.m12;
403.     var c1 = this.m13;
404.     var d1 = this.m14;
405.     var a2 = this.m21;
406.     var b2 = this.m22;
407.     var c2 = this.m23;
408.     var d2 = this.m24;
409.     var a3 = this.m31;
410.     var b3 = this.m32;
411.     var c3 = this.m33;
412.     var d3 = this.m34;
413.     var a4 = this.m41;
414.     var b4 = this.m42;
415.     var c4 = this.m43;
416.     var d4 = this.m44;
417.     return a1 * this._determinant3x3(b2, b3, b4, c2, c3, c4, d2, d3, d4) -
    b1 * this._determinant3x3(a2, a3, a4, c2, c3, c4, d2, d3, d4) + c1 * this._d
    eterminant3x3(a2, a3, a4, b2, b3, b4, d2, d3, d4) - d1 * this._determinant3x
    3(a2, a3, a4, b2, b3, b4, c2, c3, c4)
418. };
419. CanvasMatrix4.prototype._makeAdjoint = function() {
420.     var a1 = this.m11;
421.     var b1 = this.m12;
422.     var c1 = this.m13;
423.     var d1 = this.m14;
424.     var a2 = this.m21;
425.     var b2 = this.m22;
426.     var c2 = this.m23;
427.     var d2 = this.m24;
428.     var a3 = this.m31;
429.     var b3 = this.m32;
430.     var c3 = this.m33;
```

```
431.     var d3 = this.m34;
432.     var a4 = this.m41;
433.     var b4 = this.m42;
434.     var c4 = this.m43;
435.     var d4 = this.m44;
436.     this.m11 = this._determinant3x3(b2, b3, b4, c2, c3, c4, d2, d3, d4);
437.     this.m21 = -this._determinant3x3(a2, a3, a4, c2, c3, c4, d2, d3, d4);
438.     this.m31 = this._determinant3x3(a2, a3, a4, b2, b3, b4, d2, d3, d4);
439.     this.m41 = -this._determinant3x3(a2, a3, a4, b2, b3, b4, c2, c3, c4);
440.     this.m12 = -this._determinant3x3(b1, b3, b4, c1, c3, c4, d1, d3, d4);
441.     this.m22 = this._determinant3x3(a1, a3, a4, c1, c3, c4, d1, d3, d4);
442.     this.m32 = -this._determinant3x3(a1, a3, a4, b1, b3, b4, d1, d3, d4);
443.     this.m42 = this._determinant3x3(a1, a3, a4, b1, b3, b4, c1, c3, c4);
444.     this.m13 = this._determinant3x3(b1, b2, b4, c1, c2, c4, d1, d2, d4);
445.     this.m23 = -this._determinant3x3(a1, a2, a4, c1, c2, c4, d1, d2, d4);
446.     this.m33 = this._determinant3x3(a1, a2, a4, b1, b2, b4, d1, d2, d4);
447.     this.m43 = -this._determinant3x3(a1, a2, a4, b1, b2, b4, c1, c2, c4);
448.     this.m14 = -this._determinant3x3(b1, b2, b3, c1, c2, c3, d1, d2, d3);
449.     this.m24 = this._determinant3x3(a1, a2, a3, c1, c2, c3, d1, d2, d3);
450.     this.m34 = -this._determinant3x3(a1, a2, a3, b1, b2, b3, d1, d2, d3);
451.     this.m44 = this._determinant3x3(a1, a2, a3, b1, b2, b3, c1, c2, c3)
452. }
```

MV.js

```
1.  //=============================================================================
    //
2.  //
3.  //  lMV.js
4.  //
5.  //=============================================================================
    //
6.
7.  //-----------------------------------------------------------------------------
    --
8.  //
9.  //  Helper functions
10. //
11.
12.
13. //Float32Array.prototype.type = '';
14. /*
15. Float32Array.prototype.index = 0;
```

```
Float32Array.prototype.push = function(x){
  for(var i=0; i<x.length; i++) {
    this[i+this.index] = x[i];
  }
  this.index += x.length;
};

*/

function patch() {
  var out = new Array(4);
  for(var i = 0; i< 4; i++) out[i] = new Array(4);
  //for(var i = 0; i< 4; i++) for(var j=0; j<4;j++) out[i][j] = vec4();
  out.type = "patch";
  return out;
}

function curve() {
  var out = new Array(4);
  out.type = "curve";
  return out;
}

function MVbuffer(size) {
  var b = {};
  b.buf = new Float32Array(size);
  b.index = 0;
  b.push = function(x) {
    for(var i=0; i<x.length; i++) {
      b.buf[b.index+i] = x[i];
    }
    b.index += x.length;
    b.type = '';
  }
  return b;
}

function _argumentsToArray( args )
{
    return [].concat.apply( [], Array.prototype.slice.apply(args) );
}

function isVector(v) {
```

```
59.   if(v.type == "vec2" || v.type == "vec3" || v.type == "vec4") return true;

60.   return false;
61. }
62.
63. function isMatrix(v) {
64.   if(v.type == "mat2" || v.type == "mat3" || v.type == "mat4") return true;

65.   return false;
66. }
67. //----------------------------------------------------------------------------
    --
68.
69. function radians( degrees ) {
70.     return degrees * Math.PI / 180.0;
71. }
72.
73. //----------------------------------------------------------------------------
    --
74. //
75. //  Vector Constructors
76. //
77.
78. function vec2()
79. {
80.     //var result = _argumentsToArray( arguments );
81.     var out = new Float32Array(2);
82.     out.type = 'vec2';
83.
84.     switch ( arguments.length ) {
85.       case 0:
86.         out[0] = 0.0;
87.         out[1] = 0.0;
88.         break;
89.       case 1:
90.         if(isVector(arguments[0] && (arguments[0].type != 'vec2'))) {
91.         out[0] = arguments[0][0];
92.         out[1] = arguments[0][1];
93.       }
94.         break;
95.
96.       case 2:
97.         out[0] = arguments[0];
98.         out[1] = arguments[1];
```

```
99.         break;
100.    }
101.    return out;
102. }
103.
104. function vec3()
105. {
106. //var result = _argumentsToArray( arguments );
107.
108.    var out = new Float32Array(3);
109.    out.type = 'vec3';
110.
111.    switch ( arguments.length ) {
112.    case 0:
113.      out[0] = 0.0;
114.      out[1] = 0.0;
115.      out[2] = 0.0;
116.      return out;
117.    case 1:
118.    if(isVector(arguments[0]) && (arguments[0].type == "vec3")) {
119.      out[0] = arguments[0][0];
120.      out[1] = arguments[0][1];
121.      out[2] = arguments[0][2];
122.      return out;
123.    }
124.    case 3:
125.      out[0] = arguments[0];
126.      out[1] = arguments[1];
127.      out[2] = arguments[2];
128.      return out;
129.    default:
130.        throw "vec3: wrong arguments";
131.    }
132.
133.    return out;
134. }
135.
136. function vec4()
137. {
138.    var out = new Float32Array(4);
139.    out.type = 'vec4';
140.    switch ( arguments.length ) {
141.
142.      case 0:
```

```
143.
144.         out[0] = 0.0;
145.         out[1] = 0.0;
146.         out[2] = 0.0;
147.         out[3] = 0.0;
148.         return out;
149.
150.       case 1:
151.         if(isVector(arguments[0])) {
152.           //console.log("vec4 case 1", arguments[0].type)
153.           if(arguments[0].type == "vec4") {
154.             out[0] = arguments[0][0];
155.             out[1] = arguments[0][1];
156.             out[2] = arguments[0][2];
157.             out[3] = arguments[0][3];
158.             //console.log(out)
159.             return out;
160.           }
161.           else if(arguments[0].type == "vec3") {
162.             out[0] = arguments[0][0];
163.             out[1] = arguments[0][1];
164.             out[2] = arguments[0][2];
165.             out[3] = 1.0;
166.             return out;
167.           }
168.           else {
169.             out[0] = arguments[0][0];
170.             out[1] = arguments[0][1];
171.             out[2] = arguments[0][2];
172.             out[3] = arguments[0][3];
173.             return out;
174.           }
175.         }
176.
177.       case 2:
178.         if(typeof(arguments[0])=='number'&&arguments[1].type == 'vec3') {
179.           out[0] = arguments[0];
180.           out[1] = arguments[1][0];
181.           out[2] = arguments[1][1];
182.           out[3] = arguments[1][2];
183.           return out;
184.       }
185.       return out;
186.
```

```
187.       case 4:
188.       //console.log('in case 4')
189.       if(isVector(arguments[0])) {
190.         out[0] = arguments[0][0];
191.         out[1] = arguments[0][1];
192.         out[2] = arguments[0][2];
193.         out[3] = arguments[0][3];
194.         return out;
195.       }
196.         out[0] = arguments[0];
197.         out[1] = arguments[1];
198.         out[2] = arguments[2];
199.         out[3] = arguments[3];
200.         return out;
201.       case 3:
202.         out[0] = arguments[0][0];
203.         out[1] = arguments[0][1];
204.         out[2] = arguments[0][2];
205.         out[3] = 1.0;
206.         return out;
207.       default:
208.         throw "vec4: wrong arguments";
209.   }
210. }
211.
212. //----------------------------------------------------------------------------
    ---
213. //
214. //  Matrix Constructors
215. //
216.
217. function mat2()
218. {
219.     //var v = _argumentsToArray( arguments );
220.     var out = new Float32Array(4);
221.
222.     switch ( arguments.length ) {
223.     case 0:
224.         out[0]=out[3]=1.0;
225.         out[1]=out[2]=0.0;
226.         break;
227.     case 1:
228.       if(arguments[0].type == 'mat2') {
229.         out[0] = arguments[0][0];
```

```
230.         out[1] = arguments[0][1];
231.         out[2] = arguments[0][2];
232.         out[3] = arguments[0][3];
233.         break;
234.       }
235.
236.     case 4:
237.         out[0] = arguments[0];
238.         out[1] = arguments[1];
239.         out[2] = arguments[2];
240.         out[3] = arguments[3];
241.         break;
242.      default:
243.          throw "mat2: wrong arguments";
244.     }
245.     out.type = 'mat2';
246.
247.     return out;
248. }
249.
250. //-------------------------------------------------------------------------
    ---
251.
252. function mat3()
253. {
254.     // v = _argumentsToArray( arguments );
255.
256.     var out = new Float32Array(9);
257.     switch ( arguments.length ) {
258.       case 0:
259.           out[0]=out[4]=out[8]=1.0;
260.           out[1]=out[2]=out[3]=out[5]=out[6]=out[7]=0.0;
261.           break;
262.     case 1:
263.          for(var i=0; i<9; i++) {
264.            out[i]=v[0][i];
265.          }
266.         break;
267.
268.     case 9:
269.         for(var i=0; i<9; i++) {
270.           out[i] = arguments[i];
271.         }
272.         break;
```

```
    default:
        throw "mat3: wrong arguments";
    }
    out.type = 'mat3';

    return out;
}

//----------------------------------------------------------------------------

function mat4()
{
    //var v = _argumentsToArray( arguments );

    var out = new Float32Array(16);
    switch ( arguments.length ) {
    case 0:
      out[0]=out[5]=out[10]=out[15] = 1.0;
      out[1]=out[2]=out[3]=out[4]=out[6]=out[7]=out[8]=out[9]=out[11]=out[12]=out[13]=out[14]=0.0;
      break;

    case 1:
      for(var i=0; i<16; i++) {
        out[i] = arguments[0][i];
      }
      break;

    case 4:
      if(arguments[0].type == "vec4") {
      for( var i=0; i<4; i++)
        for(var j=0; j<4; j++)
          out[4*i+j] = arguments[i][j];
       break;
      }

    case 16:
      for(var i=0; i<16; i++) {
        out[i] = arguments[i];
      }
      break;
    }
    out.type = "mat4";
```

```
315.
316.     return out;
317. }
318.
319. //----------------------------------------------------------------------------
320. //
321. //  Generic Mathematical Operations for Vectors and Matrices
322. //
323.
324. function equal( u, v )
325. {
326.     if(!(isMatrix(u)&&isMatrix(v) || (isVector(u)&&isVector(v))))
327.       throw "equal: at least one input not a vec or mat";
328.     if ( u.type != v.type ) throw "equal: types different";
329.
330.         for ( var i = 0; i < u.length; ++i ) {
331.             if ( u[i] !== v[i] ) { return false; }
332.         }
333.
334.     return true;
335. }
336.
337.
338. //----------------------------------------------------------------------------
339.
340. function add( u, v )
341. {
342.
343.   if ( u.type != v.type ) {
344.       throw "add(): trying to add different types";
345.   }
346.
347.   var result = new Float32Array(u.length);
348.   result.type = u.type;
349.   for(var i=0; i<u.length; i++) {
350.       result[i] = u[i] + v[i];
351.   }
352.   return result;
353. }
354.
355. //----------------------------------------------------------------------------
```

```
356.
357. function subtract( u, v )
358. {
359.   if ( u.type != v.type ) {
360.       throw "subtract(): trying to subtract different types";
361.   }
362.
363.   var result = new Float32Array(u.length);
364.   result.type = u.type;
365.   for(var i=0; i<u.length; i++) {
366.       result[i] = u[i] - v[i];
367.   }
368.   return result;
369. }
370.
371. //----------------------------------------------------------------------------
   ---
372.
373. function mult( u, v )
374. {
375.
376.   if(typeof(u)=="number" && (isMatrix(v)||isVector(v))) {
377.     result = new Float32Array(v.length);
378.     result.type = v.type;
379.     for(var i =0; i<v.length; i++) {
380.       result[i] = u*v[i];
381.     }
382.     return result;
383.   }
384.   if(typeof(v)=="number" && (isMatrix(u)||isVector(u))) {
385.     result = new Float32Array(u.length);
386.     result.type = u.type;
387.     for(var i = 0; i < u.length; i++) {
388.       result[i] = v * u[i];
389.     }
390.     return result;
391.   }
392.   if(u.type=='mat2' && v.type == 'vec2') {
393.     var result = vec2();
394.     result[0] =u[0]*v[0]+u[1]*v[1];
395.     result[1] =u[2]*v[0]+u[3]*v[1];
396.     return result;
397.   }
398.   if(u.type=='mat3'&& v.type=='vec3') {
```

```
399.     var result = vec3();
400.     result[0] =u[0]*v[0]+u[1]*v[1]+u[2]*v[2];
401.     result[1] =u[3]*v[0]+u[4]*v[1]+u[5]*v[2];
402.     result[2] =u[6]*v[0]+u[7]*v[1]+u[8]*v[2];
403.     return result;
404.   }
405.   if(u.type=='mat4'&& v.type=='vec4')  {
406.     var result = vec4();
407.     result[0] =u[0]*v[0]+u[1]*v[1]+u[2]*v[2]+u[3]*v[3];
408.     result[1] =u[4]*v[0]+u[5]*v[1]+u[6]*v[2]+u[7]*v[3];
409.     result[2] =u[8]*v[0]+u[9]*v[1]+u[10]*v[2]+u[11]*v[3];
410.     result[3] =u[12]*v[0]+u[13]*v[1]+u[14]*v[2]+u[15]*v[3];
411.     return result;
412.   }
413.  if (u.type=='mat2'&&v.type=='mat2'){
414.     result = mat2();
415.     result[0] = u[0]*v[0]+u[1]*v[2];
416.     result[1] = u[0]*v[1]+u[1]*v[3];
417.     result[2] = u[2]*v[0]+u[3]*v[2];
418.     result[3] = u[2]*v[1]+u[3]*v[3];
419.     return result;
420.   }
421.  if (u.type=='mat3'&&v.type=='mat3'){
422.     result = mat3();
423.     result[0] = u[0]*v[0]+u[1]*v[3]+u[2]*v[6];
424.     result[1] = u[0]*v[1]+u[1]*v[4]+u[2]*v[7];
425.     result[2] = u[0]*v[2]+u[1]*v[5]+u[2]*v[8];
426.     result[3] = u[3]*v[0]+u[4]*v[3]+u[5]*v[6];
427.     result[4] = u[3]*v[1]+u[4]*v[4]+u[5]*v[7];
428.     result[5] = u[3]*v[2]+u[4]*v[5]+u[5]*v[8];
429.     result[6] = u[6]*v[0]+u[7]*v[3]+u[8]*v[6];
430.     result[7] = u[6]*v[1]+u[7]*v[4]+u[8]*v[7];
431.     result[8] = u[6]*v[2]+u[7]*v[5]+u[8]*v[8];
432.   }
433.   else if (u.type=='mat4'&&v.type=='mat4'){
434.     result = mat4();
435.     result[0] = u[0]*v[0]+u[1]*v[4]+u[2]*v[8]+u[3]*v[12];
436.     result[1] = u[0]*v[1]+u[1]*v[5]+u[2]*v[9]+u[3]*v[13];
437.     result[2] = u[0]*v[2]+u[1]*v[6]+u[2]*v[10]+u[3]*v[14];
438.     result[3] = u[0]*v[3]+u[1]*v[7]+u[2]*v[11]+u[3]*v[15];
439.
440.     result[4] = u[4]*v[0]+u[5]*v[4]+u[6]*v[8]+u[7]*v[12];
441.     result[5] = u[4]*v[1]+u[5]*v[5]+u[6]*v[9]+u[7]*v[13];
442.     result[6] = u[4]*v[2]+u[5]*v[6]+u[6]*v[10]+u[7]*v[14];
```

```
443.     result[7] = u[4]*v[3]+u[5]*v[7]+u[6]*v[11]+u[7]*v[15];
444.
445.     result[8] = u[8]*v[0]+u[9]*v[4]+u[10]*v[8]+u[11]*v[12];
446.     result[9] = u[8]*v[1]+u[9]*v[5]+u[10]*v[9]+u[11]*v[13];
447.     result[10] = u[8]*v[2]+u[9]*v[6]+u[10]*v[10]+u[11]*v[14];
448.     result[11] = u[8]*v[3]+u[9]*v[7]+u[10]*v[11]+u[11]*v[15];
449.
450.     result[12] = u[12]*v[0]+u[13]*v[4]+u[14]*v[8]+u[15]*v[12];
451.     result[13] = u[12]*v[1]+u[13]*v[5]+u[14]*v[9]+u[15]*v[13];
452.     result[14] = u[12]*v[2]+u[13]*v[6]+u[14]*v[10]+u[15]*v[14];
453.     result[15] = u[12]*v[3]+u[13]*v[7]+u[14]*v[11]+u[15]*v[15];
454.
455.     return result;
456.   }
457.   if (u.type=='vec3'&&v.type=='vec3'){
458.     var result = vec3(u[0]*v[0], u[1]*v[1], u[2]*v[2]);
459.     return result;
460.   }
461.   if (u.type=='vec4'&&v.type=='vec4'){
462.     var result = vec4(u[0]*v[0], u[1]*v[1], u[2]*v[2], u[3]*v[3]);
463.     return result;
464.   }
465.     throw "mult(): trying to mult incompatible types";
466. }
467.
468.
469. //----------------------------------------------------------------------------
    ---
470. //
471. //  Basic Transformation Matrix Generators
472. //
473.
474. function translate( x, y, z )
475. {
476.     if(arguments.length!=2 && arguments.length != 3) {
477.       throw "translate(): not a mat3 or mat4";
478.     }
479.     if(arguments.length == 2) {
480.       var result = mat3();
481.       result[2] = x;
482.       result[5] = y;
483.
484.       return result;
485.     }
```

```
486.
487.     var result = mat4();
488.
489.     result[3] = x;
490.     result[7] = y;
491.     result[11] = z;
492.
493.     return result;
494.
495. }
496.
497. //----------------------------------------------------------------------------
    ---
498.
499. function rotate( angle, axis )
500. {
501.     if ( axis.length == 3 ) {
502.         axis = vec3(axis[0], axis[1], axis[2] );
503.     }
504.     if(arguments.length == 4) {
505.         axis = vec3(arguments[1], arguments[2], arguments[3]);
506.     }
507.     if(axis.type != 'vec3') throw "rotate: axis not a vec3";
508.     var v = normalize( axis );
509.
510.     var x = v[0];
511.     var y = v[1];
512.     var z = v[2];
513.
514.     var c = Math.cos( radians(angle) );
515.     var omc = 1.0 - c;
516.     var s = Math.sin( radians(angle) );
517.
518.     var result = mat4(
519.         x*x*omc + c,   x*y*omc - z*s, x*z*omc + y*s, 0.0 ,
520.         x*y*omc + z*s, y*y*omc + c,   y*z*omc - x*s, 0.0 ,
521.         x*z*omc - y*s, y*z*omc + x*s, z*z*omc + c,   0.0 ,
522.         0.0,           0.0,           0.0,           1.0
523.     );
524.     return result;
525. }
526.
527. function rotateX(theta) {
528.   var c = Math.cos( radians(theta) );
```

```
529.   var s = Math.sin( radians(theta) );
530.   var rx = mat4( 1.0,  0.0,  0.0, 0.0,
531.                  0.0,  c,    -s,  0.0,
532.                  0.0,  s,    c,   0.0,
533.                  0.0,  0.0,  0.0, 1.0 );
534.   return rx;
535. }
536.
537. function rotateY(theta) {
538.   var c = Math.cos( radians(theta) );
539.   var s = Math.sin( radians(theta) );
540.   var ry = mat4( c,     0.0,    s,  0.0,
541.                  0.0,   1.0,    0.0,    0.0,
542.                  -s,    0.0,    c,      0.0,
543.                  0.0,   0.0,    0.0,    1.0 );
544.   return ry;
545. }
546.
547. function rotateZ(theta) {
548.   var c = Math.cos( radians(theta) );
549.   var s = Math.sin( radians(theta) );
550.   var rz = mat4( c,     -s,     0.0,    0.0,
551.                  s,     c,      0.0,    0.0,
552.                  0.0,   0.0,    1.0,    0.0,
553.                  0.0,   0.0,    0.0,    1.0 );
554.   return rz;
555. }
556. //----------------------------------------------------------------------------
    ---
557.
558.
559. function scale( )
560. {
561.
562.     if(arguments.length == 2 && isVector(arguments[1])) {
563.       result = new Float32Array(arguments[1].length);
564.       result.type = arguments[1].type;
565.       for(var i=0; i<arguments[1].length; i++)
566.           result[i] = arguments[0]*arguments[1][i];
567.       return result;
568.     }
569.
570.
571.     if(arguments.length == 3) {
```

```
572.
573.         var result = mat4();
574.         result[0] = arguments[0];
575.         result[5] = arguments[1];
576.         result[10] = arguments[2];
577.         result[15] = 1.0;
578.         return result;
579.     }
580.
581.     throw "scale: wrong arguments";
582.
583. }
584.
585.
586. //----------------------------------------------------------------------------
    ---
587. //
588. //  ModelView Matrix Generators
589. //
590.
591. function lookAt( eye, at, up )
592. {
593.     if ( eye.type != 'vec3') {
594.         throw "lookAt(): first parameter [eye] must be an a vec3";
595.     }
596.
597.     if ( at.type != 'vec3') {
598.         throw "lookAt(): first parameter [at] must be an a vec3";
599.     }
600.
601.     if (up.type != 'vec3') {
602.         throw "lookAt(): first parameter [up] must be an a vec3";
603.     }
604.
605.     if ( equal(eye, at) ) {
606.         return mat4();
607.     }
608.
609.     var v = normalize( subtract(at, eye) );  // view direction vector
610.     var n = normalize( cross(v, up) );       // perpendicular vector
611.     var u = normalize( cross(n, v) );        // "new" up vector
612.
613.     v = negate( v );
614.
```

```
615.     var result = mat4(
616.         n[0], n[1], n[2], -dot(n, eye),
617.         u[0], u[1], u[2], -dot(u, eye),
618.         v[0], v[1], v[2], -dot(v, eye),
619.         0.0,  0.0,  0.0,  1.0
620.     );
621. 
622.     return result;
623. }
624. 
625. //----------------------------------------------------------------------------
     ---
626. //
627. //  Projection Matrix Generators
628. //
629. 
630. function ortho( left, right, bottom, top, near, far )
631. {
632.     if ( left == right ) { throw "ortho(): left and right are equal"; }
633.     if ( bottom == top ) { throw "ortho(): bottom and top are equal"; }
634.     if ( near == far )   { throw "ortho(): near and far are equal"; }
635. 
636.     var w = right - left;
637.     var h = top - bottom;
638.     var d = far - near;
639. 
640.     var result = mat4();
641. 
642.     result[0] = 2.0 / w;
643.     result[5] = 2.0 / h;
644.     result[10] = -2.0 / d;
645. 
646.     result[3] = -(left + right) / w;
647.     result[7] = -(top + bottom) / h;
648.     result[11] = -(near + far) / d;
649.     result[15] = 1.0;
650. 
651.     return result;
652. }
653. 
654. function ortho2D( left, right, bottom, top ){
655.     return ortho(left, right, bottom, top, -1.0, 1.0);
656. }
657. 
```

```
658. //----------------------------------------------------------------------------
659.
660. function perspective( fovy, aspect, near, far )
661. {
662.     var f = 1.0 / Math.tan( radians(fovy) / 2 );
663.     var d = far - near;
664.
665.     var result = mat4();
666.     result[0] = f / aspect;
667.     result[5] = f;
668.     result[10] = -(near + far) / d;
669.     result[11] = -2 * near * far / d;
670.     result[14] = -1;
671.     result[15] = 0.0;
672.
673.     return result;
674. }
675.
676. //----------------------------------------------------------------------------
677. //
678. //  Matrix Functions
679. //
680.
681. function transpose( m )
682. {
683.     if(m.type == 'patch') {
684.         var out = patch()
685.         for(var i=0; i<4; i++) out[i] = new Array(4);
686.         for(var i=0; i<4; i++)
687.           for(var j=0; j<4; j++) out[i][j] = m[j][i];
688.         return out;
689.     }
690.
691.     switch(m.type) {
692.       case 'mat2':
693.         var result = mat2(m[0], m[2],
694.                           m[1], m[3]
695.                          );
696.         return result;
697.         break;
698.
699.       case 'mat3':
```

```
700.         var result = mat3(m[0], m[3], m[6],
701.                           m[1], m[4], m[7],
702.                           m[2], m[5], m[8]
703.                         );
704.         return result;
705.         break;
706.
707.       case 'mat4':
708.         var result = mat4(m[0], m[4], m[8], m[12],
709.                           m[1], m[5], m[9], m[13],
710.                           m[2], m[6], m[10], m[14],
711.                           m[3], m[7], m[11], m[15]
712.         );
713.         return result;
714.         break;
715.
716.       default: throw "transpose(): trying to transpose a non-matrix";
717.     }
718. }
719.
720.
721. //-------------------------------------------------------------------------
    ---
722. //
723. //  Vector Functions
724. //
725.
726. function dot( u, v )
727. {
728.
729.     if ( u.type != v.type ) {
730.       throw "dot(): types are not the same ";
731.     }
732.     if (u.type != 'vec2' && u.type != 'vec3' && u.type != 'vec4') {
733.       throw "dot(): not a vector ";
734.     }
735.
736.     var sum = 0.0;
737.     for ( var i = 0; i < u.length; i++ ) {
738.         sum += u[i] * v[i];
739.     }
740.     return sum;
741. }
742.
```

```
743. //----------------------------------------------------------------------------
744.
745. function negate( u )
746. {
747.     if (u.type != 'vec2' && u.type != 'vec3' && u.type != 'vec4') {
748.         throw "negate(): not a vector ";
749.     }
750.     var result = new Float32Array(u.length);
751.     result.type = u.type;
752.     for ( var i = 0; i < u.length; ++i ) {
753.         result[i] = -u[i];
754.     }
755.     return result;
756. }
757.
758. //----------------------------------------------------------------------------
759.
760. function cross( u, v )
761. {
762.     if ( u.type == 'vec3' && v.type == 'vec3') {
763.       var result = vec3(
764.           u[1]*v[2] - u[2]*v[1],
765.           u[2]*v[0] - u[0]*v[2],
766.           u[0]*v[1] - u[1]*v[0]
767.       );
768.       return result;
769.     }
770.
771.     if ( v.type == 'vec4' && v.type == 'vec4') {
772.       var result = vec3(
773.           u[1]*v[2] - u[2]*v[1],
774.           u[2]*v[0] - u[0]*v[2],
775.           u[0]*v[1] - u[1]*v[0]
776.       );
777.       return result;
778.     }
779.
780.     throw "cross: types aren't matched vec3 or vec4";
781. }
782.
783. //----------------------------------------------------------------------------
```

```
784.
785. function length( u )
786. {
787.     return Math.sqrt( dot(u, u) );
788. }
789.
790. //----------------------------------------------------------------------------
    ---
791.
792. function normalize( u, excludeLastComponent )
793. {
794.     if(u.type != 'vec3' && u.type != 'vec4') {
795.
796.       throw "normalize: not a vector type";
797.     }
798.     switch(u.type) {
799.       case 'vec2':
800.         var len = Math.sqrt(u[0]*u[0]+u[1]*u[1]);
801.         var result = vec2(u[0]/len, u[1]/len);
802.         return result;
803.       break;
804.       case 'vec3':
805.         if(excludeLastComponent) {
806.           var len = Math.sqrt(u[0]*u[0]+u[1]*u[1]);
807.           var result = vec3(u[0]/len, u[1]/len, u[2]);
808.           return result;
809.           break;
810.         }
811.         else {
812.         var len = Math.sqrt(u[0]*u[0]+u[1]*u[1]+u[2]*u[2]);
813.         var result = vec3(u[0]/len, u[1]/len, u[2]/len);
814.         return result;
815.         break;
816.       }
817.       case 'vec4':
818.       if(excludeLastComponent) {
819.         var len = Math.sqrt(u[0]*u[0]+u[1]*u[1]+u[2]*u[2]);
820.         var result = vec4(u[0]/len, u[1]/len, u[2]/len, u[3]);
821.         return result;
822.         break;
823.       }
824.       else {
825.         var len = Math.sqrt(u[0]*u[0]+u[1]*u[1]+u[2]*u[2]+u[3]*u[3]);
826.         var result = vec4(u[0]/len, u[1]/len, u[2]/len, u[3]/len);
```

```
827.         return result;
828.         break;
829.       }
830.     }
831. }
832.
833. //----------------------------------------------------------------------------
834.
835. function mix( u, v, s )
836. {
837.     //console.log(u,v,s);
838.     if ( typeof(s) !== "number" ) {
839.         throw "mix: the last paramter " + s + " must be a number";
840.     }
841.     if(typeof(u)=='number'&&typeof(v)=='number') {
842.       return (1.0-s)*u + s*v;
843.     }
844.
845.     if ( u.length != v.length ) {
846.
847.         throw "vector dimension mismatch";
848.     }
849.
850.     var result = new Float32Array(u.length);
851.     for ( var i = 0; i < u.length; ++i ) {
852.         result[i] =  (1.0 - s) * u[i] + s * v[i] ;
853.     }
854.     result.type = u.type;
855.     return result;
856. }
857.
858. //----------------------------------------------------------------------------
859. //
860. // Vector and Matrix utility functions
861. //
862.
863.
864. function flatten( v )
865. {
866.
867.     //if(!Array.isArray(v)) return v;
868.
```

```
869.     if ( isMatrix(v) ) {
870.         v = transpose(v);
871.     }
872.
873.     if(typeof(v[0])=='number'){
874.       var floats = new Float32Array( v.length );
875.
876.       for(var i = 0; i<v.length; i++)
877.           floats[i] = v[i];
878.
879.       return floats;
880.     }
881.
882.     var floats = new Float32Array( v.length*v[0].length  );
883.
884.     for(var i = 0; i<v.length; i++) for(var j=0; j<v[0].length; j++) {
885.       floats[i*v[0].length+j] = v[i][j];
886.
887.     }
888.
889.     return floats;
890. }
891.
892. //----------------------------------------------------------------------------
    ---
893. /*
894. var sizeof = {
895.     'vec2' : new Float32Array( flatten(vec2()) ).byteLength,
896.     'vec3' : new Float32Array( flatten(vec3()) ).byteLength,
897.     'vec4' : new Float32Array( flatten(vec4()) ).byteLength,
898.     'mat2' : new Float32Array( flatten(mat2()) ).byteLength,
899.     'mat3' : new Float32Array( flatten(mat3()) ).byteLength,
900.     'mat4' : new Float32Array( flatten(mat4()) ).byteLength
901. };
902. */
903. //----------------------------------------------------------------------------
    ---
904.
905. function printm(m)
906. {
907.     switch(m.type) {
908.       case 'mat2':
909.         console.log(m[0], m[1]);
910.         console.log(m[2], m[3]);
```

```
911.         break;
912.       case 'mat3':
913.         console.log(m[0], m[1], m[2]);
914.         console.log(m[3], m[4], m[5]);
915.         console.log(m[6], m[7], m[8]);
916.         break;
917.       case 'mat4':
918.         console.log(m[0], m[1], m[2], m[3]);
919.         console.log(m[4], m[5], m[6], m[7]);
920.         console.log(m[8], m[9], m[10], m[11]);
921.         console.log(m[12], m[13], m[14], m[15]);
922.         break;
923.       case 'patch':
924.         for(var i=0;i<4;i++)
925.           console.log(m[i][0], m[i][1], m[i][2], m[i][3]);
926.         break;
927.       default: throw "printm: not a matrix";
928.     }
929. }
930.
931. // determinants
932.
933. function det2(m)
934. {
935.     return m[0]*m[3]-m[1]*m[2];
936. }
937.
938. function det3(m)
939. {
940.     var d = m[0]*m[4]*m[8]
941.           + m[1]*m[5]*m[6]
942.           + m[2]*m[3]*m[7]
943.           - m[6]*m[4]*m[2]
944.           - m[3]*m[1]*m[8]
945.           - m[0]*m[5]*m[7]
946.           ;
947.     return d;
948. }
949.
950. //-------------------------------------------------------
951.
952. function det4(m)
953. {
954.     var m0 = mat3(
```

```
955.          m[5], m[6], m[7],
956.          m[9], m[10], m[11],
957.          m[13], m[14], m[15]
958.      );
959.      var m1 = mat3(
960.          m[4], m[6], m[7],
961.          m[8], m[10], m[11],
962.          m[12], m[14], m[15]
963.      );
964.      var m2 = mat3(
965.          m[4], m[5], m[7],
966.          m[8], m[9], m[11],
967.          m[12], m[14], m[15]
968.      );
969.      var m3 = mat3(
970.          m[4], m[5], m[6],
971.          m[8], m[9], m[10],
972.          m[12], m[13], m[14]
973.      );
974.
975.      return m[0]*det3(m0) - m[1]*det3(m1)
976.          + m[2]*det3(m2) - m[3]*det3(m3);
977.
978. }
979.
980. function det(m)
981. {
982.      switch(m.type) {
983.        case 'mat2':
984.         return det3(m);
985.       case 'mat3':
986.          return det3(m);
987.       case 'mat4':
988.         return det4(m);
989.       default:
990.         throw "det: not a matrix";
991.      }
992. }
993.
994. //-------------------------------------------------------
995.
996. // inverses
997.
998. function inverse2(m)
```

```
999. {
1000.          var a = mat2();
1001.          var d = det2(m);
1002.          a[0] = m[3]/d;
1003.          a[1] = -m[2]/d;
1004.          a[2] = -m[1]/d;
1005.          a[3] = m[0]/d;
1006.          return a;
1007.      }
1008.
1009.      //------------------------------------------------------------
1010.
1011.      function inverse3(m)
1012.      {
1013.          var a = mat3();
1014.          var d = det3(m);
1015.
1016.          var a00 = mat2(
1017.             m[4], m[5],
1018.             m[7], m[8]
1019.          );
1020.          var a01 = mat2(
1021.             m[3], m[5],
1022.             m[6], m[8]
1023.          );
1024.          var a02 = mat2(
1025.             m[3], m[4],
1026.             m[6], m[7]
1027.          );
1028.          var a10 = mat2(
1029.             m[1], m[2],
1030.             m[7], m[8]
1031.          );
1032.          var a11 = mat2(
1033.             m[0], m[2],
1034.             m[6], m[8]
1035.          );
1036.          var a12 = mat2(
1037.             m[0], m[1],
1038.             m[6], m[7]
1039.          );
1040.          var a20 = mat2(
1041.             m[1], m[2],
1042.             m[4], m[5]
```

```
1043.         );
1044.         var a21 = mat2(
1045.            m[0], m[2],
1046.            m[3], m[5]
1047.         );
1048.         var a22 = mat2(
1049.            m[0], m[1],
1050.            m[3], m[4]
1051.         );
1052.
1053.        a[0] = det2(a00)/d;
1054.        a[1] = -det2(a10)/d;
1055.        a[2] = det2(a20)/d;
1056.        a[3] = -det2(a01)/d;
1057.        a[4] = det2(a11)/d;
1058.        a[5] = -det2(a21)/d;
1059.        a[6] = det2(a02)/d;
1060.        a[7] = -det2(a12)/d;
1061.        a[8] = det2(a22)/d;
1062.
1063.        return a;
1064.
1065.    }
1066.
1067.    //------------------------------------------------------------
1068.
1069.    function inverse4(m)
1070.    {
1071.        var a = mat4();
1072.        var d = det4(m);
1073.
1074.        var a00 = mat3(
1075.           m[5], m[6], m[7],
1076.           m[9], m[10], m[11],
1077.           m[13], m[14], m[15]
1078.        );
1079.        var a01 = mat3(
1080.           m[4], m[6], m[7],
1081.           m[8], m[10], m[11],
1082.           m[12], m[14], m[15]
1083.        );
1084.        var a02 = mat3(
1085.           m[4], m[5], m[7],
1086.           m[8], m[9], m[11],
```

```
            m[12], m[13], m[15]
        );
        var a03 = mat3(
            m[4], m[5], m[6],
            m[8], m[9], m[10],
            m[12], m[13], m[14]
        );
        var a10 = mat3(
            m[1], m[2], m[3],
            m[9], m[10], m[11],
            m[13], m[14], m[15]
        );
        var a11 = mat3(
            m[0], m[2], m[3],
            m[8], m[10], m[11],
            m[12], m[14], m[15]
        );
        var a12 = mat3(
            m[0], m[1], m[3],
            m[8], m[9], m[11],
            m[12], m[13], m[15]
        );
        var a13 = mat3(
            m[0], m[1], m[2],
            m[8], m[9], m[10],
            m[12], m[13], m[14]
        );
        var a20 = mat3(
            m[1], m[2], m[3],
            m[5], m[6], m[7],
            m[13], m[14], m[15]
        );
        var a21 = mat3(
            m[0], m[2], m[3],
            m[4], m[6], m[7],
            m[12], m[14], m[15]
        );
        var a22 = mat3(
            m[0], m[1], m[3],
            m[4], m[5], m[7],
            m[12], m[13], m[15]
        );
        var a23 = mat3(
            m[0], m[1], m[2],
```

```
            m[4], m[5], m[6],
            m[12], m[13], m[14]
        );

        var a30 = mat3(
            m[1], m[2], m[3],
            m[5], m[6], m[7],
            m[9], m[10], m[11]
        );
        var a31 = mat3(
            m[0], m[2], m[3],
            m[4], m[6], m[7],
            m[8], m[10], m[11]
        );
        var a32 = mat3(
            m[0], m[1], m[3],
            m[4], m[5], m[7],
            m[8], m[9], m[11]
        );
        var a33 = mat3(
            m[0], m[1], m[2],
            m[4], m[5], m[6],
            m[8], m[9], m[10]
        );


    a[0] = det3(a00)/d;
    a[1] = -det3(a10)/d;
    a[2] = det3(a20)/d;
    a[3] = -det3(a30)/d;
    a[4] = -det3(a01)/d;
    a[5] = det3(a11)/d;
    a[6] = -det3(a21)/d;
    a[7] = det3(a31)/d;
    a[8] = det3(a02)/d;
    a[9] = -det3(a12)/d;
    a[10] = det3(a22)/d;
    a[11] = -det3(a32)/d;
    a[12] = -det3(a03)/d;
    a[13] = det3(a13)/d;
    a[14] = -det3(a23)/d;
    a[15] = det3(a33)/d;

```

```
1175.       return a;
1176.    }
1177.
1178.    //-----------------------------------------------------------
1179.
1180.    function inverse(m)
1181.    {
1182.       switch(m.type) {
1183.         case 'mat2':
1184.          return inverse2(m);
1185.          case 'mat3':
1186.           return inverse3(m);
1187.          case 'mat4':
1188.            return inverse4(m);
1189.          default:
1190.            throw "inverse: not a matrix";
1191.       }
1192.    }
1193.
1194.    //-----------------------------------------------------------
1195.
1196.    // normal matrix
1197.
1198.    function normalMatrix(m, flag)
1199.    {
1200.        if(m.type != 'mat4') {
1201.            throw "normalMatrix: not a mat4";
1202.        }
1203.        var a = mat3(m[0], m[1], m[2],
1204.                     m[4], m[5], m[6],
1205.                     m[8], m[9], m[10]);
1206.        return inverse(transpose(a));
1207.    }
```