Prof. Dr. Daniel Huson
Algorithmen der Bioinformatik
Fachbereich Informatik
Mathematisch-Naturwissenschaftliche Fakultät

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Advanced Java for Bioinformatics

Summer 2025

Assignment 5

Due: 29-May-2025

In this assignment, we will write a basic 3D object viewer that shows a tripod representing the origin and the three main axes, and that can be used to open a file in OBJ format and display the 3D object.

Make sure that you adhere to the approaches discussed in the lectures. Follow the instructions closely. If you don't understand how you are supposed to do something, ask.

(If your secret strategy is to simply copy code from an AI without even trying to understand what the code is supposedly doing, then please note that "a fool with a tool is still a fool" and that this strategy will not get you very far.)

1 Basic 3D viewer (4 points)

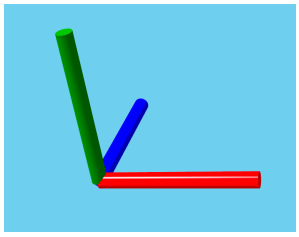
Setup a package `assignment5` that has the usual components:

- `ObjectViewer.java` – the main application entry point
- `window.Window.fxml` – the FXML layout file
- `window.WindowView.java` – constructs the view
- `window.WindowController.java` – provides access to the scene graph nodes defined in the FXML file
- `window.WindowPresenter.java` – connects the view to the model
- `model` package – contains all data structures, algorithms, and I/O (no JavaFX nodes)

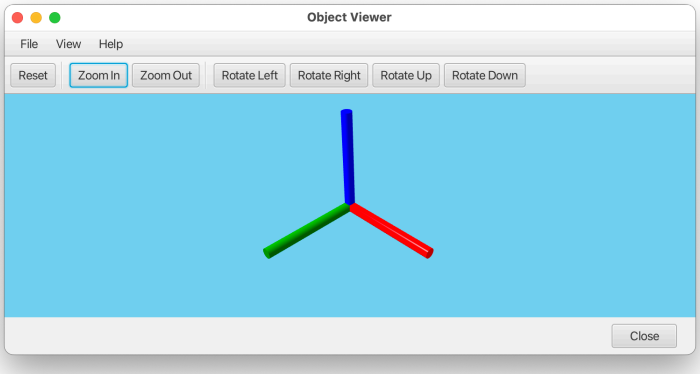
To show some initial content, in the `window` package, define a class `Axes` that has the following signature:

```
public class Axes extends Group {
  public Axes(double length) {
    ...
  }
}
```

This class should extend `Group` and contain three cylinders indicating the three main x, y and z axes, shown in red, green and blue, respectively. (For each cylinder, set the material up so that it has diffuse color red, green or blue, and specular color white.) The `length` parameter determines the length of the axes, call it with a value of 20, say, and the radius should be 5% of the length:



The viewer should look like this:



The reset button resets the zoom and rotation, while the other buttons are used to zoom in and out, and to rotate the object (relative to the screen, not relative to the object, e.g. rotate up always rotates up, independent of what the current orientation of the object is.)

The view menu should contain one corresponding menu item for each toolbar button, providing short cuts to allow zoom and rotation using the key board.

When setting up the view in FXML, place a `Pane` called `centerPane`, say, in the center of your border pane. When completing the setup in your presenter class `WindowPresenter`, setup a sub scene that has a group called `root3d` as its root node, and then add that sub scene to your center pane. Bind the width and height of the sub scene to the width and height of `centerPane` to ensure that the sub scene changes size when the window is resized.

Setup a perspective camera and add it to the sub scene (as discussed in the lecture).

Create a second group `contentGroup` and add it to your `root3d` group.

Place any objects to view in this group, such as your axes. When you implement the functionality of rotating, the corresponding transform should be applied to this group. Use a method like this to help implement rotation functionality:

```
private static void applyGlobalRotation(Group contentGroup, Point3D axis, double angle) {
  var currentTransform=contentGroup.getTransforms().get(0);
  var rotate = new Rotate(angle, axis);
  currentTransform = rotate.createConcatenation(currentTransform);
  contentGroup.getTransforms().setAll(currentTransform);
}
```

To implement zooming in and out, move the camera toward or away from the origin (in z-direction).

Add a point light and an ambient light, as discussed in the lecture.

2 Displaying objects from OBJ files (3 points)

The OBJ format is a simple, text-based file format used to represent 3D geometry (file suffix: `.obj`). It was developed by Wavefront Technologies and is widely supported in 3D modeling and graphics tools.

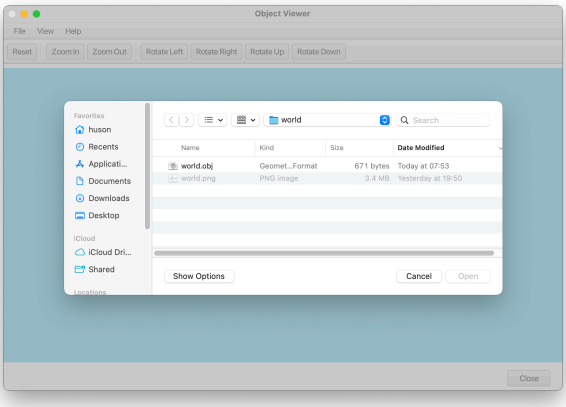
It describes:

- Vertices, texture coordinates, normals
- Faces as collections of vertex indices (with optional texture and normal indices)
- Optionally, materials (.mtl files)

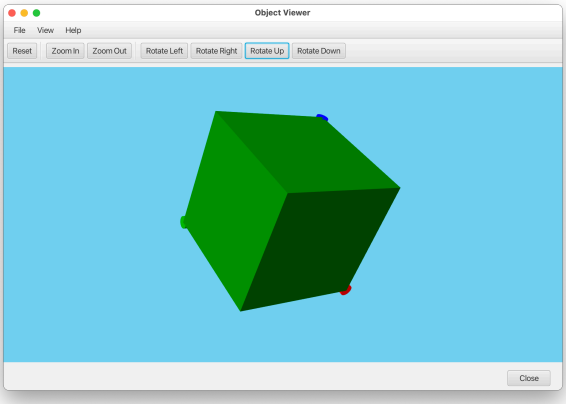
Use the provided class `ObjIO.java`, but take a look at the code so that you can explain how it works. Make sure you put it into the correct package (which one is that?) (JavaFX only supports triangular

meshes, however, the OBJ format also allows other meshes, so when parsing OBJ format non-triangular faces are triangulated.)

Add an `Open...` item to your file menu that allows the user to select an OBJ file:



If a file is selected, then parse the file, setup the triangular mesh and show it using a mesh view. (Use green, say, for the diffuse color of the mesh view and white for the specular color, as discussed in the lecture.)



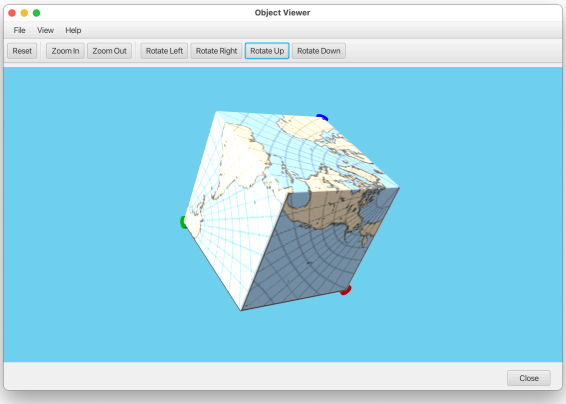
3 Discuss the Earth OBJ file (1 point)

Take a look inside the provided file `earth.obj` and explain what the `v`, `vt`, `vn` and `f` entries represent, and why:

- the number of `v` entries is 8,
- the number of `vt` entries is 14,
- the number of `vn` entries is 6, and
- the number of `f` entries is 12?

4 Texture mapping (2 points)

Modify your code so that, when opening an OBJ file, you check whether a file with the same path ending on `.png` exists, and, if it does, use the image texture mapping (as discussed in the lecture). This should work for the pair of provided files `earth.obj` and `earth.png`:



Note that 29-May is a public holiday, so there will be no lecture on 29-May. However, there will be a new assignment sheet (Assignment 6).