**Lab1**

There are 3 packages : graphic, GUI and helper

1. graphic:
2. the class Camera get the input of camera parameter

(c, pref, v\_, h, d, f, back-face culling)

and compute n, u and v

as well as M\_pers and M\_view

finally dot M\_pers and M\_view to get the final matrix M\_trans

1. the class Polygon records one polygon’s point number and point index

and it’s back-face culling or not

1. the class Geometry read the .d.txt and records all the points’ coordinate and polygons’ points’ indexes. When reading data, the class records the points’ range for the model’s location range, which will display in UI.

the function polygon\_bac compute every polygon's back-face culling.

the function project change the world coordinate to camera coordinate using the M\_trans

1. the class Core use the function draw to let the Class camera to compute matrix and let the Class Geometry to compute the back-face culling using camera’s input and camera’s n and then compute every node’s projection.

In order to fit in the JPanel’s draw coordinate, it also needs some computation to change the size [-1, 1][-1, 1] to screen size and change their direction.

Then use the JPanel to draw.

1. GUI
2. helper

1) the Compute class has a lot of static functions to compute matrix and vector and homogeneous coordinate.

**Lab2**

Add a new class Rasterization to scan and create and update z-buffer.

For every ymax, using (int) to cull.

For every line that is parallel to scan line, they are not added into the ET.

**Lab3**

Changed a lot for the computing for normal vectors needed for illumination normal N.

Use different methods in Core and Rasterization to apply constant, gouraud or phong.

New class Illumination can compute the illumination at one single point.

**Lab4**

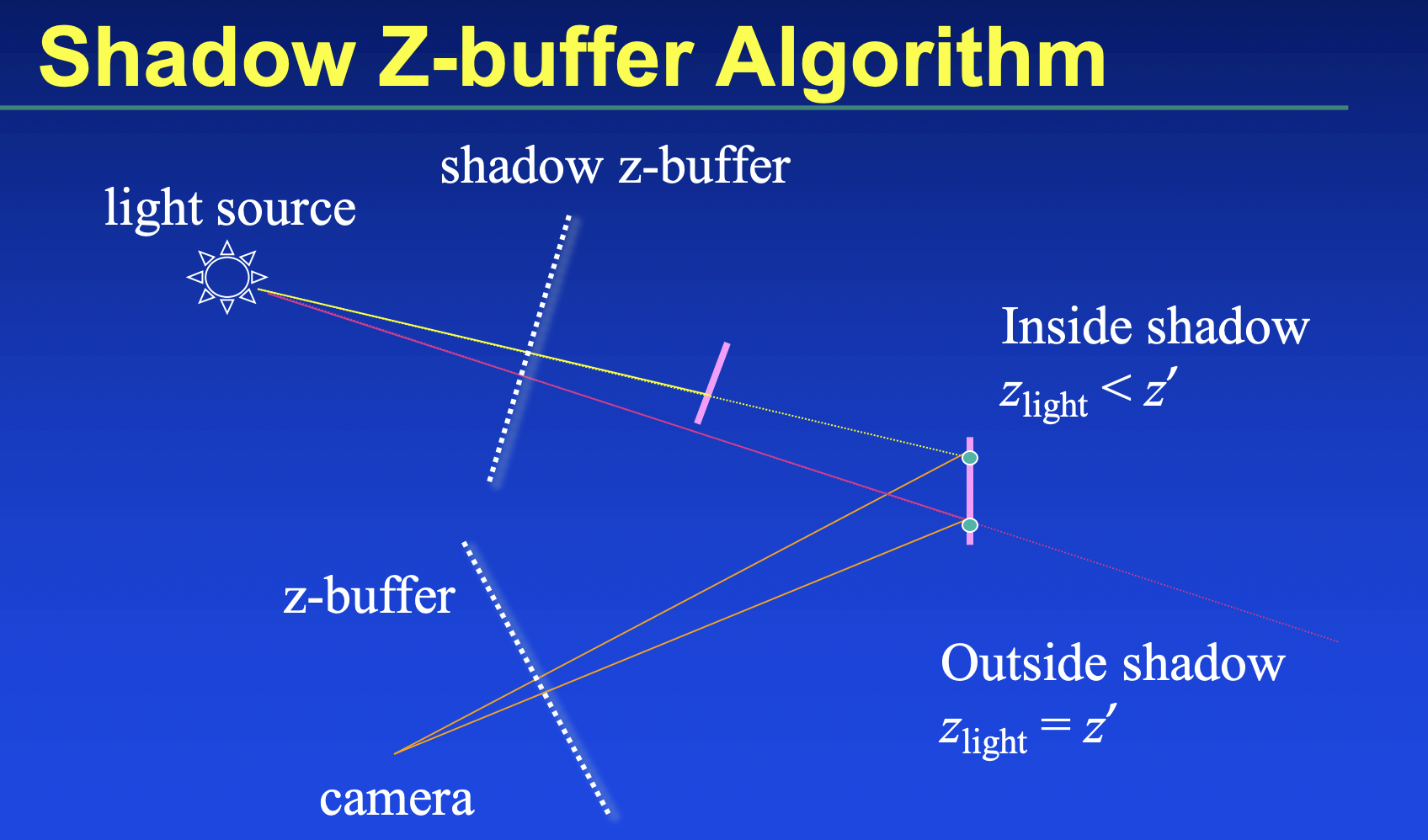
Create a new class Texture to read and map the texture.

Using the phong shading when using texture mapping.

Using simple mapping strategy, which is just using the pixel’s wordcoordinate position’s x and y.

**Lab5**

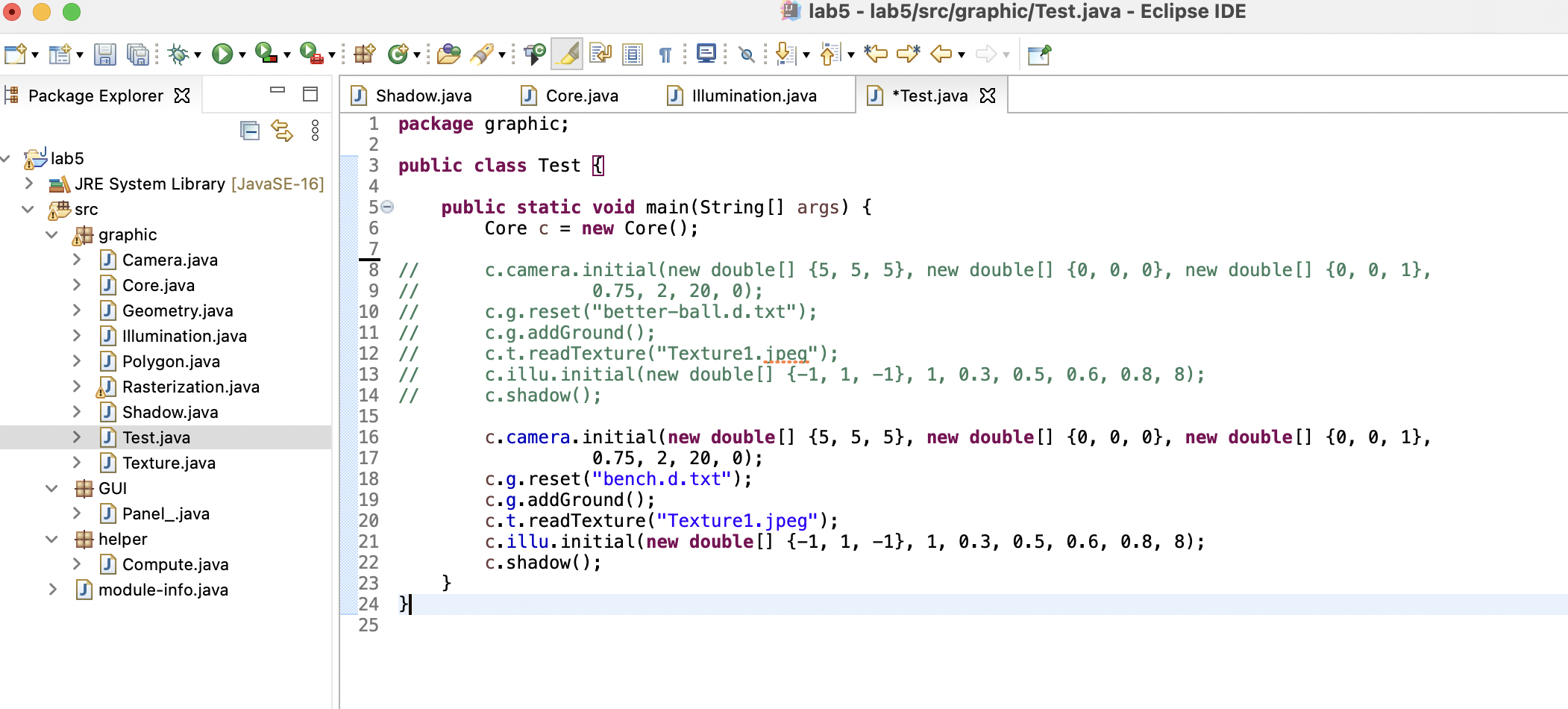
Add a new class shadow to implement the shadow z-buffer algorithm.



The picture is based on the previews labs’ result, adding a point light source that can be changed, and the previews parallel light is also existing.

Since the implement is not available in any situation or data, I didn’t update the front end.

So the entrance for drawing picture code is in Test.java.



And the data for point light source’s position can be changed in line 249 in Core.java.

