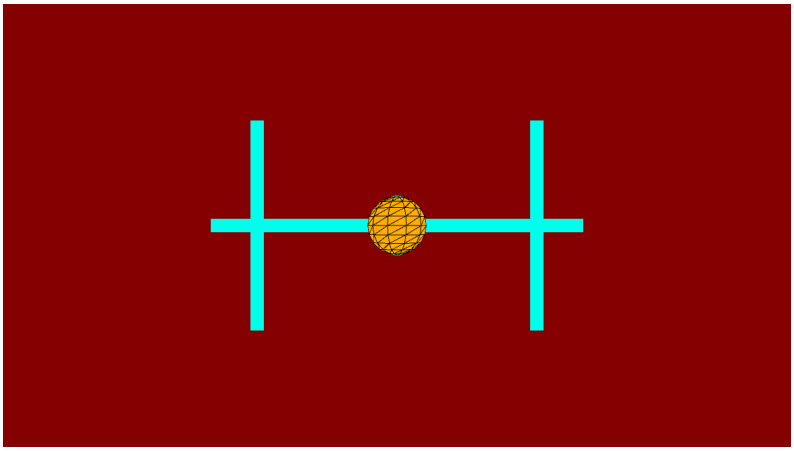
CSC 220 Object-Oriented Multimedia Programming, Fall 2020

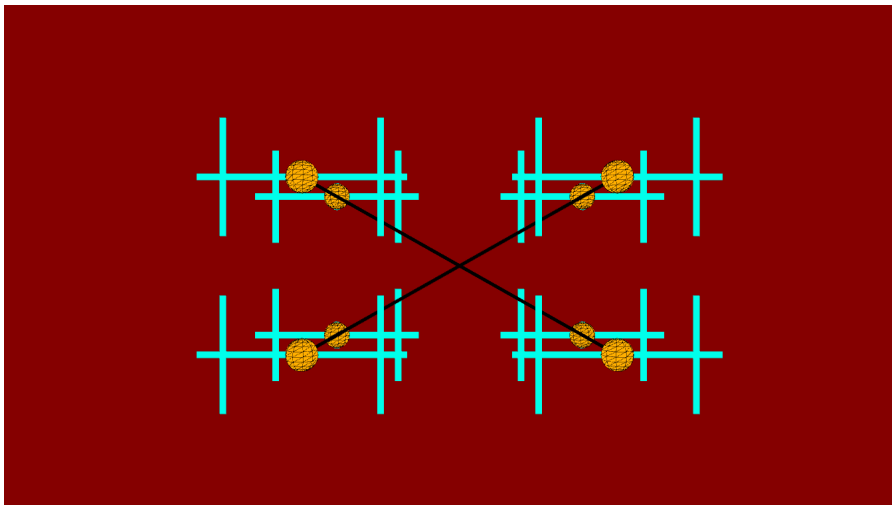
Dr. Dale E. Parson, Assignment 4, recursive partitioning of 3D space.
This assignment is due via **D2L Dropbox** **Assignment 4** due by **11:59 PM on Thursday December 3**. **10% penalty for each day it is late**.

The starting point code for RecursiveShape is in the page entitled **CSC220Fall2020Recur3D.txt** linked to the course page. Copy & paste it into **Processing** and **Save As CSC220Fall2020Recur3D**. Proceed according to the following instructions.

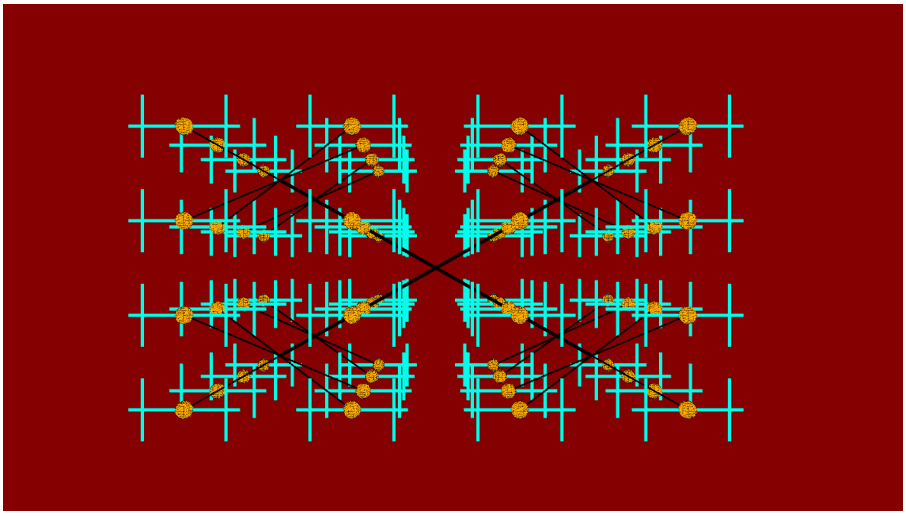
Recursion is a programming technique in which a function calls itself, either directly, or indirectly through another function. Graphical programming often makes use of a recursive function like a "cookie cutter" with parameters that customize the location and scale of the cookie cutter. You will use recursion within sketch **CSC220Fall2020Recur3D** to divide the space of one Processing graphical window that is **width X height** in size into a number of smaller, adjacent "virtual graphical windows" that typically **may** maintain the same width X height aspect ratio, but where the actual width and height have been scaled, and the 0,0 center location has been translated, to fit each of the smaller virtual windows. The next few illustrations show part of my handout solution to the problem. You will create your own shape according to requirements appearing later in this document. We will go over my handout code and the concepts of recursion in class.



Base case showing my custom shape from function drawShape with a recursion depth of 0.



Recursive case with a recursion depth of 1. Each level of recursion subdivides a cuboid region into 8 adjacent rectangular regions, each with ½ the width, ½ the height, and double the layers of the original "parent" region.



Recursive case with a recursion depth of 2 gives 64 sub-regions, or generally, 8^{depth} sub-regions.

The basic strategy is this:

1. Decide whether it is time to draw the shape, based on a depth parameter to function **drawRecursiveShape** in the handout code.
2. If it is time to draw the shape:
   a. push(), then rotate() if the rotate parameter to drawRecursiveShape is non-0.
   b. Call **drawShape**() to draw the shape at the current scale, then pop().
   c. Increment parameter baseCaseIndex to assign a unique integer ID to each base case call to drawShape(…).
3. Else (not time to draw the shape due to insufficient depth):
   a. For each region to be subdivided from this region (Mine is 8; yours will be something else.)
      I. push(), set strokeWeight() and stroke()
      II. If rotating the overall group of shapes (isSpaceRotating), do that.

III. Optionally draw a line from the region center to this sub-region center.
IV. translate() to the center of the sub-region.
V. scale() to the size of the sub-region, relative to the region being divided.
VI. Call **drawRecursiveShape** recursively to subdivide that sub-region.
VII. pop() and return.

Base case function **drawShape**() simply draws the shape, using **width and height** to delimit its size. Its **myShapeIndex** parameter is a non-negative, consecutive integer value for each call to drawShape(), used to ID the call that displays the EasterEgg. We will go over the whole thing in class.

REQUIREMENT 1 (40%) Replace my code in function drawShape() with your own distinct shape, using a new combination of both 2D and 3D shapes. You can still use the shapes I used, but you **must** use at least two other shape-drawing functions. Also, you **must** nest at least 1 2D shape inside a 3D shape. Your code should use width and height to help ensure that the shape displays in the currently scaled sub-region.

REQUIREMENT 2 (40%) Change else portion of function **drawRecursiveShape** so that it subdivides its **width X height X depth** region into different sub-regions than mine. Mine divides the region into 4 adjacent, identically sized sub-regions, stacked 2 deep to give 8 overall. The simplest way to satisfy this requirement would be to subdivide each layer into 9 adjacent, identically sized sub-regions, but you could also subdivide each layer into 3 adjacent, non-identically sized sub-regions as shown here.

REQUIREMENT 3 (20%) Supply and display your own **PImage** file (JPG, PNG, etc.) OR custom **PShape** construction OR SVG file OR custom formatted **text()** to replace my EasterEgg.png file. Display it inside your drawShape(…) function for one of your shapes, using the approach I will discuss on 11/17. Note that in my drawShape(…) function, the 8 in this line of code depends on the number of shapes expanded by each recursive call. Yours will most likely be other than 8. Your Easter egg must lie partially inside and partially outside of its surrounding shape, so that a user can navigate in to see the center of the egg.

```
float nodes = pow(8, recursionDepth); // based on 8 shapes per subdivision
```

Save this sketch as **CSC220Fall2020Recur3D**.

If you use image files or .svg vector files in your sketch (this is an option open to you), make sure to turn in those files as well, or turn in the entire sketch directories if you use such files.