Lab 08- Lab 09
Geometric Transformations
Working in 2D & viewing in 3D

Discussion is based on F.S. Hill Chapter 05,06

BSCS – 514 Computer Graphics
Course Supervisor Dr. Humera Tariq

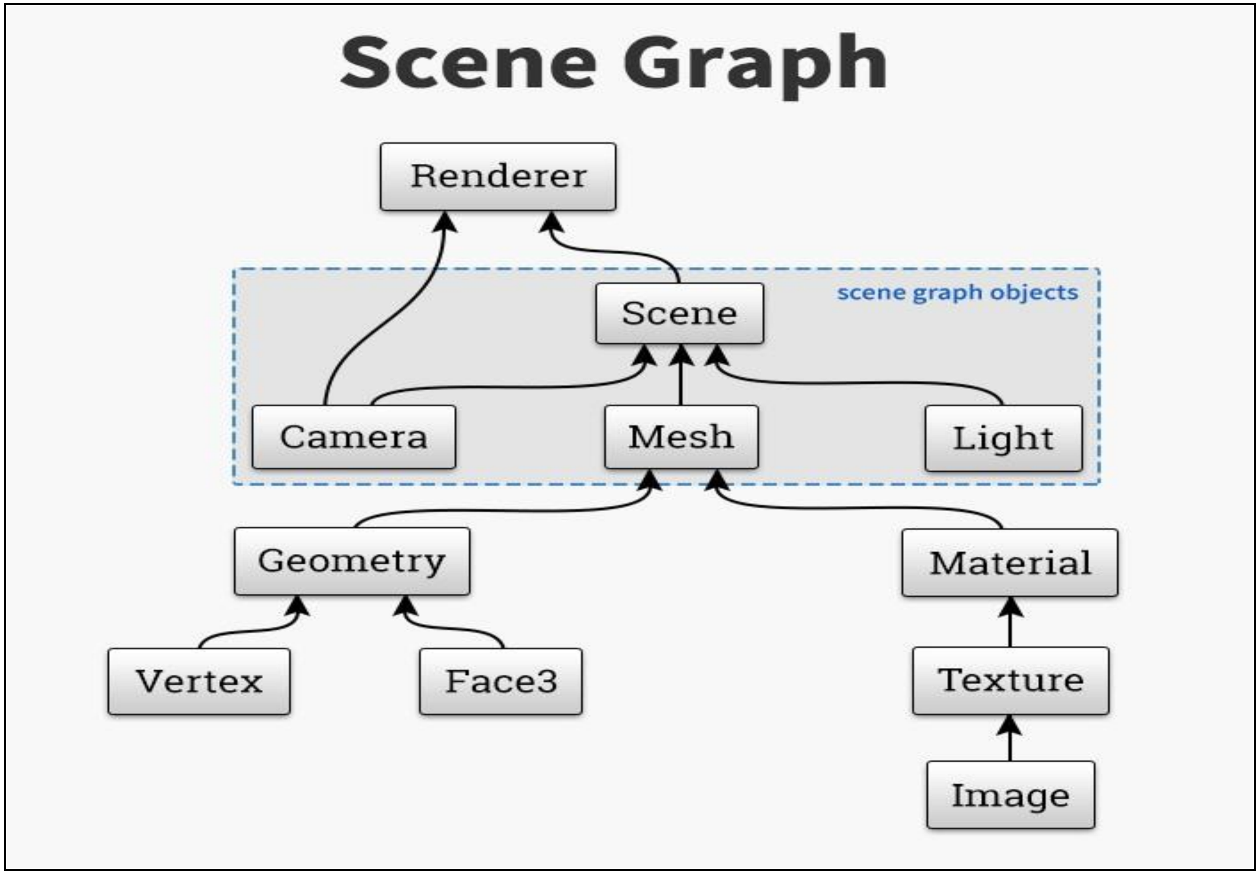
Contents

- Scene Graph Concept
- npm commands to run node.js project
- Opengl/glut code vs. JS coding
- Code to setup camera
- Code to model Triangle, Cube, Grid , Pyramid
- Modeling Data Structure: Vertex List, Indexed Face List
- Code to apply object Transformations
- Solving Book Exercise on Transformations using JS Code
- Switching from 2D to 3D viewing and vice versa

Reading notes for three.js

http://math.hws.edu/eck/cs424/notes2013/15_Threejs_Intro.html

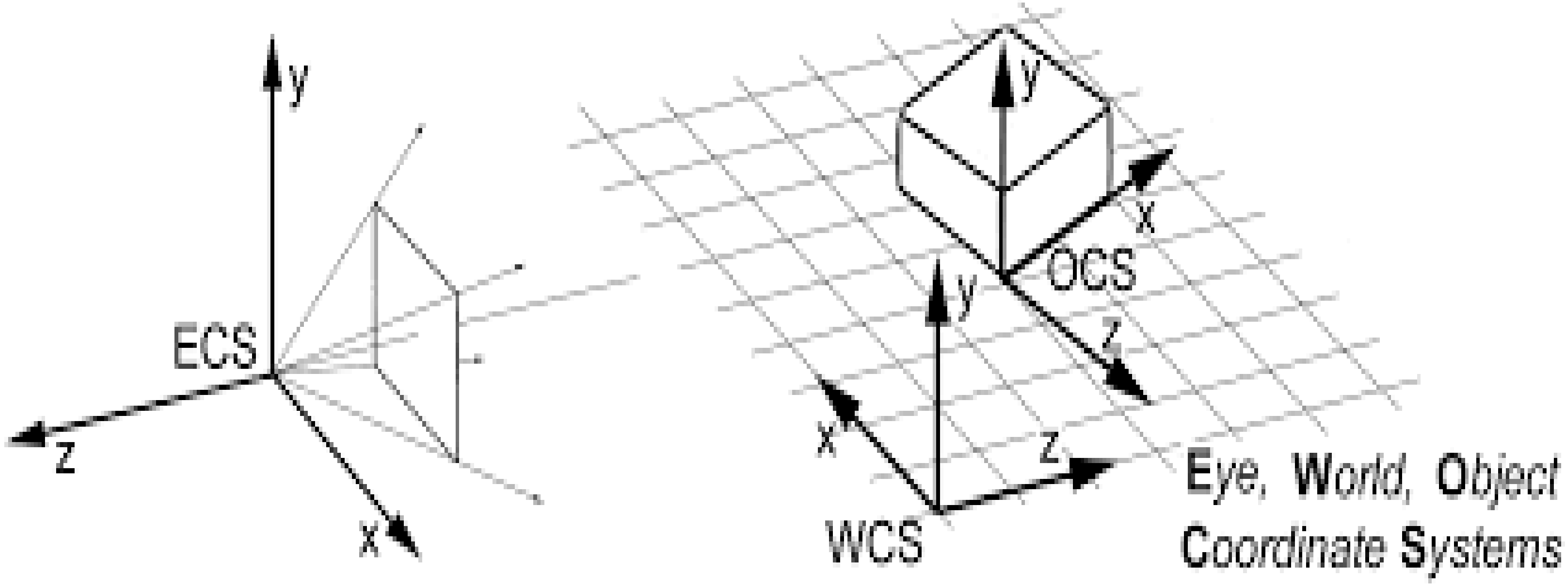
http://math.hws.edu/graphicsbook/c5/s1.html



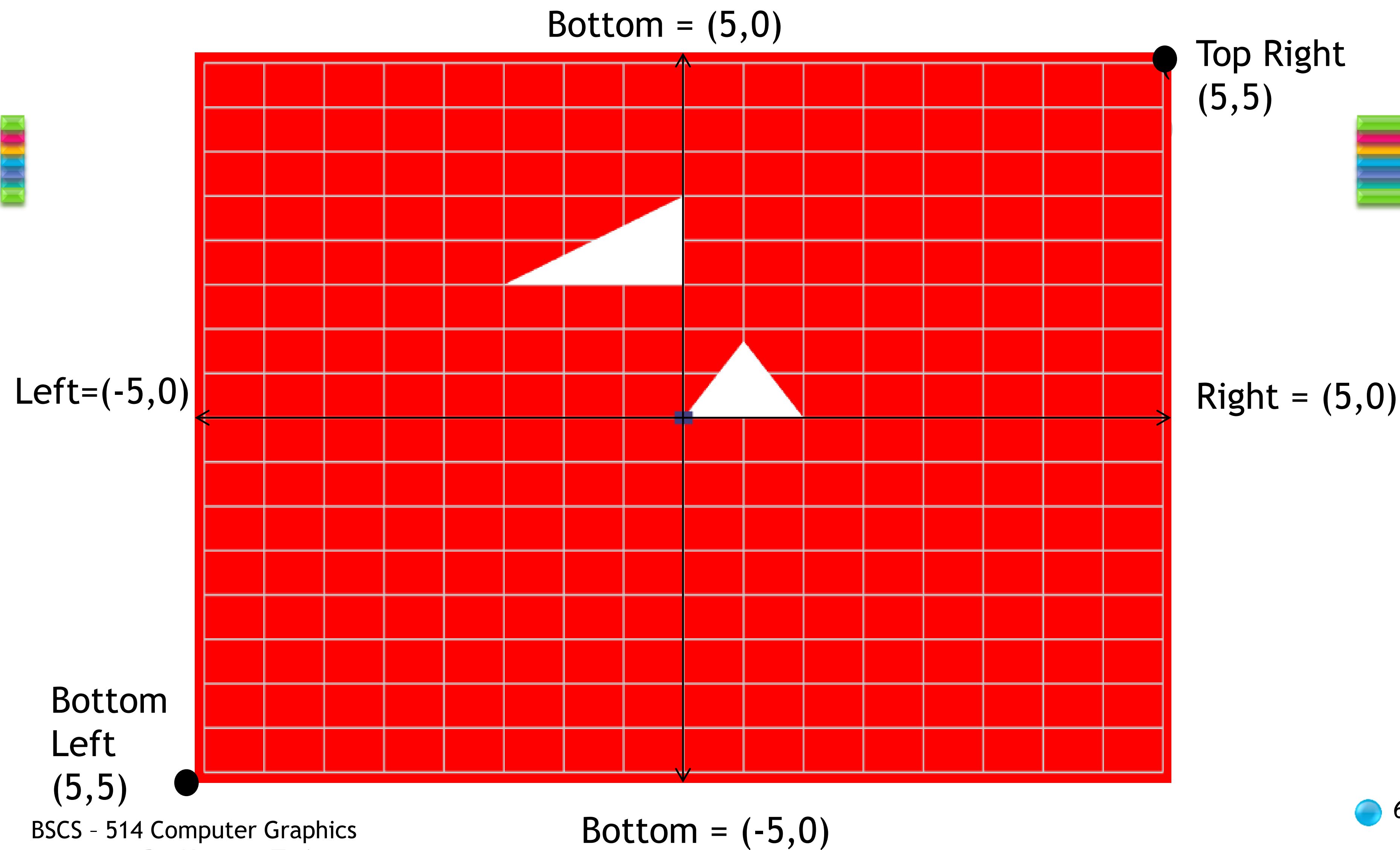
This Lab also uses Webpack file protocol to run node.js application

1) npm init
2) npm install –-save three
3) npm install –-save webpack
4) npm run build
5) Npm run build-dev-watch
6) Files to be used
   - > package.json
   - > webpack.config.js
   - > index.html
   - > index.js

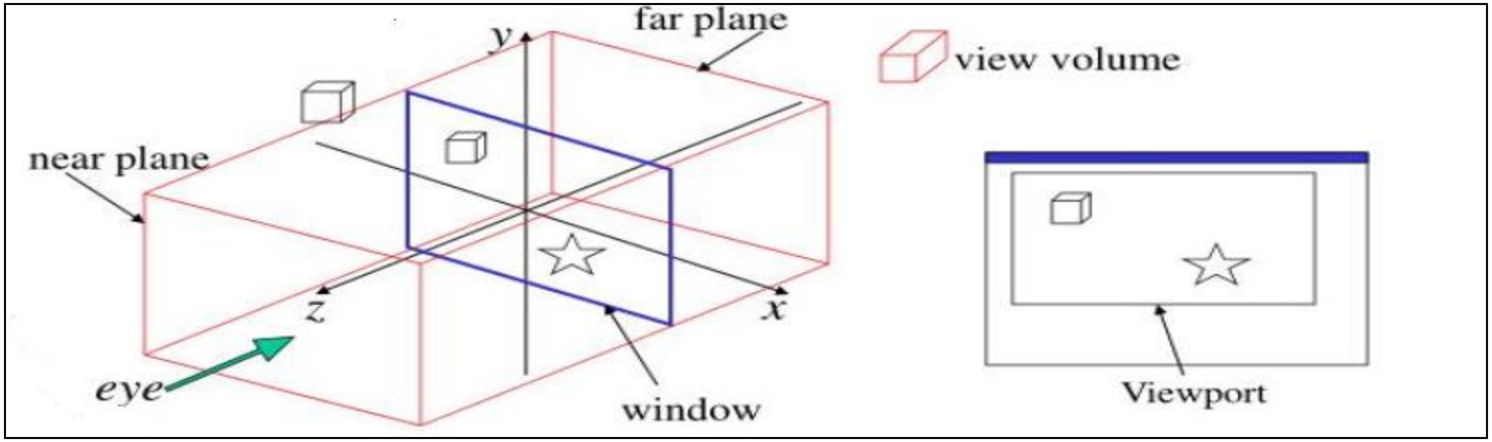
Code to setup camera in 2D and 3D



A Camera can only view a rectangular portion of infinite world which is Known as world window (WW) in 2D computer graphics. WW = (L,R,B,T)



Camera setup for viewing in both 2D and 3D World Space

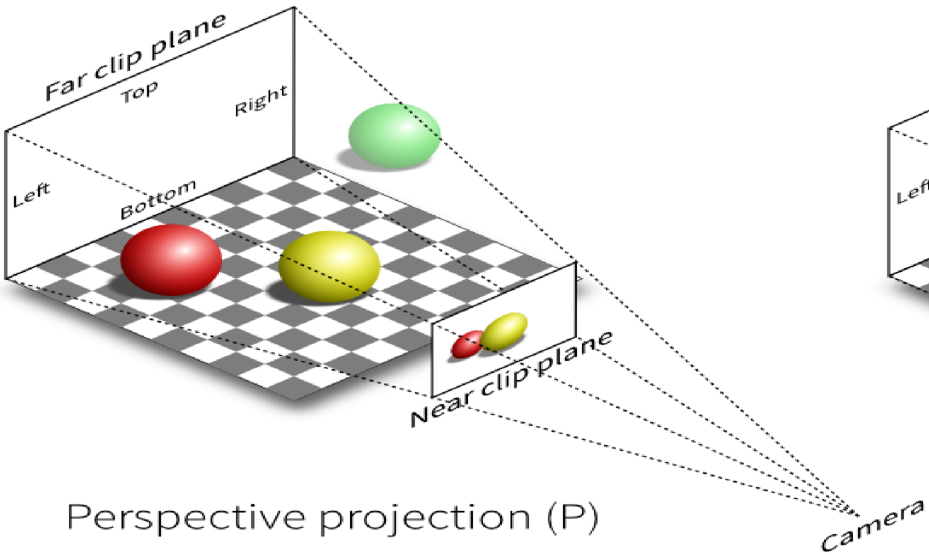


package.json webpack.config.js index.html index.js

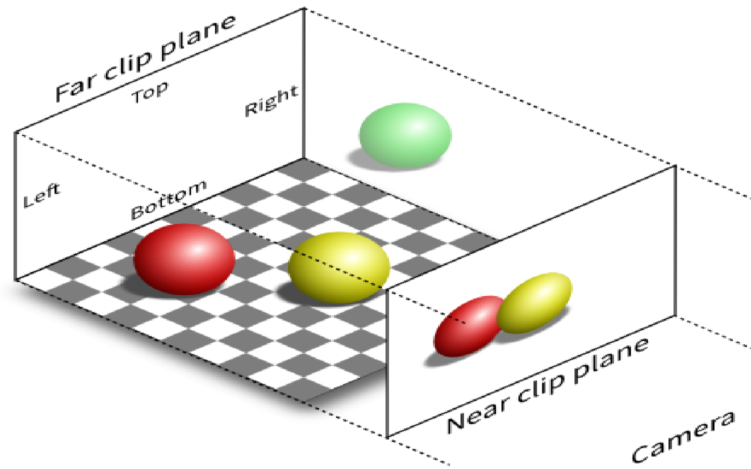
```
// Create an empty scene
var scene = new THREE.Scene();

// Create a basic perspective camera
//var camera = new THREE.PerspectiveCamera( 50, window.innerWidth/window.innerHeight, 0.1, 1000 );
//var camera = new THREE.OrthographicCamera( -10,10,-10,10, 0.1, 1000 );
var camera = new THREE.OrthographicCamera( -10,10,10,-10, 0.1, 1000 );



camera.position.z = 4; // doesn't hurt for 2D
```

Study gluOrtho2D(L,R,B,T), gluPerspective(fov,AR,n,f) and glFrustum(L,R,B,T) in glut from Book Chap 02-Chap 05

3D Camera setup : Perspective vs. Orthographic Projections & Views



Perspective projection (P)



Orthographic projection (O)

package.json | webpack.config.js | index.html | index.js

```
// Create an empty scene
var scene = new THREE.Scene();

// Create a basic perspective camera
//var camera = new THREE.PerspectiveCamera( 50, window.innerWidth/window.innerHeight, 0.1, 1000 );
//var camera = new THREE.OrthographicCamera( -10,10,-10,10, 0.1, 1000 );
var camera = new THREE.OrthographicCamera( -10,10,10,-10, 0.1, 1000 );


camera.position.z = 4; // doesn't hurt for 2D
```

gluPerspective(fov,AR,n,f) and glFrustum(L,R,B,T) in glut

Note the flip on y axis

Code to initialize GPU and output window

Renderer in Java basically meant for certain GPU and browser settings OpenGL vs. JS Code

```
// Create a renderer with Antialiasing
var renderer = new THREE.WebGLRenderer({antialias:true});

// Configure renderer clear color
//renderer.setClearColor("#000000");
renderer.setClearColor("#FF0000");

// Configure renderer size
renderer.setSize( window.innerWidth, window.innerHeight );

// Append Renderer to DOM
document.body.appendChild( renderer.domElement );
```

1 GPU initializes and renderer writes/sets bits in various buffer on GPU

```
//<<<<<<<<<<<<<<<<<<<<<<<< main >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
void main(int argc, char **argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB| GLUT_DEPTH);
      glutInitWindowSize(640,480);
      glutInitWindowPosition(100, 100);
      glutCreateWindow("shaded example - 3D scene");
      glutDisplayFunc(displaySolid);
      glEnable(GL_LIGHTING); // enable the light
      glEnable(GL_LIGHT0);
      glShadeModel(GL_SMOOTH);
      glEnable(GL_DEPTH_TEST); // for hidden surface removal
      glEnable(GL_NORMALIZE); // normalize vectors for proper shading
      glClearColor(0.1f,0.1f,0.1f,0.0f);  // background is light gray
      glViewport(0, 0, 640, 480);
      glutMainLoop();
}
```

2 set Viewport/browser

3 Set LIGHTS/Color & Depth

4 An endless loop that wait and execute events/call backs

Figure 5.63. Complete program to draw the shaded scene.

Modeling

Data Structure: Vertex List, Face List, Normal List

2D Models: Grid, Triangle, Quad, 2D camera

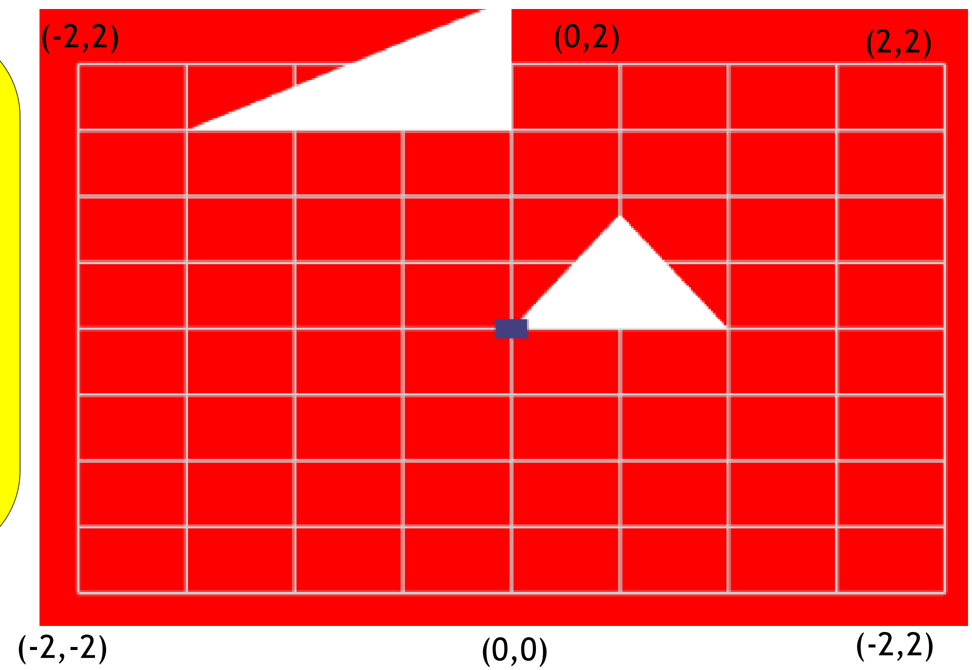
3D Models: Basic Barn, Cube, Tetrahedron, 3D camera

Define Geometry i.e. vertices of Grid : Try labelling all vertices on this mini grid.

```
// ---------------------------------------------
// FUN STARTS HERE
// ---------------------------------------------

//////////////////////////////////////////////////
///////////////////////////////////////////////////
// Create a grid to understand transformation
var size = 4, step = 1;
var geometryGrid = new THREE.Geometry();
var materialGrid = new THREE.LineBasicMaterial( { color: 0xcccccc, opacity: 0.2 } );

```

1- start from -2 on x-axis, treat y as variable i where -2<=*i*<=2 and draw horizontal lines incrementally until *i* becomes equal to +2.

2- Similarly start from -2 on y-axis and do the same as in 1 but this time treat increment in x as variable *i* to draw vertical lines of grid.



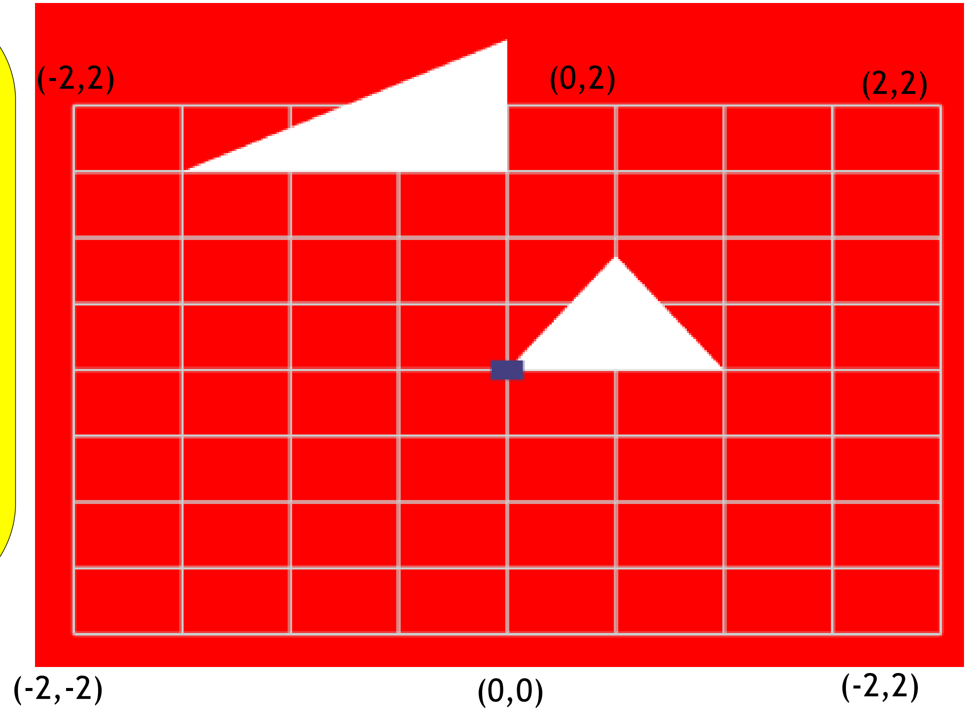
Note same Grid Drawing logic works in 3D except that role of y has been swapped with z to model xz plane instead of xy plane

```
for ( var i = - size; i <= size; i += step ) {
// xy plane for 2D working
geometryGrid.vertices.push(new THREE.Vector3(-size, i,0 )); geometryGrid.vertices.push(new THREE.Vector3(size,i,0));
geometryGrid.vertices.push(new THREE.Vector3(i,-size,0 )); geometryGrid.vertices.push(new THREE.Vector3(i,size,0));

 //xz plane for 3D view and working
 //geometryGrid.vertices.push(new THREE.Vector3(- size,0,i)); geometryGrid.vertices.push(new THREE.Vector3(size,0,i));
 //geometryGrid.vertices.push(new THREE.Vector3(i, 0,-size)); geometryGrid.vertices.push(new THREE.Vector3(i,0,size));
}
```

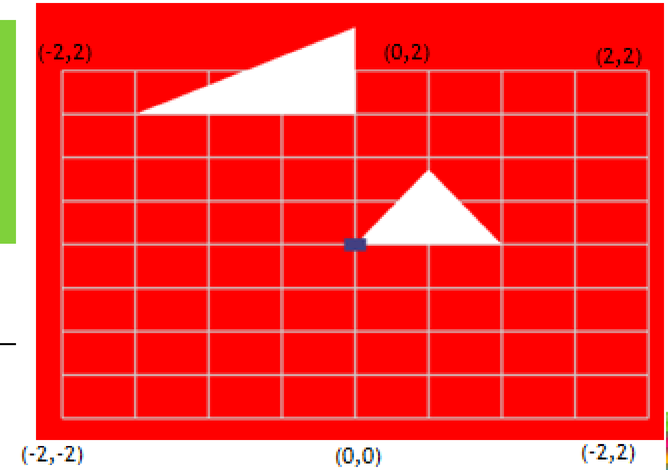
<u>Logic to Model Grid</u>

1- start from -2 on x-axis, treat y as variable i where -2<=*i*<=2 and draw horizontal lines incrementally until *i* becomes equal to +2.

2- Similarly start from -2 on y-axis and do the same as in 1 but this time treat increment in x as variable *i* to draw vertical lines of grid.



Modeling Grid in JS



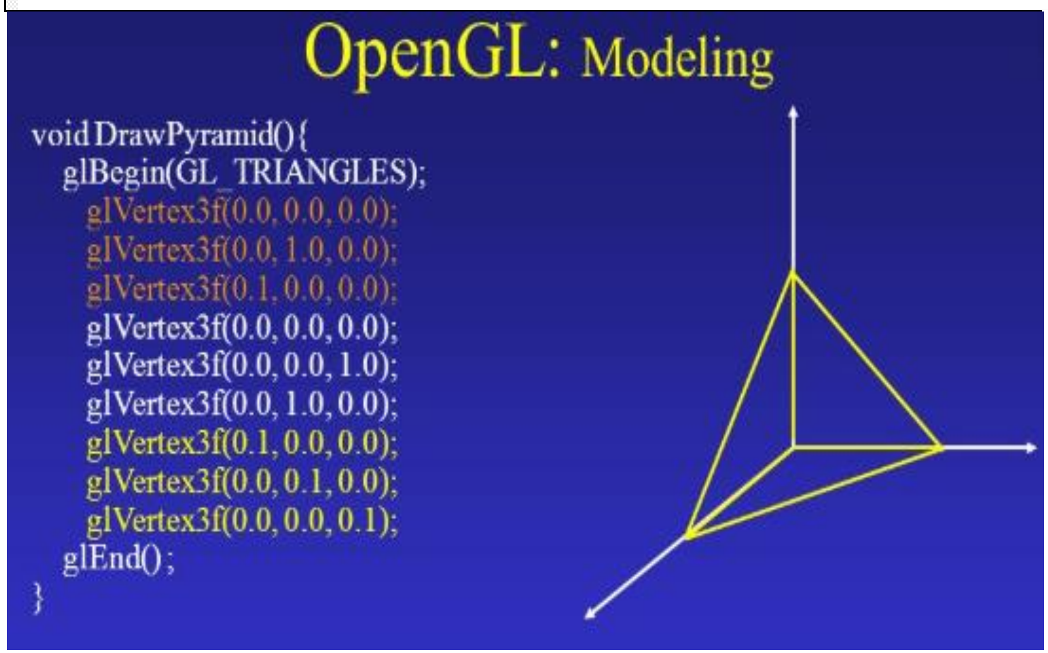

```
// Create a grid to understand transformation
var size = 4, step = 1;
var geometryGrid = new THREE.Geometry();
var materialGrid = new THREE.LineBasicMaterial( { color: 0xcccccc, opacity: 0.2 } );


for ( var i = - size; i <= size; i += step ) {
// xy plane for 2D working
geometryGrid.vertices.push(new THREE.Vector3(-size, i,0 )); geometryGrid.vertices.push(new THREE.Vector3(size,i,0));
geometryGrid.vertices.push(new THREE.Vector3(i,-size,0 )); geometryGrid.vertices.push(new THREE.Vector3(i,size,0));

 //xz plane for 3D view and working
 //geometryGrid.vertices.push(new THREE.Vector3(- size,0,i)); geometryGrid.vertices.push(new THREE.Vector3(size,0,i));
 //geometryGrid.vertices.push(new THREE.Vector3(i, 0,-size)); geometryGrid.vertices.push(new THREE.Vector3(i,0,size));
}
//var line = new THREE.Line( geometryGrid, materialGrid, THREE.LinePieces );
//var line = new THREE.Line( geometryGrid, materialGrid, THREE.Line);
//var line = new THREE.Line( geometryGrid, materialGrid, THREE.LineSegments);
//var line = new THREE.LineLoop( geometryGrid, materialGrid );
var line = new THREE.LineSegments( geometryGrid, materialGrid);  // OK
scene.add(line);
```

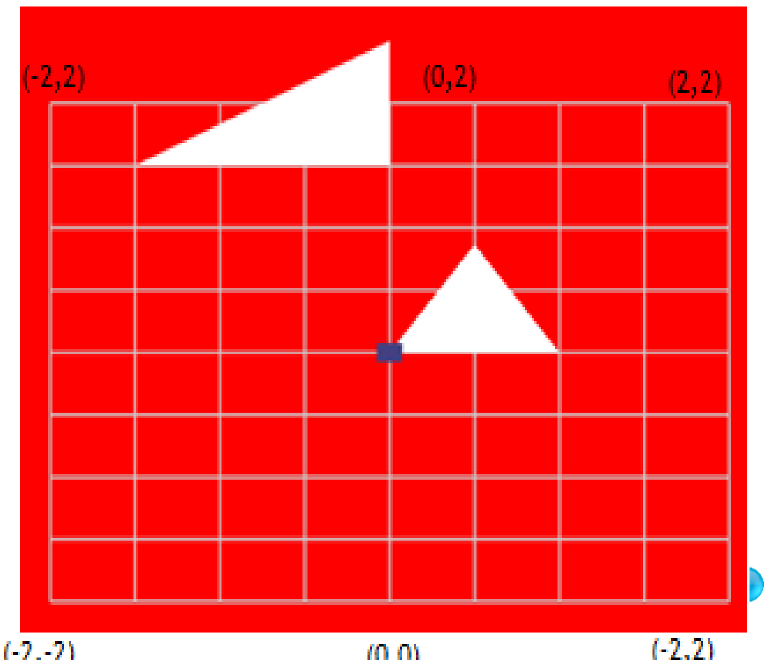
Modeling Input Triangle in JS and glut

Note importance of vertex list and face list, we already practiced in class. Also be careful about clockwise and anticlockwise ordering of vertices

```
////////////////////////////////////////////////////////////////////////
//////////////////// Creating Triangle of our choice ////////////////////////////////////
/////////////Exercise 5.2.16 2nd Edition  Pg No. 227(-3,3) (0,3) (0,5)
var geometryTri = new THREE.Geometry();
var materialTri= new THREE.LineBasicMaterial( { color: 0xff00ff, opacity: 0.2,side:THREE.
geometryTri.vertices.push( new THREE.Vector3( -3, 3, 0 ) );
geometryTri.vertices.push( new THREE.Vector3(0, 3, 0 ) );
geometryTri.vertices.push( new THREE.Vector3( 0, 5, 0 ) );
geometryTri.faces.push(new THREE.Face3(0, 1, 2));
var triangle = new THREE.Mesh(geometryTri,materialTri );
triangle.position.set(0.0, 0.0, 0.0);
scene.add(triangle);
```

Must practice following examples of 3D:
1) Examples 6.2.1 The basic barn
2) Example 6.2.3 Tetrahedron
3) Exercise 6.3.2 The perched cube

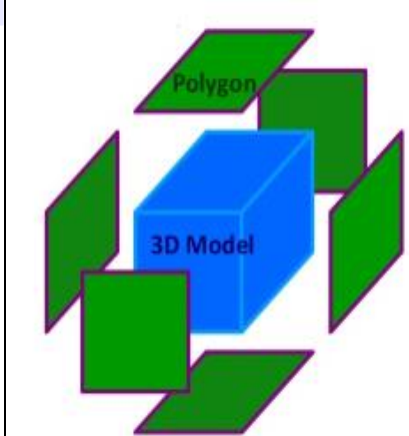




Modeling Cube in JS and glut

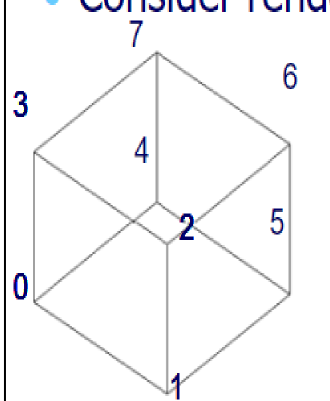
Note importance of vertex list and face list, we already practiced in class. Also be careful about clockwise and anticlockwise ordering of vertices

```
/////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
// Create and testign First Primitive Cube Mesh with basic material
var geometryCube = new THREE.BoxGeometry( 1, 1, 1 );
var materialCube = new THREE.MeshBasicMaterial( { color: "#433F81" } );
var cube = new THREE.Mesh( geometryCube, materialCube );
 // Add cube to Scene
 scene.add( cube );
```



Redundant calculations

- An example: Vertex arrays.
- Consider rendering a cube in OpenGL.




```
glBegin(GL_QUAD);
        glVertex3f(x0,y0,z0);glVertex3f(x1,y1,z1);
        glVertex3f(x2,y2,z2); glVertex3f(x3,y3,z3);
glEnd();
glBegin(GL_QUAD);
        glVertex3f(x1,y1,z1); glVertex3f(x5,y5,z5);
        glVertex3f(x6,y6,z6); glVertex3f(x2,y2,z2);
glEnd();
```

```
// wireBox(w, h, d) makes a wireframe box with width w, height h and
// depth d centered at the origin.  It uses the GLUT wire cube function.
// The calls to glPushMatrix and glPopMatrix are essential here; they enable
// this function to be called from just about anywhere and guarantee that
// the glScalef call does not pollute code that follows a call to myWireBox.
void wireBox(GLdouble width, GLdouble height, GLdouble depth) {
  glPushMatrix();
  glScalef(width, height, depth);
  glutWireCube(1.0);
  glPopMatrix();
}
```

Indexed Face List

- Consists of two lists:
  - A list of coordinates.
  - A list of polygons = a list of lists of vertex indices.

Define global arrays for vertices and colors

1

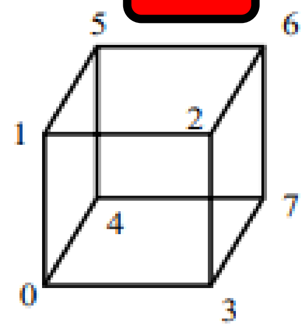
```
GLfloat vertices[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},
{1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},
{1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};
```

```
GLfloat colors[][3] = {{0.0,0.0,0.0},{1.0,0.0,0.0},
{1.0,1.0,0.0}, {0.0,1.0,0.0}, {0.0,0.0,1.0},
{1.0,0.0,1.0}, {1.0,1.0,1.0}, {0.0,1.0,1.0}};
```

Draw cube from faces

2

```
void colorcube( )
{
    polygon(0,3,2,1);
    polygon(2,3,7,6);
    polygon(0,4,7,3);
    polygon(1,2,6,5);
    polygon(4,5,6,7);
    polygon(0,1,5,4);
}
```



Note that vertices are ordered so that we obtain correct outward facing normals

Draw a quadrilateral from a list of indices into the array `vertices` and use color corresponding to first index

3

```
void polygon(int a, int b, int c
, int d)
{
    glBegin(GL_POLYGON);
        glColor3fv(colors[a]);
        glVertex3fv(vertices[a]);
        glVertex3fv(vertices[b]);
        glVertex3fv(vertices[c]);
        glVertex3fv(vertices[d]);
    glEnd();
}
```

- Drawing a cube by its faces in the most straight forward way requires
  - 6 `glBegin`, 6 `glEnd`
  - 6 `glColor`
  - 24 `glVertex`
  - More if we use texture and lighting

4

Reminder

Must practice following examples of 3D:
1) Examples 6.2.1 The basic barn
2) Example 6.2.3 Tetrahedron
3) Exercise 6.3.2 The perched cube

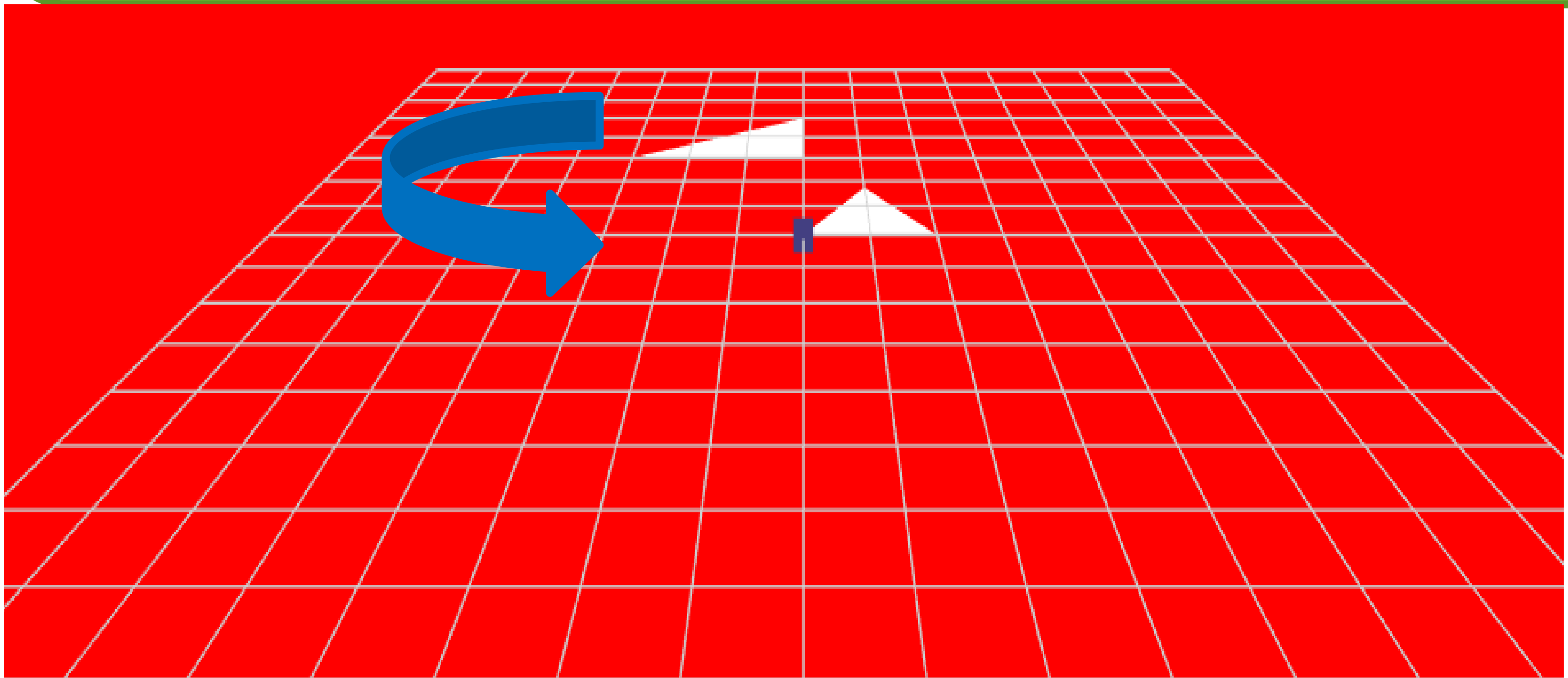
Transformations

Translate, Rotate, Scale

Object Transformation

Coordinate System Transformation

Book Exercise 01:
2nd Edition Pg. No. 277
Ex 5.2.16 Transforming three points



Transform Right Angled Triangle C to Equilateral Triangle D

Chap 05 screen shot
Transformation of Objects

5.2.16. Transforming Three Points. An affine transformation is completely determined by specifying what it does to three points. To illustrate this, find the affine transformation that converts triangle C with vertices (- 3, 3), (0, 3), and (0, 5) into equilateral triangle D with vertices (0, 0), (2, 0), and $(1, \sqrt{3})$, as shown in Figure 5.20.

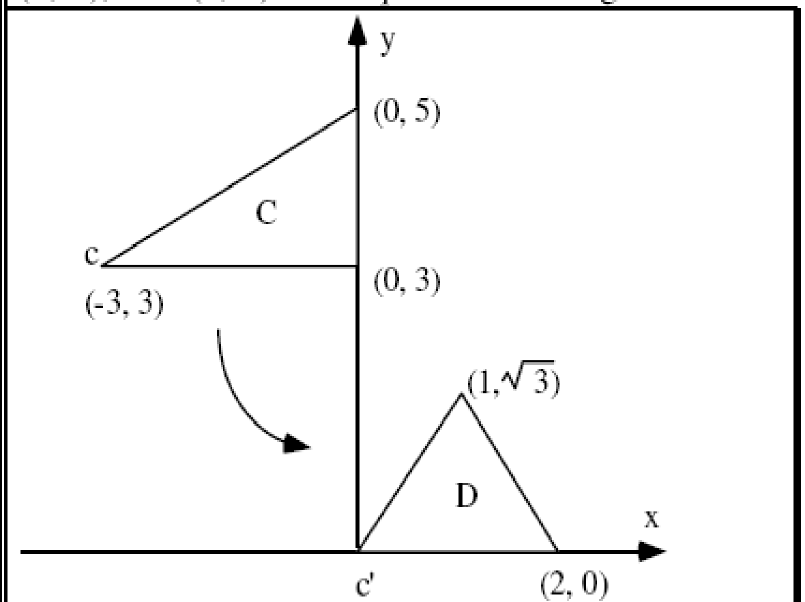


Figure 5.20. Converting one triangle into another.

Do this by a sequence of three elementary transformations:

1. Translate C down by 3 and right by 3 to place vertex c at c'.
2. Scale in x by 2/3 and in y by $\sqrt{3}/2$ so C matches D in width and height.
3. Shear by $1/\sqrt{3}$ in the x-direction to align the top vertex of C with that of D.

Check that this transformation does in fact transform triangle C into triangle D. Also find the inverse of this transformation and show that it converts triangle D back into triangle C.

Inside Rendering Loop / render() function / myDisplay () function, build matrices for transformations given in Triangle Exercise

```
// Render Loop
var render = function () {
  //requestAnimationFrame( render );
```

1: small size cube to represent/mark origin

```
  ////////////////////////////////////////////////////
  ////////////////Practice NO. 1/////////////////////
  //cube.rotation.x += 0.01;
  //cube.rotation.y += 0.01;

  //cube.position.set(0, 0 , 0);
  //cube.rotation.y = Math.PI/4;
  //cube.updateMatrix();

   // Configure renderer clear color
   //renderer.setClearColor("#000000");
   //renderer.setClearColor("#FF0000");


   /////////////////Initialize/////////////////////
   cube.position.set( 0, 0, 0 ); // T , transform matrix
   cube.rotation.set( 0, 0, 0);//R , rotation matrix
   cube.scale.set( 0.3, 0.3, 0.3 );//S , scale matrix
```

Do this by a sequence of three elementary transformations:

1. Translate C down by 3 and right by 3 to place vertex c at c'.
2. Scale in x by 2/3 and in y by $\sqrt{3}/2$ so C matches D in width and height.
3. Shear by $1/\sqrt{3}$ in the x-direction to align the top vertex of C with that of D.

```
var M1 = new THREE.Matrix4().makeTranslation(3, -3, 0);
var M2 = new THREE.Matrix4();
var M3 = new THREE.Matrix4();

var Sx = 2.0/3.0;
var Sy = Math.sqrt(3)/2;
var Sz = 1.0;


M2.set(Sx, 0,  0,  0,
                0,      Sy,  0,  0,
                0,   0,   Sz,   0,
                0,      0,   0,   1  );

var xShear = -1.0/ Math.sqrt(3.0);
var yShear = 1;
var zShear = 1;
M3.set(   1,    xShear,  0,  0,
                0,      yShear,  0,  0,
                0,   0,   zShear,   0,
                0,      0,   0,   1  );
```

T1 = Translation
T2 = Scaling
T3 = Shear in x

2

Now Combine T1, T2 and T3 in a single matrix in reverse order

T1 = Translation
T2 = Scaling
T3 = Shear in x

Do this by a sequence of three elementary transformations:

1. Translate *C* down by 3 and right by 3 to place vertex *c* at *c'*.
2. Scale in *x* by 2/3 and in *y* by $\sqrt{3}/2$ so *C* matches *D* in width and height.
3. Shear by $1/\sqrt{3}$ in the *x*-direction to align the top vertex of *C* with that of *D*.

```
    //var desiredTransform = new THREE.Matrix4().makeRotationY(Math.PI / 180);
    var M = new THREE.Matrix4();
    M = M.multiply(M3).multiply(M2).multiply(M1);
    //M = M.multiply(M2).multiply(M1);
    console.log(M);

    var newTriangle = triangle.clone();
    newTriangle.matrixAutoUpdate = false;
    var color = 0xffffff; // Your color
    newTriangle.material.color.setHex( color );


    newTriangle.applyMatrix(M);
    //axesHelper.applyMatrix(M)
    newTriangle.verticesNeedUpdate = true;
        scene.add(newTriangle);

   // Render the scene
    renderer.render(scene, camera);
};

render();
```

Why Reverse

Compare Your Manual Calculation and Results with JS solution

Composite Matrix M = T3 T2 T1

$$Q = T3\ T2\ T1\ P$$

Composite Matrix $M^T = (T3\ T2\ T1)^T$

$$Q^T = P^T (T1^T\ T2^T\ T3^T)$$

```
index.js:356
▼ Matrix4 {elements: Array(16)}
  ▼ elements: Array(16)
      0: 0.6666666666666666
      1: 0
      2: 0
      3: 0
      4: -0.5
      5: 0.8660254037844386
      6: 0
      7: 0
      8: 0
      9: 0
      10: 1
      11: 0
      12: 3.5
      13: -2.598076211353316
      14: 0
      15: 1
      length: 16
    ► __proto__: Array(0)
  ► __proto__: Object
```

JS results

New Transformed Triangle

Manual results

$$\Delta_{NEW} = M \cdot \Delta_{old}$$

$$\Delta_{NEW} = \begin{pmatrix} 2/3 & -1/2 & 7/2 \\ 0 & \sqrt{3}/2 & 3\sqrt{3}/2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -3 & 0 & 0 \\ 3 & 3 & 5 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\Delta_{NEW} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 2 & \sqrt{3} \\ 1 & 1 & 1 \end{pmatrix}$$

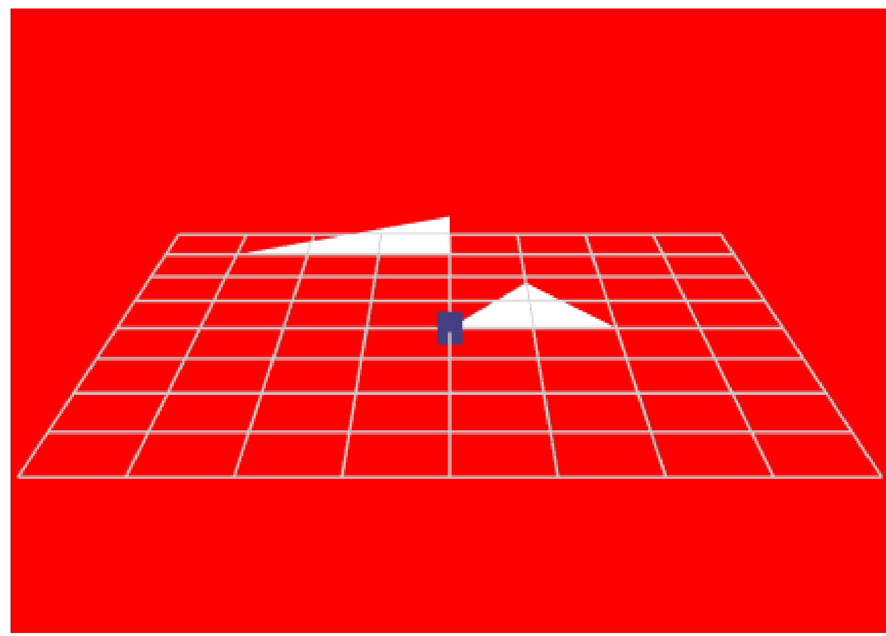
Change View from 2D to 3D for Triangle Problem

1

```
// Create an empty scene
var scene = new THREE.Scene();

// Create a basic perspective camera
var camera = new THREE.PerspectiveCamera( 50, window.innerWidth/window.innerHeight, 0.1, 1000 ); // angle decrease is zoom i
//var camera = new THREE.OrthographicCamera( -10,10,-10,10, 0.1, 1000 ); // 3D world window = gluOrtho2D(-10,10,10,-10)
//var camera = new THREE.OrthographicCamera( -10,10,10,-10, 0.1, 1000 ); // flipped 3D world window = gluOrtho2D(-10,10,-10,1
```

2

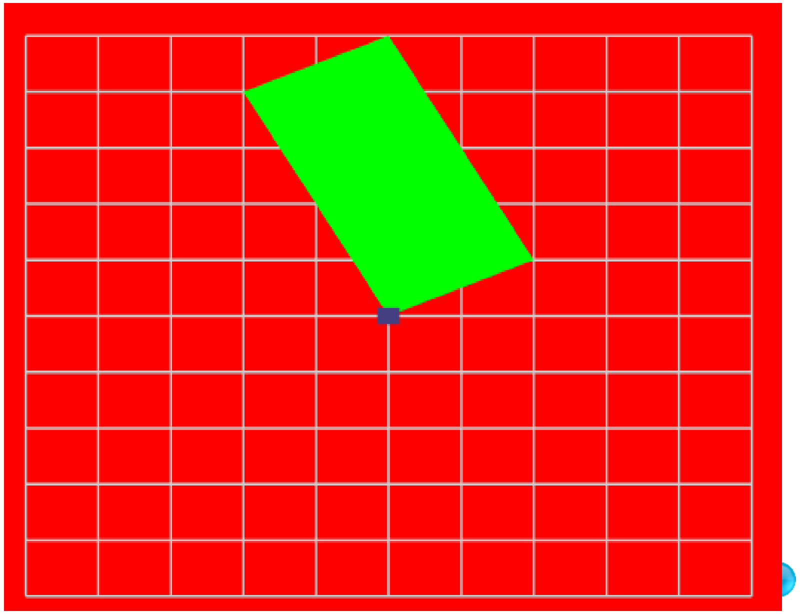
```
renderer.setClearColor("#ff0000");


camera.position.x = 0;//5;
camera.position.y = -15.5;
camera.position.z = 5.5;
camera.lookAt(0, 0, 0);

  var M1 = new THREE.Matrix4().makeTranslation(3, -3, 0);
  var M2 = new THREE.Matrix4();
  var M3 = new THREE.Matrix4();

  var Sx = 2.0/3.0;
  var Sy = Math.sqrt(3)/2;
  var Sz = 1.0;
```



Book Exercise 02: 2nd Edition Pg. No. 278
Ex 5.2.16 Normalize a Box

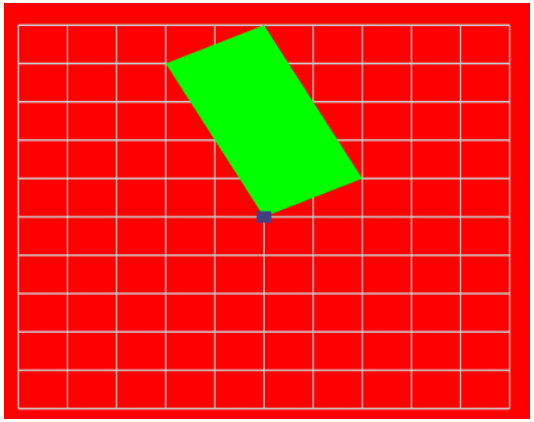
5.2.19. Normalizing a Box. Find the affine transformation that maps the box with corners (0, 0), (2, 1), (0, 5), and (- 2, 4) into the square with corners (0, 0), (1, 0), (1, 1), and (0, 1). Sketch the boxes.

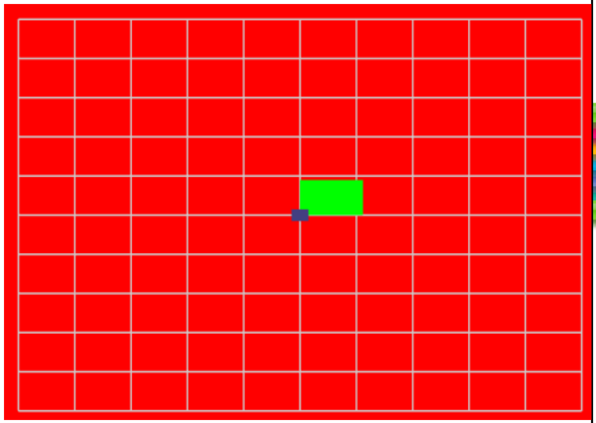
```
/////////////////////////////////////////////////////////////////////////
//////////////////// Creating Box of my choice///////////////////////////////////////
/////////////Exercise 5.2.19 2nd Edtion Pg 228(0,0) (2,1) (0,5) (-2,4)
var geometryBox = new THREE.Geometry();
var materialBox= new THREE.LineBasicMaterial( { color: 0x00ff00, opacity: 0.2,side:THREE.DoubleSide   } );
geometryBox.vertices.push( new THREE.Vector3( 0, 0, 0 ) );
geometryBox.vertices.push( new THREE.Vector3(2, 1, 0 ) );
geometryBox.vertices.push( new THREE.Vector3( 0, 5, 0 ) );
geometryBox.vertices.push( new THREE.Vector3( -2, 4,0 ) );
//geometryBox.vertices.push( new THREE.Vector3( 0, 0,0 ) );
geometryBox.faces.push(new THREE.Face3(0, 1, 2));
geometryBox.faces.push(new THREE.Face3(0, 2, 3));
var boxPolygon = new THREE.Mesh(geometryBox,materialBox );
boxPolygon.position.set(0.0, 0.0, 0.0);
scene.add(boxPolygon);
```



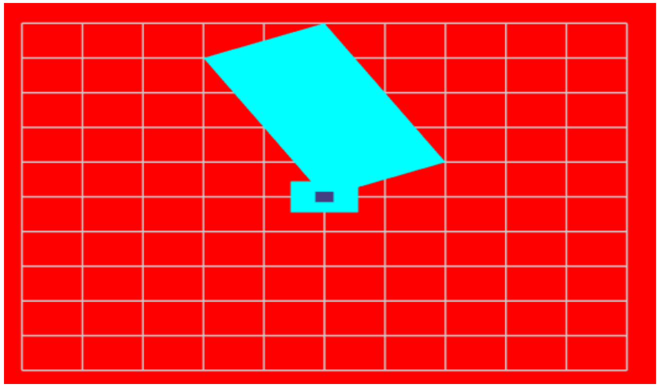
Book Exercise 02: 2nd Edition Pg. No. 278
Normalize a Box Solution No. 1

```
var scaleMatrix = new THREE.Matrix4();
var Sx = 1.0/2.0;
var Sy = 1.0/5.0;

scaleMatrix.set(   Sx, 0,  0,  0,
              0,    Sy, 0,  0,
              0,  0,   1,   0,
              0,    0,   0,   1  );
//boxPolygon.applyMatrix( matrix );
//boxPolygon.rotateOnAxis(axis,-11 * Math.PI/180);// ok to make it align with axis T2
//boxPolygon.scale.set(1.0/2.0,1.0/5.0,1); // T3  sad skewed axis problem due to non-uniform scaling

/////////////Normalize Box Solution No.1////////////////////
boxPolygon.matrixAutoUpdate = false;
let R= new THREE.Matrix4().makeTranslation(0, 0, 0).makeRotationZ(-26.57*Math.PI/180);
var SR = scaleMatrix.multiply(R);   // T1: Rotation; T2: Scaling  ==> T2T1 = S*R ==> Read from Right to Left
/////  var RS = R.multiply(scaleMatrix);   // T1: Scaling; T2: Rotation  // wrong attempt and wrong intuition
boxPolygon.applyMatrix(SR); // T2 premultiplying
boxPolygon.verticesNeedUpdate = true;
/////////////end Normalize Box Solution No.1////////////////////
```





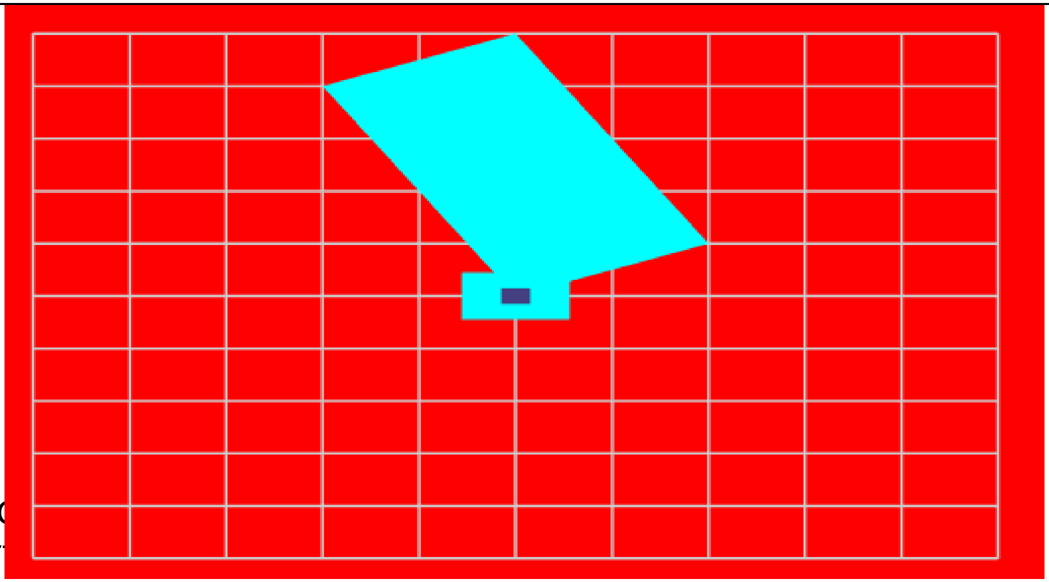
Book Exercise 02: 2nd Edition Pg. No. 278
Normalize a Box Solution No. 2



```
/////////////Normalize Box Solution No.2/////////////////////
    var newPolygon = boxPolygon.clone();
    var color = 0x00ffff; // Your color
    newPolygon.material.color.setHex( color );
    scene.add(newPolygon);

     newPolygon.matrixAutoUpdate = false;
     var T1 = new THREE.Matrix4().makeTranslation(0, -2.5, 0);

     var T2= new THREE.Matrix4().makeRotationZ(-26.57*Math.PI/180);  // not rotate w.r.t origin as explained below
     //var rotationMatrix= new THREE.Matrix4(); //var theta = 26.57;
     //let c = Math.cos(26.57 * Math.PI/180); let s = Math.sin(26.57 * Math.PI/180);
     //rotationMatrix.set(   c,   s,  0,  0, -s,    c,  0,  0, 0,   0,   1,   0, 0, 0,   0,   1  );
     //T2 = T2.copy(rotationMatrix);
     var T3 = new THREE.Matrix4();
     var scaleMatrix = new THREE.Matrix4();
     var Sx = 1.0/2.0;    var Sy = 1.0/5.0;
     scaleMatrix.set(   Sx,  0,   0,   0,
                0,     Sy,  0,   0,
                0,   0,   1,   0,
                0,     0,   0,   1  );
     T3 = T3.copy(scaleMatrix);
```

Book Exercise 02: 2nd Edition Pg. No. 278 Normalize a Box Solution No. 2

```
var TSRT = new THREE.Matrix4();
var TSRT = TSRT.copy(T3); // First Load Rotation  T2
//TSRT = TSRT.multiply(T2); // Now Load  T1 =Translation to make sure that center point of box is at origin
//TSRT = TSRT.multiply(T1);
var TSRT = TSRT.multiply(T2).multiply(T1); //T3: Sclaing; T2 = Rotation  T1 = Translation => T3*T2*T1=T*R*S
console.log(TSRT); //Read Object Transformation from Left to right
newPolygon.applyMatrix(TSRT);
 //newPolygon.verticesNeedUpdate = true;
 //newPolygon.matrixWorldNeedsUpdate = true;

  /////////////end Normalize Box Solution No.2////////////////////
```



Practice Codes from Chap 05 Transformation of Objects

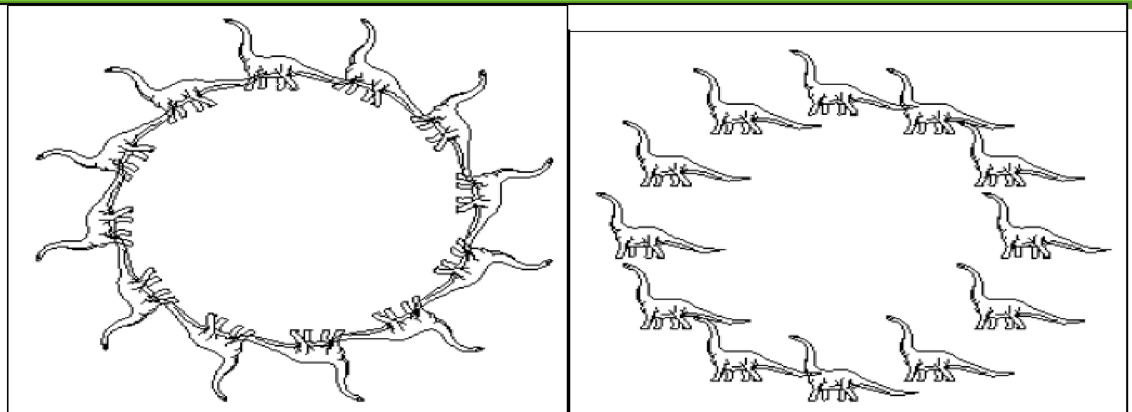


Figure 5.42. Two patterns based on a motif. a). each motif is rotated separately. b). all motifs are upright.

Suppose that drawDino() draws an upright dinosaur centered at the origin. In part a) the coordinate system for each motif is first rotated about the origin through a suitable angle, and then this coordinate system is translated along its y-axis by H units as shown in the following code. Note that the CT is reinitialized each time through the loop so that the transformations don't accumulate. (Think through the transformations you would use if instead you took the point of view of transforming points of the motif.)

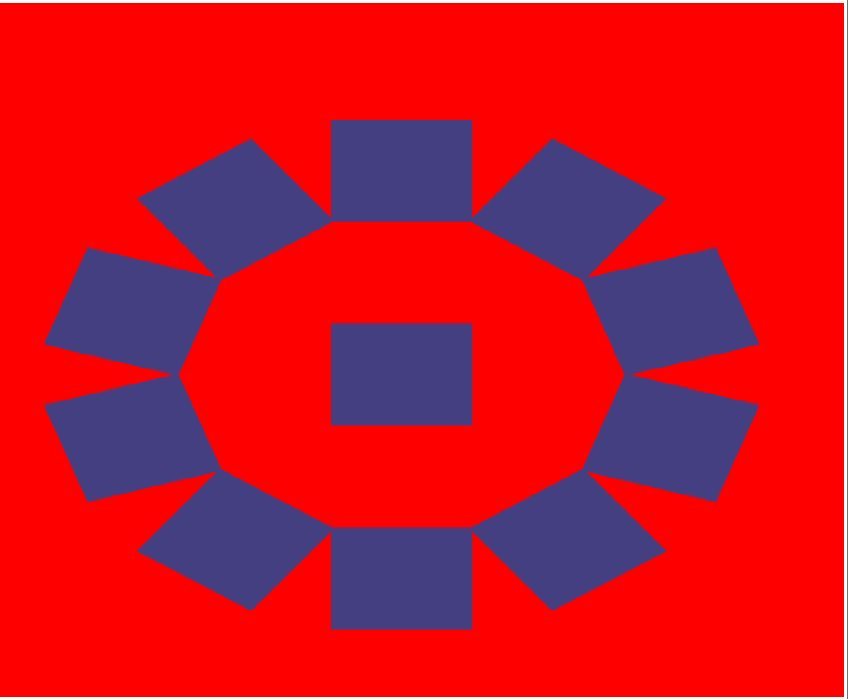
```
const int numMotifs = 12;
for(int i = 0; i < numMotifs; i++)
{
       cvs.initCT(); // init CT at each iteration
       cvs.rotate2D(i * 360 / numMotifs); // rotate
       cvs.translate2D(0.0, H); // shift along y-axis
       drawDino();
}
```

An easy way to keep the motifs upright as in part b) is to "pre-rotate" each motif before translating it. If a particular motif is to appear finally at 120°, it is first rotated (while still at the origin) through -120°, then translated up by H units, and then rotated through 120°. What ajustments to the preceding code will achieve this?

Recall Code and Output of First Dino Pattern

Figure 5.42. Two patterns based on a motif. a). each motif is rotated separately

The problem is to undo rotation at final position of box



```
// Render Loop
var render = function () {
  //requestAnimationFrame( render );

  /////////////////Practice NO. 1/////////////////////
  //cube.rotation.x += 0.01;
  //cube.rotation.y += 0.01;

   /////////////////Practice NO. 2/////////////////////
   //cube.rotatiorn.z = Math.PI/4; // 45 degree
   //cube.translateX(2.5);
   //cube.translateY(3.0); // output shows that y increases downward


   /////////////////Practice NO. 3/////////////////////
   ///////////////// Drawing multiple objects without timer////////////
   /////////////////Walking/turning effect along Circle//////////

   for(var i = 0; i < 10; i++){
        var newCube = cube.clone();
            newCube.rotation.z = i*(360/10) * (Math.PI/180); // 45 degree
            newCube.translateY(2.0);
            //newCube.scale.set(2,2,2);
            scene.add(newCube);
}
    // Render the scene
    renderer.render(scene, camera);
};

render();
```

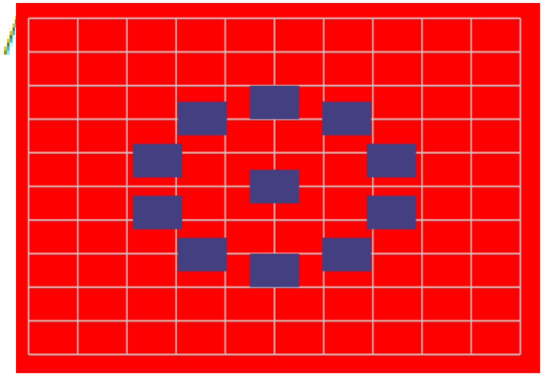
Recall Code and Output of 2nd Dino Pattern

Figure 5.42. Two patterns based on a motif. a). each motif is rotated separately

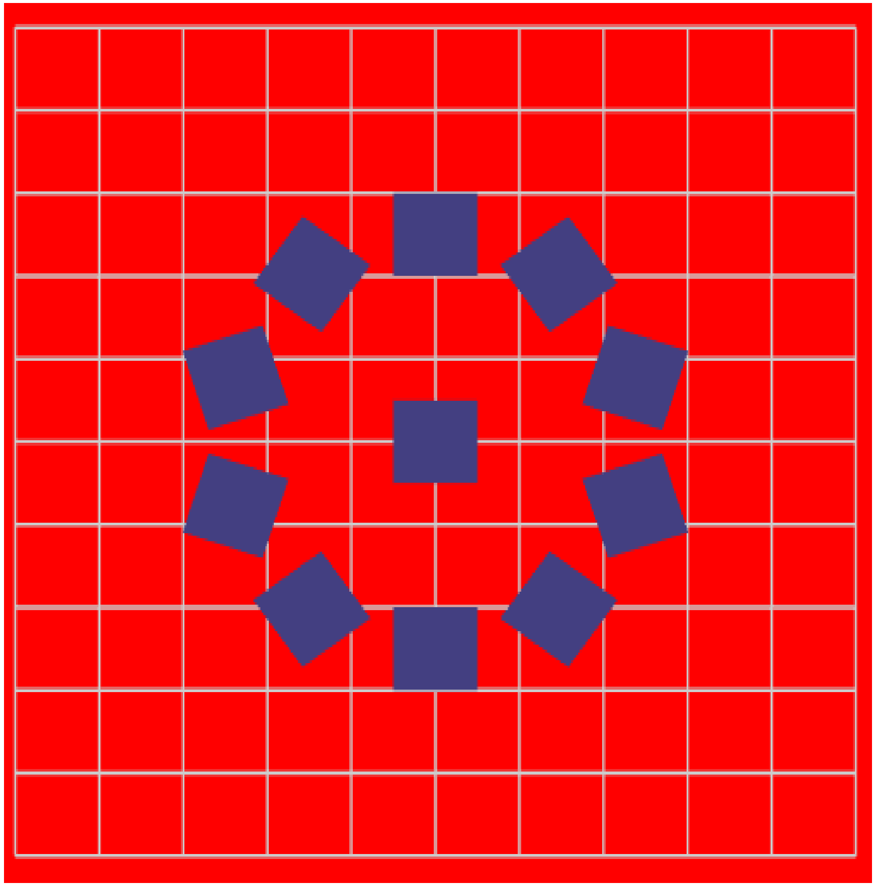
The problem is to undo rotation at final position of box

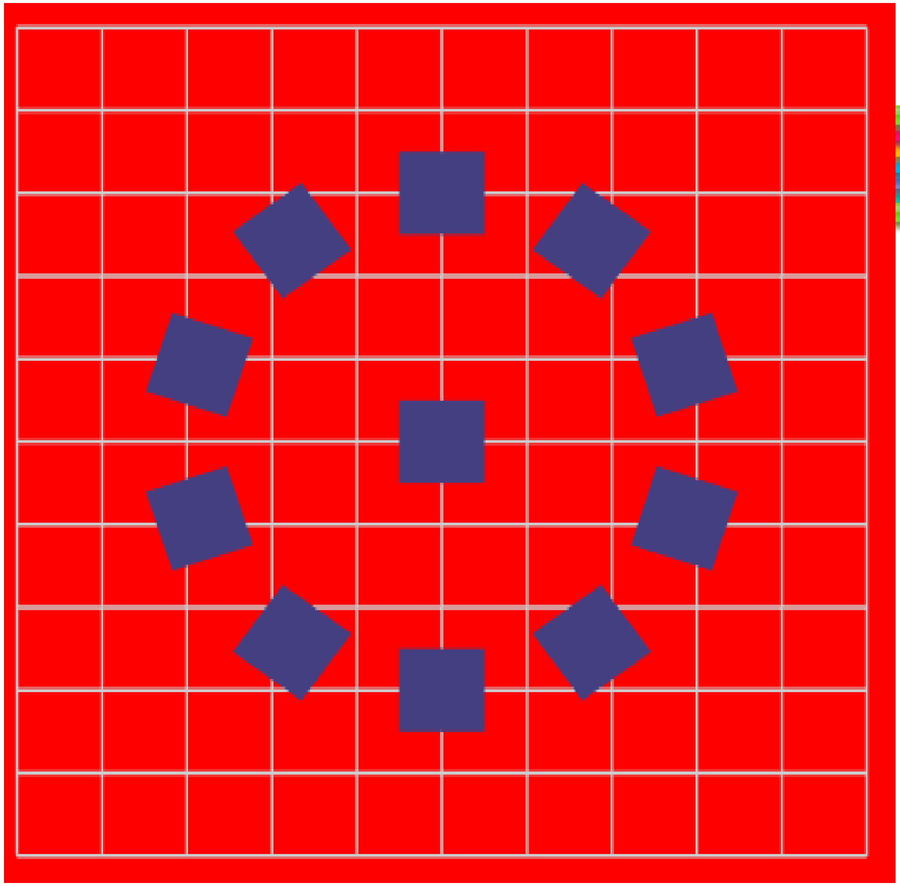
```
/////////////////////////////////////////////////
// building up raw scene for practice
 var axis = new THREE.Vector3(0,0,1);
 axis.normalize();

// Render Loop
var render = function () {
  //requestAnimationFrame( render );
```

```
//////////////////correctged RTRInv Arrangement //////////////////////
//////////////////Single Seq is working ///////////////////
 for(var i = 0; i < 10; i++){
    var newCube = cube.clone();
    //var newAxis = axis.clone();
    //newAxis.normalize();

    newCube.rotateOnAxis(axis, i*(360/10) * (Math.PI/180));
    newCube.translateY(2.5); // output shows that y increases downward
    newCube.rotateOnAxis(axis, -i*(360/10) * (Math.PI/180));
    scene.add(newCube);
}
```

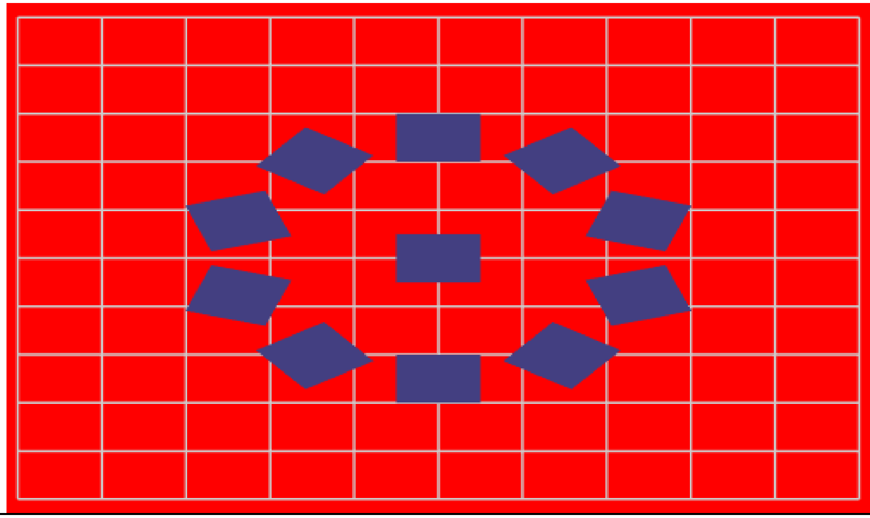


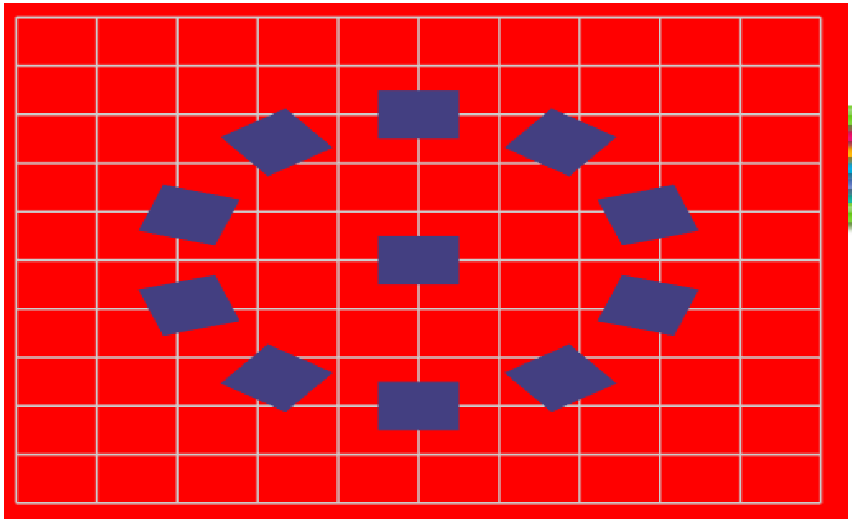
Trouble and problem faced in box orientation correction





Trouble and problem faced in box orientation correction





```
/////////////////Troubled RTRInv Arrangement from book///////////////////
////////////inverseRotate at origin- Go to desired position- UndoRotation ,
 for(var i = 0; i < 10; i++){
    var newCube = cube.clone();

    newCube.rotation.z =  i*(360/10) * (Math.PI/180) // 45 degree
    newCube.translateY(2.5); // output shows that y increases downward
    newCube.rotation.z = - i*(360/10) * (Math.PI/180); // 45 degree

    scene.add(newCube);
}
```

```
/////////////////T+R combination 2 //////////////////////
////////////////Practice NO. 3 //////////


    for(var i = 0; i < 10; i++){
        var newCube = cube.clone();
            newCube.rotation.z = i*(360/10) * (Math.PI/180);
            newCube.translateY(3.0);
            newCube.scale.set(1,1,1);
            scene.add(newCube);
}
```

Try more examples and exercises from Book .

Practice Exercises.

Exercise 5.2.8. The classic: the Window to Viewport Transformation.

We developed this transformation in Chapter 3. Rewriting Equation 3.2 in the current notation we have:

$$\tilde{M} = \begin{pmatrix} A & 0 & C \\ 0 & B & D \\ 0 & 0 & 1 \end{pmatrix}$$

where the ingredients A, B, C, and D depend on the window and viewport and are given in Equation 3.3. Show that this transformation is composed of:

- A translation through $(-W.l, -W.b)$ to place the lower left corner of the window at the origin;
- A scaling by (A, B) to size things.
- A translation through $(V.l, V.b)$ to move the corner of the viewport to the desired position.