ReadMe:

There're 2 project folder.
1. Project "Assignment2_xshu" is for PartI and PartIII
2. Project "Surface_of_revolution" is for PartII

You do not need to do any set up, just open the folder in IntelliJ and run the main().

1.
a) the scene is set up in function initObjects() and draw().
In initObjects(), I loaded 3 model files:
box.obj, cylinder.obj, top.obj(surface of revolution)

In draw(), I created 3 walls, a desk, a laptop and a dumbbell in sequence
It basically uses the stack of modelview, and combines with transformations(translate, rotate and scale)

b) lookAt() function in line 148
c) Trackball:
At first, I'm trying to solve it by changing the camera. Because it's easier to think of.
And soon, I have to face the problem of Gimbal Lock, and I spent several hours trying to solve it.
But it's too complicate in Math, so I tried to change the modelview instead of camera position.

The modelview accumulatively multiply the rotation matrix is the key to solve the problem:

```
trackballTransform =trackballTransform
        .rotate(delta_x, 0,1,0)
        .rotate(delta_y, 1, 0,0);

modelview.peek().mul(trackballTransform);
```

2.Surface of Revolution

A surface of revolution is a surface created by rotating around an axis.
e.g.Pottery. And it's top view should look like circles with different radius.
If you can find a direction to rotate it, and when you're rotating, it's side view
should be exactly same.

If you rotate a rectangle, you get a cylinder

If you rotate a triangle, you get a cone

Here’s how I made the top.obj:
1. make a cylinder
2. make a cone
3. combine them
(the other way is to make them together)

for cylinder:
initial 4 points to draw the rectangle
use rotate matrix to rotate these 4 point to get new positions

for cone:
initial 3 points to draw a triangle
use rotate matrix to rotate these 3 point x times to get new positions list

pseudo code:

```
//get a list of vertex
for(int i = 0; i < NUM_SLICE; i++){ //NUM_SLICE means rotate how many times
  for(each position in position list){
    add position to vertex list;
    matrix.transform(position); //to get new rotated positions
  }
}

for(){ // get a list of indices
    indices.add(…)
}

use vertex list and indices list to create Polygon mesh
export mesh to obj file
```