```
// global variables
var canvas=null;
var gl=null; // webgl context
var bFullscreen=false;
var canvas_original_width;
var canvas_original_height;

const WebGLMacros= // when whole 'WebGLMacros' is 'const', all inside it are automatically 'const'
{
VDG_ATTRIBUTE_VERTEX:0,
VDG_ATTRIBUTE_COLOR:1,
VDG_ATTRIBUTE_NORMAL:2,
VDG_ATTRIBUTE_TEXTURE0:3,
};

var vertexShaderObject;
var fragmentShaderObject;
var shaderProgramObject;

var vao_pyramid;
var vao_cube;
var vbo_position;
var vbo_texture;
var mvpUniform;

var perspectiveProjectionMatrix;

var pyramid_texture=0;
var uniform_texture0_sampler;

var anglePyramid=0.0;
var cube_texture=0;
var angleCube=0.0;

// To start animation : To have requestAnimationFrame() to be called "cross-browser" compatible
var requestAnimationFrame =
window.requestAnimationFrame ||
window.webkitRequestAnimationFrame ||
window.mozRequestAnimationFrame ||
window.oRequestAnimationFrame ||
window.msRequestAnimationFrame;

// To stop animation : To have cancelAnimationFrame() to be called "cross-browser" compatible
var cancelAnimationFrame =
window.cancelAnimationFrame ||
window.webkitCancelRequestAnimationFrame || window.webkitCancelAnimationFrame ||
window.mozCancelRequestAnimationFrame || window.mozCancelAnimationFrame ||
window.oCancelRequestAnimationFrame || window.oCancelAnimationFrame ||
window.msCancelRequestAnimationFrame || window.msCancelAnimationFrame;
```

```
50
51  // onload function
52  function main()
53  {
54      // get <canvas> element
55      canvas = document.getElementById("AMC");
56      if(!canvas)
57          console.log("Obtaining Canvas Failed\n");
58      else
59          console.log("Obtaining Canvas Succeeded\n");
60      canvas_original_width=canvas.width;
61      canvas_original_height=canvas.height;
62
63      // register keyboard's keydown event handler
64      window.addEventListener("keydown", keyDown, false);
65      window.addEventListener("click", mouseDown, false);
66      window.addEventListener("resize", resize, false);
67
68      // initialize WebGL
69      init();
70
71      // start drawing here as warming-up
72      resize();
73      draw();
74  }
75
76  function toggleFullScreen()
77  {
78      // code
79      var fullscreen_element =
80      document.fullscreenElement ||
81      document.webkitFullscreenElement ||
82      document.mozFullScreenElement ||
83      document.msFullscreenElement ||
84      null;
85
86      // if not fullscreen
87      if(fullscreen_element==null)
88      {
89          if(canvas.requestFullscreen)
90              canvas.requestFullscreen();
91          else if(canvas.mozRequestFullScreen)
92              canvas.mozRequestFullScreen();
93          else if(canvas.webkitRequestFullscreen)
94              canvas.webkitRequestFullscreen();
95          else if(canvas.msRequestFullscreen)
96              canvas.msRequestFullscreen();
97          bFullscreen=true;
98      }
99      else // if already fullscreen
100     {
101         if(document.exitFullscreen)
```

```
            document.exitFullscreen();
        else if(document.mozCancelFullScreen)
            document.mozCancelFullScreen();
        else if(document.webkitExitFullscreen)
            document.webkitExitFullscreen();
        else if(document.msExitFullscreen)
            document.msExitFullscreen();
        bFullscreen=false;
    }
}

function init()
{
    // code
    // get WebGL 2.0 context
    gl = canvas.getContext("webgl2");
    if(gl==null) // failed to get context
    {
        console.log("Failed to get the rendering context for WebGL");
        return;
    }
    gl.viewportWidth = canvas.width;
    gl.viewportHeight = canvas.height;

    // vertex shader
    var vertexShaderSourceCode=
    "#version 300 es"+
    "\n"+
    "in vec4 vPosition;"+
    "in vec2 vTexture0_Coord;"+
    "out vec2 out_texture0_coord;"+
    "uniform mat4 u_mvp_matrix;"+
    "void main(void)"+
    "{"+
    "gl_Position = u_mvp_matrix * vPosition;"+
    "out_texture0_coord = vTexture0_Coord;"+
    "}";
    vertexShaderObject=gl.createShader(gl.VERTEX_SHADER);
    gl.shaderSource(vertexShaderObject,vertexShaderSourceCode);
    gl.compileShader(vertexShaderObject);
    if(gl.getShaderParameter(vertexShaderObject,gl.COMPILE_STATUS)==false)
    {
        var error=gl.getShaderInfoLog(vertexShaderObject);
        if(error.length > 0)
        {
            alert(error);
            uninitialize();
        }
    }

    // fragment shader
    var fragmentShaderSourceCode=
```

```
154     "#version 300 es"+
155     "\n"+
156     "precision highp float;"+
157     "in vec2 out_texture0_coord;"+
158     "uniform highp sampler2D u_texture0_sampler;"+
159     "out vec4 FragColor;"+
160     "void main(void)"+
161     "{"+
162     "FragColor = texture(u_texture0_sampler, out_texture0_coord);"+
163     "}"
164     fragmentShaderObject=gl.createShader(gl.FRAGMENT_SHADER);
165     gl.shaderSource(fragmentShaderObject,fragmentShaderSourceCode);
166     gl.compileShader(fragmentShaderObject);
167     if(gl.getShaderParameter(fragmentShaderObject,gl.COMPILE_STATUS)==false)
168     {
169         var error=gl.getShaderInfoLog(fragmentShaderObject);
170         if(error.length > 0)
171         {
172             alert(error);
173             uninitialize();
174         }
175     }
176
177     // shader program
178     shaderProgramObject=gl.createProgram();
179     gl.attachShader(shaderProgramObject,vertexShaderObject);
180     gl.attachShader(shaderProgramObject,fragmentShaderObject);
181
182     // pre-link binding of shader program object with vertex shader attributes
183     gl.bindAttribLocation
          (shaderProgramObject,WebGLMacros.VDG_ATTRIBUTE_VERTEX,"vPosition");
184     gl.bindAttribLocation
          (shaderProgramObject,WebGLMacros.VDG_ATTRIBUTE_TEXTURE0,"vTexture0_Coord");
185
186     // linking
187     gl.linkProgram(shaderProgramObject);
188     if (!gl.getProgramParameter(shaderProgramObject, gl.LINK_STATUS))
189     {
190         var error=gl.getProgramInfoLog(shaderProgramObject);
191         if(error.length > 0)
192         {
193             alert(error);
194             uninitialize();
195         }
196     }
197
198     // Load pyramid Textures
199     pyramid_texture = gl.createTexture();
200     pyramid_texture.image = new Image();
201     pyramid_texture.image.src="stone.png";
202     pyramid_texture.image.onload = function ()
203     {
```

```
204         gl.bindTexture(gl.TEXTURE_2D, pyramid_texture);
205         gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);
206         gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE,
              pyramid_texture.image);
207         gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
208         gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.NEAREST);
209         gl.bindTexture(gl.TEXTURE_2D, null);
210     }
211
212     // Load cube Textures
213     cube_texture = gl.createTexture();
214     cube_texture.image = new Image();
215     cube_texture.image.src="Vijay_Kundali.png";
216     cube_texture.image.onload = function ()
217     {
218         gl.bindTexture(gl.TEXTURE_2D, cube_texture);
219         gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);
220         gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE,
              cube_texture.image);
221         gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.NEAREST);
222         gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.NEAREST);
223         gl.bindTexture(gl.TEXTURE_2D, null);
224     }
225
226     // get MVP uniform location
227     mvpUniform=gl.getUniformLocation(shaderProgramObject,"u_mvp_matrix");
228     // get texture0_sampler uniform location
229     uniform_texture0_sampler=gl.getUniformLocation
          (shaderProgramObject,"u_texture0_sampler");
230
231     // *** vertices, colors, shader attribs, vbo, vao initializations ***
232     var pyramidVertices=new Float32Array([
233                                          0.0, 1.0, 0.0,    // front-top
234                                          -1.0, -1.0, 1.0,  // front-left
235                                          1.0, -1.0, 1.0,   // front-right
236
237                                          0.0, 1.0, 0.0,    // right-top
238                                          1.0, -1.0, 1.0,   // right-left
239                                          1.0, -1.0, -1.0,  // right-right
240
241                                          0.0, 1.0, 0.0,    // back-top
242                                          1.0, -1.0, -1.0,  // back-left
243                                          -1.0, -1.0, -1.0, // back-right
244
245                                          0.0, 1.0, 0.0,    // left-top
246                                          -1.0, -1.0, -1.0, // left-left
247                                          -1.0, -1.0, 1.0   // left-right
248                                         ]);
249
250     var pyramidTexcoords=new Float32Array([
251                                           0.5, 1.0, // front-top
252                                           0.0, 0.0, // front-left
```

```
                                                1.0, 0.0, // front-right

                                                0.5, 1.0, // right-top
                                                1.0, 0.0, // right-left
                                                0.0, 0.0, // right-right

                                                0.5, 1.0, // back-top
                                                1.0, 0.0, // back-left
                                                0.0, 0.0, // back-right

                                                0.5, 1.0, // left-top
                                                0.0, 0.0, // left-left
                                                1.0, 0.0, // left-right
                                                ]);

    var cubeVertices=new Float32Array([
                                            // top surface
                                            1.0, 1.0,-1.0,  // top-right of top
                                            -1.0, 1.0,-1.0, // top-left of top
                                            -1.0, 1.0, 1.0, // bottom-left of top
                                            1.0, 1.0, 1.0,  // bottom-right of top

                                            // bottom surface
                                            1.0,-1.0, 1.0,  // top-right of bottom
                                            -1.0,-1.0, 1.0, // top-left of bottom
                                            -1.0,-1.0,-1.0, // bottom-left of bottom
                                            1.0,-1.0,-1.0,  // bottom-right of bottom

                                            // front surface
                                            1.0, 1.0, 1.0,  // top-right of front
                                            -1.0, 1.0, 1.0, // top-left of front
                                            -1.0,-1.0, 1.0, // bottom-left of front
                                            1.0,-1.0, 1.0,  // bottom-right of front

                                            // back surface
                                            1.0,-1.0,-1.0,  // top-right of back
                                            -1.0,-1.0,-1.0, // top-left of back
                                            -1.0, 1.0,-1.0, // bottom-left of back
                                            1.0, 1.0,-1.0,  // bottom-right of back

                                            // left surface
                                            -1.0, 1.0, 1.0, // top-right of left
                                            -1.0, 1.0,-1.0, // top-left of left
                                            -1.0,-1.0,-1.0, // bottom-left of left
                                            -1.0,-1.0, 1.0, // bottom-right of left

                                            // right surface
                                            1.0, 1.0,-1.0,  // top-right of right
                                            1.0, 1.0, 1.0,  // top-left of right
                                            1.0,-1.0, 1.0,  // bottom-left of right
                                            1.0,-1.0,-1.0,  // bottom-right of right
                                            ]);
```

```
305
306     // If above -1.0 Or +1.0 Values Make Cube Much Larger Than Pyramid,
307     // then follow the code in following loop which will convertt all 1s And -1s
          to -0.75 or +0.75
308     for(var i=0;i<72;i++)
309     {
310         if(cubeVertices[i]<0.0)
311             cubeVertices[i]=cubeVertices[i]+0.25;
312         else if(cubeVertices[i]>0.0)
313             cubeVertices[i]=cubeVertices[i]-0.25;
314         else
315             cubeVertices[i]=cubeVertices[i]; // no change
316     }
317
318     var cubeTexcoords=new Float32Array([
319                                         0.0,0.0,
320                                         1.0,0.0,
321                                         1.0,1.0,
322                                         0.0,1.0,
323
324                                         0.0,0.0,
325                                         1.0,0.0,
326                                         1.0,1.0,
327                                         0.0,1.0,
328
329                                         0.0,0.0,
330                                         1.0,0.0,
331                                         1.0,1.0,
332                                         0.0,1.0,
333
334                                         0.0,0.0,
335                                         1.0,0.0,
336                                         1.0,1.0,
337                                         0.0,1.0,
338
339                                         0.0,0.0,
340                                         1.0,0.0,
341                                         1.0,1.0,
342                                         0.0,1.0,
343
344                                         0.0,0.0,
345                                         1.0,0.0,
346                                         1.0,1.0,
347                                         0.0,1.0,
348                                         ]);
349
350     // pyramid code
351     vao_pyramid=gl.createVertexArray();
352     gl.bindVertexArray(vao_pyramid);
353
354     vbo_position = gl.createBuffer();
355     gl.bindBuffer(gl.ARRAY_BUFFER,vbo_position);
```

```
356     gl.bufferData(gl.ARRAY_BUFFER,pyramidVertices,gl.STATIC_DRAW);
357     gl.vertexAttribPointer(WebGLMacros.VDG_ATTRIBUTE_VERTEX,
358                            3, // 3 is for X,Y,Z co-ordinates in our
                       pyramidVertices array
359                            gl.FLOAT,
360                            false,0,0);
361     gl.enableVertexAttribArray(WebGLMacros.VDG_ATTRIBUTE_VERTEX);
362     gl.bindBuffer(gl.ARRAY_BUFFER,null);
363
364     vbo_texture = gl.createBuffer();
365     gl.bindBuffer(gl.ARRAY_BUFFER,vbo_texture);
366     gl.bufferData(gl.ARRAY_BUFFER,pyramidTexcoords,gl.STATIC_DRAW);
367     gl.vertexAttribPointer(WebGLMacros.VDG_ATTRIBUTE_TEXTURE0,
368                            2, // 2 is for S and T co-ordinates in our
                       pyramidTexcoords array
369                            gl.FLOAT,
370                            false,0,0);
371     gl.enableVertexAttribArray(WebGLMacros.VDG_ATTRIBUTE_TEXTURE0);
372     gl.bindBuffer(gl.ARRAY_BUFFER,null);
373
374     gl.bindVertexArray(null);
375
376     // cube code
377     vao_cube=gl.createVertexArray();
378     gl.bindVertexArray(vao_cube);
379
380     vbo_position = gl.createBuffer();
381     gl.bindBuffer(gl.ARRAY_BUFFER,vbo_position);
382     gl.bufferData(gl.ARRAY_BUFFER,cubeVertices,gl.STATIC_DRAW);
383     gl.vertexAttribPointer(WebGLMacros.VDG_ATTRIBUTE_VERTEX,
384                            3, // 3 is for X,Y,Z co-ordinates in our
                       triangleVertices array
385                            gl.FLOAT,
386                            false,0,0);
387     gl.enableVertexAttribArray(WebGLMacros.VDG_ATTRIBUTE_VERTEX);
388     gl.bindBuffer(gl.ARRAY_BUFFER,null);
389
390     vbo_texture = gl.createBuffer();
391     gl.bindBuffer(gl.ARRAY_BUFFER,vbo_texture);
392     gl.bufferData(gl.ARRAY_BUFFER,cubeTexcoords,gl.STATIC_DRAW);
393     gl.vertexAttribPointer(WebGLMacros.VDG_ATTRIBUTE_TEXTURE0,
394                            2, // 2 is for S and T co-ordinates in our
                       pyramidTexcoords array
395                            gl.FLOAT,
396                            false,0,0);
397     gl.enableVertexAttribArray(WebGLMacros.VDG_ATTRIBUTE_TEXTURE0);
398     gl.bindBuffer(gl.ARRAY_BUFFER,null);
399
400     gl.bindVertexArray(null);
401
402     // set clear color
403     gl.clearColor(0.0, 0.0, 0.0, 1.0); // black
```

```
404
405     // Depth test will always be enabled
406     gl.enable(gl.DEPTH_TEST);
407
408     // We will always cull back faces for better performance
409     gl.enable(gl.CULL_FACE);
410
411     // initialize projection matrix
412     perspectiveProjectionMatrix=mat4.create();
413 }
414
415 function resize()
416 {
417     // code
418     if(bFullscreen==true)
419     {
420         canvas.width=window.innerWidth;
421         canvas.height=window.innerHeight;
422     }
423     else
424     {
425         canvas.width=canvas_original_width;
426         canvas.height=canvas_original_height;
427     }
428
429     // set the viewport to match
430     gl.viewport(0, 0, canvas.width, canvas.height);
431
432     mat4.perspective(perspectiveProjectionMatrix, 45.0, parseFloat(canvas.width)/
      parseFloat(canvas.height), 0.1, 100.0);
433 }
434
435 function draw()
436 {
437     // code
438     gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
439
440     gl.useProgram(shaderProgramObject);
441
442     var modelViewMatrix=mat4.create(); // itself creates identity matrix
443     var modelViewProjectionMatrix=mat4.create(); // itself creates identity
      matrix
444
445     // pyramid
446     mat4.translate(modelViewMatrix, modelViewMatrix, [-1.5,0.0,-5.0]);
447
448     mat4.rotateY(modelViewMatrix ,modelViewMatrix, degToRad(anglePyramid));
449
450     mat4.multiply
      (modelViewProjectionMatrix,perspectiveProjectionMatrix,modelViewMatrix);
451     gl.uniformMatrix4fv(mvpUniform,false,modelViewProjectionMatrix);
452
```

```
453     // bind with texture
454     gl.bindTexture(gl.TEXTURE_2D,pyramid_texture);
455     gl.uniform1i(uniform_texture0_sampler, 0);
456
457     gl.bindVertexArray(vao_pyramid);
458
459     gl.drawArrays(gl.TRIANGLES,0,12); // 3 (each with its x,y,z ) vertices for 4
          faces of pyramid
460
461     gl.bindVertexArray(null);
462
463     // cube
464     mat4.identity(modelViewMatrix); // reset to identity matrix
465     mat4.identity(modelViewProjectionMatrix); // reset to identity matrix
466
467     mat4.translate(modelViewMatrix, modelViewMatrix, [1.5,0.0,-5.0]);
468
469     mat4.rotateX(modelViewMatrix ,modelViewMatrix, degToRad(angleCube));
470     mat4.rotateY(modelViewMatrix ,modelViewMatrix, degToRad(angleCube));
471     mat4.rotateZ(modelViewMatrix ,modelViewMatrix, degToRad(angleCube));
472
473     mat4.multiply
          (modelViewProjectionMatrix,perspectiveProjectionMatrix,modelViewMatrix);
474     gl.uniformMatrix4fv(mvpUniform,false,modelViewProjectionMatrix);
475
476     // bind with texture
477     gl.bindTexture(gl.TEXTURE_2D,cube_texture);
478     gl.uniform1i(uniform_texture0_sampler, 0);
479
480     gl.bindVertexArray(vao_cube);
481
482     // *** draw, either by glDrawTriangles() or glDrawArrays() or glDrawElements
          ()
483     // actually 2 triangles make 1 cube, so there should be 6 vertices,
484     // but as 2 tringles while making square meet each other at diagonal,
485     // 2 of 6 vertices are common to both triangles, and hence 6-2=4
486     gl.drawArrays(gl.TRIANGLE_FAN,0,4);
487     gl.drawArrays(gl.TRIANGLE_FAN,4,4);
488     gl.drawArrays(gl.TRIANGLE_FAN,8,4);
489     gl.drawArrays(gl.TRIANGLE_FAN,12,4);
490     gl.drawArrays(gl.TRIANGLE_FAN,16,4);
491     gl.drawArrays(gl.TRIANGLE_FAN,20,4);
492
493     gl.bindVertexArray(null);
494
495     gl.useProgram(null);
496
497     anglePyramid=anglePyramid+2.0;
498     if(anglePyramid>=360.0)
499         anglePyramid=anglePyramid-360.0;
500
501     angleCube=angleCube+2.0;
```

```
502     if(angleCube>=360.0)
503         angleCube=angleCube-360.0;
504
505     // animation loop
506     requestAnimationFrame(draw, canvas);
507 }
508
509 function uninitialize()
510 {
511     // code
512     if(pyramid_texture)
513     {
514         gl.deleteTexture(pyramid_texture);
515         pyramid_texture=0;
516     }
517
518     if(cube_texture)
519     {
520         gl.deleteTexture(cube_texture);
521         cube_texture=0;
522     }
523
524     if(vao_pyramid)
525     {
526         gl.deleteVertexArray(vao_pyramid);
527         vao_pyramid=null;
528     }
529
530     if(vao_cube)
531     {
532         gl.deleteVertexArray(vao_cube);
533         vao_cube=null;
534     }
535
536     if(vbo_texture)
537     {
538         gl.deleteBuffer(vbo_texture);
539         vbo_texture=null;
540     }
541
542     if(vbo_position)
543     {
544         gl.deleteBuffer(vbo_position);
545         vbo_position=null;
546     }
547
548     if(shaderProgramObject)
549     {
550         if(fragmentShaderObject)
551         {
552             gl.detachShader(shaderProgramObject,fragmentShaderObject);
553             gl.deleteShader(fragmentShaderObject);
```

```
554             fragmentShaderObject=null;
555         }
556
557         if(vertexShaderObject)
558         {
559             gl.detachShader(shaderProgramObject,vertexShaderObject);
560             gl.deleteShader(vertexShaderObject);
561             vertexShaderObject=null;
562         }
563
564         gl.deleteProgram(shaderProgramObject);
565         shaderProgramObject=null;
566     }
567 }
568
569 function keyDown(event)
570 {
571     // code
572     switch(event.keyCode)
573     {
574         case 27: // Escape
575             // uninitialize
576             uninitialize();
577             // close our application's tab
578             window.close(); // may not work in Firefox but works in Safari and
                chrome
579             break;
580         case 70: // for 'F' or 'f'
581             toggleFullScreen();
582             break;
583     }
584 }
585
586 function mouseDown()
587 {
588     // code
589 }
590
591 function degToRad(degrees)
592 {
593     // code
594     return(degrees * Math.PI / 180);
595 }
596
```