## おまけ

- この資料は授業とは関係ないので，参考程度に読んでください
- 解説はしません

## WebGL

- WebGLはOpenGL風のライブラリであり，ウェブページとして，ウェブブラウザ上で3DCGを表示することを可能にしたものである
- 言語はjavascript
- OSに依存しないし，コンパイラも必要ないのが利点
- ウェブブラウザで見ることが出来るので，ウェブサーバにアップロードして世界中の人に自分のソフトを見てもらえるという利点がある

## プログラミングスタイル

- WebGLプログラミングのスタイルは大きく分けて3つ
    - three.jsを使うスタイル，Away3Dを使うスタイル，three.jsもAway3Dも使わないスタイル
    - それ以外のスタイル（例えば，Unityで作ったものをWebGL用に出力する，など）もある
- three.jsやAway3Dを使うスタイル
    - 高機能
    - GLUT+OpenGLプログラミングとはだいぶ違う
- three.jsやAway3Dを使わないスタイル
    - OpenGLをよく理解していることが条件
        - 特に，シェーダーによるプログラミングを勉強してからのほうがいい
        - GLUT+OpenGLとは異なるプログラミングスタイル
- いずれも，javascript/TypeScriptの知識が必要となる
- CGプログラミング初学者にはややこしいので，まずはGLUT+OpenGLでCGプログラミングに慣れてからWebGLに取り組んだほうがいい

## サンプルソースコード

- 私の作成したサンプルソースコードを載せるが，参考にしないように
    - 私はWebGL初心者・シェーダープログラミング初心者・javascript初心者である
    - このサンプルはかなり雑に作ってある
- WebGLやjavascriptのデフォルト機能には行列ライブラリがないので，別途導入する必要がある
    - 今回使った行列ライブラリはwgld.orgのものである
      https://wgld.org/d/library/l001.html
- サンプルプログラムはindex.htmlとgame.jsとminMatrix.jsの3つのファイルから構成され，3つのファイルを同じフォルダにおいて，index.htmlをブラウザで表示させればよい
    - 文字コードはUTF8を想定している
    - three.jsやAway3Dを使わないスタイル

## index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<script src="minMatrix.js" type="text/javascript"></script>
<script id="fs" type="x-shader/x-fragment">
precision mediump float;

varying vec4 vColor;

void main(void){
    gl_FragColor = vColor;
}
</script>
<script id="vs" type="x-shader/x-vertex">
attribute vec3 position;
attribute vec3 normal;
attribute vec4 color;
uniform   mat4 mvpMatrix;
uniform   mat4 invMatrix;
uniform   vec3 lightDirection;
varying   vec4 vColor;

void main(void){
    vec3  invLight = normalize(invMatrix * vec4(lightDirection, 0.0)).xyz;
    float diffuse  = clamp(dot(normal, invLight), 0.1, 1.0);
    vColor         = color * vec4(vec3(diffuse), 1.0);
    gl_Position    = mvpMatrix * vec4(position, 1.0);
}
</script>
<script src="game.js" type="text/javascript"></script>
<title>WebGLプログラミング</title>
</head>

<body bgcolor="#FFFFFF">

<h1 align="center">WebGLプログラミング</h1>

<hr />

<p>
<canvas id="canvas"></canvas>
</p>

<hr />

</body>
</html>
```

game.js

```
window.addEventListener('DOMContentLoaded', main);

function main(){
    var c = document.getElementById('canvas');
    c.width = 640;
    c.height = 480;

    var gl = c.getContext('webgl');

    var v_shader = create_shader('vs');
    var f_shader = create_shader('fs');
    var prg = create_program(v_shader, f_shader);

    var attLocation = new Array();
    attLocation[0] = gl.getAttribLocation(prg, 'position');
    attLocation[1] = gl.getAttribLocation(prg, 'normal');
    attLocation[2] = gl.getAttribLocation(prg, 'color');
    var attStride = new Array();
    attStride[0] = 3;
    attStride[1] = 3;
    attStride[2] = 4;

    var grounddata = myGround();
    var ground_position = grounddata[0];
    var ground_normal = grounddata[1];
    var ground_color = grounddata[2];
    var ground_index = grounddata[3];
    var ground_pos = create_vbo(ground_position);
    var ground_nor = create_vbo(ground_normal);
    var ground_col = create_vbo(ground_color);
    var ground_ibo = create_ibo(ground_index);

    var RADIUS = 0.5;

    var selfdata = mySelf();
    var self_position = selfdata[0];
    var self_normal = selfdata[1];
    var self_color = selfdata[2];
    var self_index = selfdata[3];
    var self_pos = create_vbo(self_position);
    var self_nor = create_vbo(self_normal);
    var self_col = create_vbo(self_color);
    var self_ibo = create_ibo(self_index);

    var itemdata = myItem();
    var item_position = itemdata[0];
    var item_normal = itemdata[1];
    var item_color = itemdata[2];
    var item_index = itemdata[3];
    var item_pos = create_vbo(item_position);
    var item_nor = create_vbo(item_normal);
    var item_col = create_vbo(item_color);
    var item_ibo = create_ibo(item_index);

    var enemydata = myEnemy();
    var enemy_position = enemydata[0];
    var enemy_normal = enemydata[1];
    var enemy_color = enemydata[2];
    var enemy_index = enemydata[3];
    var enemy_pos = create_vbo(enemy_position);
    var enemy_nor = create_vbo(enemy_normal);
    var enemy_col = create_vbo(enemy_color);
    var enemy_ibo = create_ibo(enemy_index);

    var uniLocation = new Array();
    uniLocation[0] = gl.getUniformLocation(prg, 'mvpMatrix');
    uniLocation[1] = gl.getUniformLocation(prg, 'invMatrix');
    uniLocation[2] = gl.getUniformLocation(prg, 'lightDirection');
```

game.js

```
var m = new matIV();

var mMatrix = m.identity(m.create());
var vMatrix = m.identity(m.create());
var pMatrix = m.identity(m.create());
var tmpMatrix = m.identity(m.create());
var mvpMatrix = m.identity(m.create());
var invMatrix = m.identity(m.create());

m.lookAt([0.0, -10.0, 10.0], [0, 0, 0], [0, 0, 1], vMatrix);
m.perspective(60, c.width / c.height, 0.1, 100, pMatrix);
m.multiply(pMatrix, vMatrix, tmpMatrix);

var lightDirection = [0, 0, 1];

gl.enable(gl.DEPTH_TEST);
gl.depthFunc(gl.LEQUAL);
gl.enable(gl.CULL_FACE);

document.onkeydown = handleKeyDown;

var self = [-3, 0, RADIUS];
var item = [3, 0, RADIUS];
var enemy = [0, 0, RADIUS];
var enemy_v = [0,0.1,0];

var R2 = 2 * RADIUS * 2 * RADIUS;
var speedscale = 1.0;
var then = 0;
requestAnimationFrame(drawScene);

function drawScene(now) {
    now *= 0.001;
    var deltaTime = now - then;
    then = now;

     gl.clearColor(0.0, 0.0, 0.0, 1.0);
     gl.clearDepth(1.0);
     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    var timestep=speedscale * deltaTime;
    if(enemy_v[1] > 0) enemy_v[1] = timestep;
    else enemy_v[1] = -timestep;
     enemy[1] += enemy_v[1];
     if(enemy[1] > 5.0) {
        enemy[1] = 5.0;
        enemy_v[1] = -enemy_v[1];
    }
    if(enemy[1] < -5.0) {
        enemy[1] = -5.0;
        enemy_v[1] = -enemy_v[1];
    }

     var dx, dy, dz;
     dx = self[0] - enemy[0];
     dy = self[1] - enemy[1];
     dz = self[2] - enemy[2];
     if(dx * dx + dy * dy + dz * dz < R2) {
        alert("game over");
        self = [-3, 0, RADIUS];
    }
     dx = self[0] - item[0];
     dy = self[1] - item[1];
     dz = self[2] - item[2];
     if(dx * dx + dy * dy + dz * dz < R2) {
        alert("game clear");
        self = [-3, 0, RADIUS];
    }
```

game.js

```
    gl.uniform3fv(uniLocation[2], lightDirection);

    m.identity(mMatrix);
    m.multiply(tmpMatrix, mMatrix, mvpMatrix);
    m.inverse(mMatrix, invMatrix);
    gl.uniformMatrix4fv(uniLocation[0], false, mvpMatrix);
    gl.uniformMatrix4fv(uniLocation[1], false, invMatrix);

    set_attribute([ground_pos, ground_nor, ground_col], attLocation, attStride);
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, ground_ibo);
    gl.drawElements(gl.TRIANGLES, ground_index.length, gl.UNSIGNED_SHORT, 0);
    gl.bindBuffer(gl.ARRAY_BUFFER, null);
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, null);

    m.identity(mMatrix);
    m.translate(mMatrix, self, mMatrix);
    m.multiply(tmpMatrix, mMatrix, mvpMatrix);
    m.inverse(mMatrix, invMatrix);
    gl.uniformMatrix4fv(uniLocation[0], false, mvpMatrix);
    gl.uniformMatrix4fv(uniLocation[1], false, invMatrix);

    set_attribute([self_pos, self_nor, self_col], attLocation, attStride);
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, self_ibo);
    gl.drawElements(gl.TRIANGLES, self_index.length, gl.UNSIGNED_SHORT, 0);
    gl.bindBuffer(gl.ARRAY_BUFFER, null);
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, null);

    m.identity(mMatrix);
    m.translate(mMatrix, item, mMatrix);
    m.multiply(tmpMatrix, mMatrix, mvpMatrix);
    m.inverse(mMatrix, invMatrix);
    gl.uniformMatrix4fv(uniLocation[0], false, mvpMatrix);
    gl.uniformMatrix4fv(uniLocation[1], false, invMatrix);

    set_attribute([item_pos, item_nor, item_col], attLocation, attStride);
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, item_ibo);
    gl.drawElements(gl.TRIANGLES, item_index.length, gl.UNSIGNED_SHORT, 0);
    gl.bindBuffer(gl.ARRAY_BUFFER, null);
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, null);

    m.identity(mMatrix);
    m.translate(mMatrix, enemy, mMatrix);
    m.multiply(tmpMatrix, mMatrix, mvpMatrix);
    m.inverse(mMatrix, invMatrix);
    gl.uniformMatrix4fv(uniLocation[0], false, mvpMatrix);
    gl.uniformMatrix4fv(uniLocation[1], false, invMatrix);

    set_attribute([enemy_pos, enemy_nor, enemy_col], attLocation, attStride);
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, enemy_ibo);
    gl.drawElements(gl.TRIANGLES, enemy_index.length, gl.UNSIGNED_SHORT, 0);
    gl.bindBuffer(gl.ARRAY_BUFFER, null);
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, null);

    gl.flush();

    requestAnimationFrame(drawScene);
}
```

game.js

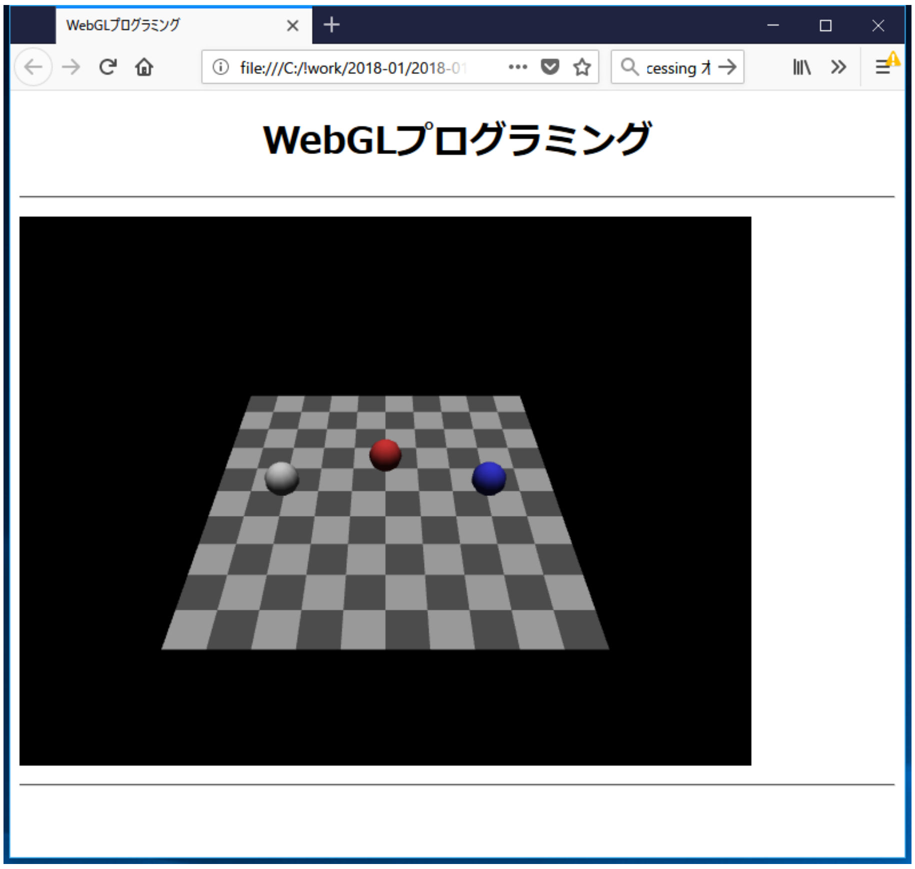
```
function create_shader(id){
    var shader;

    var scriptElement = document.getElementById(id);
    if(!scriptElement){return;}

    switch(scriptElement.type){
        case 'x-shader/x-vertex':
            shader = gl.createShader(gl.VERTEX_SHADER);
            break;
        case 'x-shader/x-fragment':
            shader = gl.createShader(gl.FRAGMENT_SHADER);
            break;
        default :
            return;
    }

    gl.shaderSource(shader, scriptElement.text);
    gl.compileShader(shader);
    if(gl.getShaderParameter(shader, gl.COMPILE_STATUS)){
        return shader;
    }else{
        alert(gl.getShaderInfoLog(shader));
    }
}

function create_program(vs, fs){
    var program = gl.createProgram();
    gl.attachShader(program, vs);
    gl.attachShader(program, fs);
    gl.linkProgram(program);
    if(gl.getProgramParameter(program, gl.LINK_STATUS)){
        gl.useProgram(program);
        return program;
    }else{
        alert(gl.getProgramInfoLog(program));
    }
}

function create_vbo(data){
    var vbo = gl.createBuffer();
    gl.bindBuffer(gl.ARRAY_BUFFER, vbo);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(data), gl.STATIC_DRAW);
    gl.bindBuffer(gl.ARRAY_BUFFER, null);
    return vbo;
}

function set_attribute(vbo, attL, attS){
    for(var i in vbo){
        gl.bindBuffer(gl.ARRAY_BUFFER, vbo[i]);
        gl.enableVertexAttribArray(attL[i]);
        gl.vertexAttribPointer(attL[i], attS[i], gl.FLOAT, false, 0, 0);
    }
}

function create_ibo(data){
    var ibo = gl.createBuffer();
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, ibo);
    gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new Int16Array(data), gl.STATIC_DRAW);
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, null);
    return ibo;
}
```

game.js

```
    function myGround(){
        var pos = new Array(), nor = new Array(),
            col = new Array(), idx = new Array();

        var k = 0;
        for(var j = -5; j < 5; j++){
           for(var i = -5; i < 5; i++) {
               pos.push(i, j, 0);
               pos.push(i+1, j, 0);
               pos.push(i+1, j+1, 0);
               pos.push(i, j+1, 0);
               for(var l = 0; l < 4; l++) {
                   nor.push(0,0,1);
                   if(Math.abs(i+j)%2==0) col.push(0.6,0.6,0.6,1.0);
                   else col.push(0.3, 0.3, 0.3, 1.0);
               }
               idx.push(k, k+1, k+2);
               idx.push(k+2, k+3, k);
               k+=4;
            }
        }
        return [pos, nor, col, idx];
    }

    function mySelf() {
        var pos = new Array(), nor = new Array(),
            col = new Array(), idx = new Array();
        for(var p = 0; p < 10; p++) {
           for(var t = 0; t < 10; t++) {
               var phi = p*36/180.0*Math.PI;
               var theta = t*18/180.0*Math.PI;
               nor.push(Math.sin(theta)*Math.cos(phi), Math.sin(theta)*Math.sin(phi), Math.cos(theta));
               pos.push(RADIUS*Math.sin(theta)*Math.cos(phi), RADIUS*Math.sin(theta)*Math.sin(phi),
RADIUS*Math.cos(theta));
               col.push(0.8,0.8,0.8,1);
               if(t < 9) {
                   idx.push(p*10+t,p*10+t+1,((p+1)%10)*10+t+1);
                   idx.push(((p+1)%10)*10+t+1,((p+1)%10)*10+t,p*10+t);
               }
           }
       }
       return [pos, nor, col, idx];
    }

    function myItem() {
        var pos = new Array(), nor = new Array(),
            col = new Array(), idx = new Array();
        for(var p = 0; p < 10; p++) {
           for(var t = 0; t < 10; t++) {
               var phi = p*36/180.0*Math.PI;
               var theta = t*18/180.0*Math.PI;
               nor.push(Math.sin(theta)*Math.cos(phi), Math.sin(theta)*Math.sin(phi), Math.cos(theta));
               pos.push(RADIUS*Math.sin(theta)*Math.cos(phi), RADIUS*Math.sin(theta)*Math.sin(phi),
RADIUS*Math.cos(theta));
               col.push(0.2,0.2,0.8,1);
               if(t < 9) {
                   idx.push(p*10+t,p*10+t+1,((p+1)%10)*10+t+1);
                   idx.push(((p+1)%10)*10+t+1,((p+1)%10)*10+t,p*10+t);
               }
           }
       }
       return [pos, nor, col, idx];
    }
```

game.js

```
    function myEnemy() {
        var pos = new Array(), nor = new Array(),
            col = new Array(), idx = new Array();
        for(var p = 0; p < 10; p++) {
           for(var t = 0; t < 10; t++) {
               var phi = p*36/180.0*Math.PI;
               var theta = t*18/180.0*Math.PI;
               nor.push(Math.sin(theta)*Math.cos(phi), Math.sin(theta)*Math.sin(phi), Math.cos(theta));
               pos.push(RADIUS*Math.sin(theta)*Math.cos(phi), RADIUS*Math.sin(theta)*Math.sin(phi),
RADIUS*Math.cos(theta));
               col.push(0.8,0.2,0.2,1);
               if(t < 9) {
                   idx.push(p*10+t,p*10+t+1,((p+1)%10)*10+t+1);
                   idx.push(((p+1)%10)*10+t+1,((p+1)%10)*10+t,p*10+t);
               }
           }
        }
        return [pos, nor, col, idx];
    }

    function handleKeyDown(event) {
        var WALKSPEED = 0.1;
        if(event.key == "ArrowLeft") self[0] -= WALKSPEED;
        if(event.key == "ArrowUp") self[1] += WALKSPEED;
        if(event.key == "ArrowRight") self[0] += WALKSPEED;
        if(event.key == "ArrowDown") self[1] -= WALKSPEED;
    }
}
```



## minMatrix.js

```
// ------------------------------------------------------------------------------------------------
// minMatrix.js
// version 0.0.1
// Copyright (c) doxas
// ------------------------------------------------------------------------------------------------

function matIV(){
    this.create = function(){
        return new Float32Array(16);
    };
    this.identity = function(dest){
        dest[0]  = 1; dest[1]  = 0; dest[2]  = 0; dest[3]  = 0;
        dest[4]  = 0; dest[5]  = 1; dest[6]  = 0; dest[7]  = 0;
        dest[8]  = 0; dest[9]  = 0; dest[10] = 1; dest[11] = 0;
        dest[12] = 0; dest[13] = 0; dest[14] = 0; dest[15] = 1;
        return dest;
    };
    this.multiply = function(mat1, mat2, dest){
        var a = mat1[0],  b = mat1[1],  c = mat1[2],  d = mat1[3],
            e = mat1[4],  f = mat1[5],  g = mat1[6],  h = mat1[7],
            i = mat1[8],  j = mat1[9],  k = mat1[10], l = mat1[11],
            m = mat1[12], n = mat1[13], o = mat1[14], p = mat1[15],
            A = mat2[0],  B = mat2[1],  C = mat2[2],  D = mat2[3],
            E = mat2[4],  F = mat2[5],  G = mat2[6],  H = mat2[7],
            I = mat2[8],  J = mat2[9],  K = mat2[10], L = mat2[11],
            M = mat2[12], N = mat2[13], O = mat2[14], P = mat2[15];
        dest[0] = A * a + B * e + C * i + D * m;
        dest[1] = A * b + B * f + C * j + D * n;
        dest[2] = A * c + B * g + C * k + D * o;
        dest[3] = A * d + B * h + C * l + D * p;
        dest[4] = E * a + F * e + G * i + H * m;
        dest[5] = E * b + F * f + G * j + H * n;
        dest[6] = E * c + F * g + G * k + H * o;
        dest[7] = E * d + F * h + G * l + H * p;
        dest[8] = I * a + J * e + K * i + L * m;
        dest[9] = I * b + J * f + K * j + L * n;
        dest[10] = I * c + J * g + K * k + L * o;
        dest[11] = I * d + J * h + K * l + L * p;
        dest[12] = M * a + N * e + O * i + P * m;
        dest[13] = M * b + N * f + O * j + P * n;
        dest[14] = M * c + N * g + O * k + P * o;
        dest[15] = M * d + N * h + O * l + P * p;
        return dest;
    };
    this.scale = function(mat, vec, dest){
        dest[0]  = mat[0]  * vec[0];
        dest[1]  = mat[1]  * vec[0];
        dest[2]  = mat[2]  * vec[0];
        dest[3]  = mat[3]  * vec[0];
        dest[4]  = mat[4]  * vec[1];
        dest[5]  = mat[5]  * vec[1];
        dest[6]  = mat[6]  * vec[1];
        dest[7]  = mat[7]  * vec[1];
        dest[8]  = mat[8]  * vec[2];
        dest[9]  = mat[9]  * vec[2];
        dest[10] = mat[10] * vec[2];
        dest[11] = mat[11] * vec[2];
        dest[12] = mat[12];
        dest[13] = mat[13];
        dest[14] = mat[14];
        dest[15] = mat[15];
        return dest;
    };
    this.translate = function(mat, vec, dest){
        dest[0] = mat[0]; dest[1] = mat[1]; dest[2]  = mat[2];  dest[3]  = mat[3];
        dest[4] = mat[4]; dest[5] = mat[5]; dest[6]  = mat[6];  dest[7]  = mat[7];
        dest[8] = mat[8]; dest[9] = mat[9]; dest[10] = mat[10]; dest[11] = mat[11];
        dest[12] = mat[0] * vec[0] + mat[4] * vec[1] + mat[8]  * vec[2] + mat[12];
        dest[13] = mat[1] * vec[0] + mat[5] * vec[1] + mat[9]  * vec[2] + mat[13];
        dest[14] = mat[2] * vec[0] + mat[6] * vec[1] + mat[10] * vec[2] + mat[14];
        dest[15] = mat[3] * vec[0] + mat[7] * vec[1] + mat[11] * vec[2] + mat[15];
        return dest;
    };
```

```
this.rotate = function(mat, angle, axis, dest){
    var sq = Math.sqrt(axis[0] * axis[0] + axis[1] * axis[1] + axis[2] * axis[2]);
    if(!sq){return null;}
    var a = axis[0], b = axis[1], c = axis[2];
    if(sq != 1){sq = 1 / sq; a *= sq; b *= sq; c *= sq;}
    var d = Math.sin(angle), e = Math.cos(angle), f = 1 - e,
        g = mat[0],  h = mat[1], i = mat[2],  j = mat[3],
        k = mat[4],  l = mat[5], m = mat[6],  n = mat[7],
        o = mat[8],  p = mat[9], q = mat[10], r = mat[11],
        s = a * a * f + e,
        t = b * a * f + c * d,
        u = c * a * f - b * d,
        v = a * b * f - c * d,
        w = b * b * f + e,
        x = c * b * f + a * d,
        y = a * c * f + b * d,
        z = b * c * f - a * d,
        A = c * c * f + e;
    if(angle){
        if(mat != dest){
            dest[12] = mat[12]; dest[13] = mat[13];
            dest[14] = mat[14]; dest[15] = mat[15];
        }
    } else {
        dest = mat;
    }
    dest[0] = g * s + k * t + o * u;
    dest[1] = h * s + l * t + p * u;
    dest[2] = i * s + m * t + q * u;
    dest[3] = j * s + n * t + r * u;
    dest[4] = g * v + k * w + o * x;
    dest[5] = h * v + l * w + p * x;
    dest[6] = i * v + m * w + q * x;
    dest[7] = j * v + n * w + r * x;
    dest[8] = g * y + k * z + o * A;
    dest[9] = h * y + l * z + p * A;
    dest[10] = i * y + m * z + q * A;
    dest[11] = j * y + n * z + r * A;
    return dest;
};
this.lookAt = function(eye, center, up, dest){
    var eyeX    = eye[0],    eyeY    = eye[1],    eyeZ    = eye[2],
        upX     = up[0],     upY     = up[1],     upZ     = up[2],
        centerX = center[0], centerY = center[1], centerZ = center[2];
    if(eyeX == centerX && eyeY == centerY && eyeZ == centerZ){return this.identity(dest);}
    var x0, x1, x2, y0, y1, y2, z0, z1, z2, l;
    z0 = eyeX - center[0]; z1 = eyeY - center[1]; z2 = eyeZ - center[2];
    l = 1 / Math.sqrt(z0 * z0 + z1 * z1 + z2 * z2);
    z0 *= l; z1 *= l; z2 *= l;
    x0 = upY * z2 - upZ * z1;
    x1 = upZ * z0 - upX * z2;
    x2 = upX * z1 - upY * z0;
    l = Math.sqrt(x0 * x0 + x1 * x1 + x2 * x2);
    if(!l){
        x0 = 0; x1 = 0; x2 = 0;
    } else {
        l = 1 / l;
        x0 *= l; x1 *= l; x2 *= l;
    }
    y0 = z1 * x2 - z2 * x1; y1 = z2 * x0 - z0 * x2; y2 = z0 * x1 - z1 * x0;
    l = Math.sqrt(y0 * y0 + y1 * y1 + y2 * y2);
    if(!l){
        y0 = 0; y1 = 0; y2 = 0;
    } else {
        l = 1 / l;
        y0 *= l; y1 *= l; y2 *= l;
    }
```

```
        dest[0] = x0; dest[1] = y0; dest[2]  = z0; dest[3]  = 0;
        dest[4] = x1; dest[5] = y1; dest[6]  = z1; dest[7]  = 0;
        dest[8] = x2; dest[9] = y2; dest[10] = z2; dest[11] = 0;
        dest[12] = -(x0 * eyeX + x1 * eyeY + x2 * eyeZ);
        dest[13] = -(y0 * eyeX + y1 * eyeY + y2 * eyeZ);
        dest[14] = -(z0 * eyeX + z1 * eyeY + z2 * eyeZ);
        dest[15] = 1;
        return dest;
    };
    this.perspective = function(fovy, aspect, near, far, dest){
        var t = near * Math.tan(fovy * Math.PI / 360);
        var r = t * aspect;
        var a = r * 2, b = t * 2, c = far - near;
        dest[0] = near * 2 / a;
        dest[1] = 0;
        dest[2] = 0;
        dest[3] = 0;
        dest[4] = 0;
        dest[5] = near * 2 / b;
        dest[6] = 0;
        dest[7] = 0;
        dest[8] = 0;
        dest[9] = 0;
        dest[10] = -(far + near) / c;
        dest[11] = -1;
        dest[12] = 0;
        dest[13] = 0;
        dest[14] = -(far * near * 2) / c;
        dest[15] = 0;
        return dest;
    };
    this.transpose = function(mat, dest){
        dest[0]  = mat[0];  dest[1]  = mat[4];
        dest[2]  = mat[8];  dest[3]  = mat[12];
        dest[4]  = mat[1];  dest[5]  = mat[5];
        dest[6]  = mat[9];  dest[7]  = mat[13];
        dest[8]  = mat[2];  dest[9]  = mat[6];
        dest[10] = mat[10]; dest[11] = mat[14];
        dest[12] = mat[3];  dest[13] = mat[7];
        dest[14] = mat[11]; dest[15] = mat[15];
        return dest;
    };
    this.inverse = function(mat, dest){
        var a = mat[0],  b = mat[1],  c = mat[2],  d = mat[3],
            e = mat[4],  f = mat[5],  g = mat[6],  h = mat[7],
            i = mat[8],  j = mat[9],  k = mat[10], l = mat[11],
            m = mat[12], n = mat[13], o = mat[14], p = mat[15],
            q = a * f - b * e, r = a * g - c * e,
            s = a * h - d * e, t = b * g - c * f,
            u = b * h - d * f, v = c * h - d * g,
            w = i * n - j * m, x = i * o - k * m,
            y = i * p - l * m, z = j * o - k * n,
            A = j * p - l * n, B = k * p - l * o,
            ivd = 1 / (q * B - r * A + s * z + t * y - u * x + v * w);
        dest[0]  = ( f * B - g * A + h * z) * ivd;
        dest[1]  = (-b * B + c * A - d * z) * ivd;
        dest[2]  = ( n * v - o * u + p * t) * ivd;
        dest[3]  = (-j * v + k * u - l * t) * ivd;
        dest[4]  = (-e * B + g * y - h * x) * ivd;
        dest[5]  = ( a * B - c * y + d * x) * ivd;
        dest[6]  = (-m * v + o * s - p * r) * ivd;
        dest[7]  = ( i * v - k * s + l * r) * ivd;
        dest[8]  = ( e * A - f * y + h * w) * ivd;
        dest[9]  = (-a * A + b * y - d * w) * ivd;
        dest[10] = ( m * u - n * s + p * q) * ivd;
        dest[11] = (-i * u + j * s - l * q) * ivd;
        dest[12] = (-e * z + f * x - g * w) * ivd;
        dest[13] = ( a * z - b * x + c * w) * ivd;
        dest[14] = (-m * t + n * r - o * q) * ivd;
        dest[15] = ( i * t - j * r + k * q) * ivd;
        return dest;
    };
}
```