Computer graphics simulates worlds that look real, act real, sound real, and feel real.
Develop a computational model of pinhole camera
  Modeling and rendering

Identify primitive elements, rules for their combination, and common patterns of use
  Geometric points, vectors, and coordinate systems

Improve computational model of pinhole camera
  Matrices and vectors
A COMPUTATIONAL MODEL OF PINHOLE CAMERA

Modeling and rendering
Computer graphics simulates worlds by dividing the task into two problems: modeling and rendering.
Geometric modeling describes objects to be displayed

```python
def cube():
    """ Describe a unit cube shape by the location of its eight corners """
    points = [(0., 0., 0.),
              (1., 0., 0.),
              (0., 1., 0.),
              (1., 1., 0.),
              (0., 0., 1.),
              (1., 0., 1.),
              (0., 1., 1.),
              (1., 1., 1.)]
    return points
```
Geometric modeling combines simple objects to describe more complex objects.

def scale_shape(s):
    """ Scale shape by scaling points with the given factor s """
    def scale(point):
        return s[0]*point[0], s[1]*point[1], s[2]*point[2]
    return scale

def move_shape(t):
    """ Move shape by translating points with the given offset """
    def translate(point):
        return point[0]+t[0], point[1]+t[1], point[2]+t[2]
    return translate
For example, scaling transformation reduces or enlarges an object

```python
smaller_cube = map(scale_shape((0.1, 0.1, 0.1)), cube())
larger_cube = map(scale_shape((10., 10., 10.)), cube())
stick = map(scale_shape((0.1., 0.1, 1.)), cube())
plank = map(scale_shape((0.1, 1., 1.)), cube())
```
And, composition can create even more complex objects

```python
bottom = map(scale_shape((1., 0.1, 1.)), cube())

middle = map(move_shape((0., 0.1, 0.)),
    map(scale_shape((0.9, 1., 0.9)), bottom))

top = map(move_shape((0., 0.2, 0.)),
    map(scale_shape((0.9*0.9, 1., 0.9*0.9)), bottom))
```
Computer graphics simulates worlds by dividing the task into two problems: modeling and rendering.
Rendering displays an image by mapping geometric models onto a two-dimensional surface.
Perspective projection computes a position of a point on the image

```python
def perspective_projection(d):
    """ Compute the image-plane \( z = -d \) position of a point expressed in camera coordinates """

def project(point):
    return (d*point[0])/-point[2], (d*point[1])/-point[2]

return project
```
The last rendering stage maps image-plane positions onto a rectangular array of pixels

```python
def image_to_array(size):
    
    """ Map image-plane coordinates onto integer indices in a rectangular array """
    def itoa(point):
        return size[0]*point[0]+(size[0]*0.5),
        -size[1]*point[1]+(size[1]*0.5)
    return itoa
```
A computational model of a pinhole camera is a composition of modeling and rendering.

\[
\text{map(image_to_raster(size),}
\text{  map(perspective_projection(1.),}
\text{    map(move_shape((0., 0., -10.)), cube()))})
\]