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

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Advanced Java for Bioinformatics

Summer 2025

Assignment 6

Due: 5-June-2025

In this assignment, we will improve upon the basic 3D object viewer that we implemented in the previous assignment (5).

As usual, create a new package, `assignment6`, and copy over any files that you want to reuse. Do not call classes from previous assignment packages.

As usual, 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.

1 Copy and mend your basic 3D viewer (1 point)

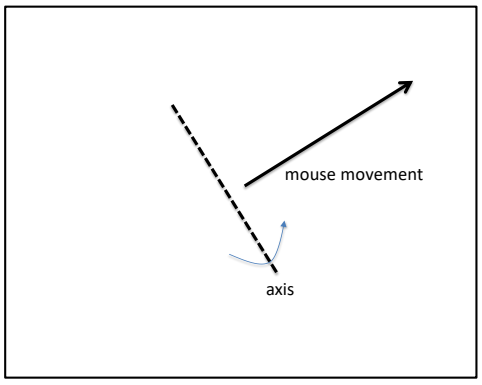
Setup a package `assignment6` that has the usual components:

- `ObjectViewer6.java` – the main application entry point (use "6" to avoid confusion)
- `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)
- `model.ObjIO.java` - -class for importing an OBJ file.

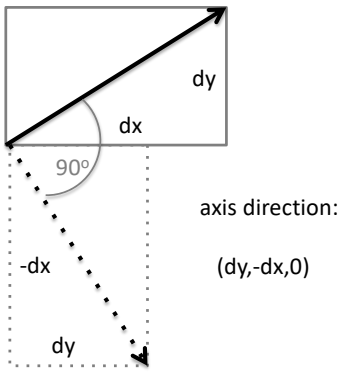
This should have all the functionally requested in Assignment 5. (Complete and fix missing features.)

2 Mouse drag interaction (2 points)

Our aim is to implement interactive rotation of the displayed figure using a mouse. What we want is that the figure rotates following the mouse, that is, around an axis that is orthogonal to the direction of the mouse movement:



It is easy to figure out the coordinates of the axis:



To implement rotation by mouse, implement a class like this, using the method `applyGlobalRotation()` from last week's assignment to perform the rotation:

```
public class MouseRotate3D {
    private static double xPrev;
    private static double yPrev;

    public static void setup(Pane pane, Group figure) {
        pane.setOnMousePressed(e -> {
            xPrev = e.getSceneX();
            yPrev = e.getSceneY();
        });

        pane.setOnMouseDragged(e -> {
            var dx = e.getSceneX() - xPrev;
            var dy = e.getSceneY() - yPrev;

            var axis = ...
            var angle = ... // based on the distance of the mouse movement

            ...

            xPrev = e.getSceneX();
            yPrev = e.getSceneY();
        });
    }
}
```

Explain how this works.

3 Mouse scroll interaction (2 points)

We want to use the mouse scroll gesture (method `setOnScroll(e)`) to reposition the camera. Note that the argument `e` has methods for determining how much the user has scrolled by in X and Y direction (`e.getDeltaX()` and `e.getDeltaY()`).

Please implement the following cases:

- If the mouse is scrolled and the shift key is *not pressed*: translate the camera in Z-direction by the value scrolled in Y direction. This implements "zooming in" and zooming out".
- If the mouse is scrolled and the shift key is *pressed*, then translate the camera in X- and Y-direction appropriately.

If your device does not support scrolling in X and Y direction, then, alternatively, extend your code for the previous task so to move the camera in Z direction if the *control key* is pressed while dragging the mouse, and to move the camera in X and Y direction when the *shift key* is pressed while dragging the

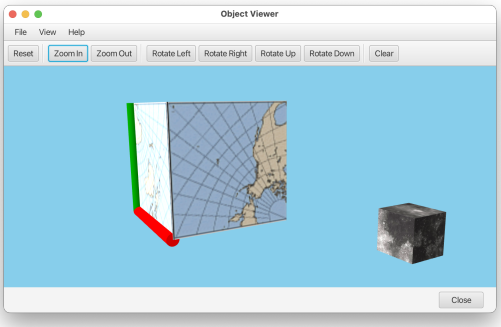
mouse. (Note that in this case, you will have to also pass the camera as a parameter to the above `setup` method.

4 Viewing multiple objects (2 points)

The main aim of the project part of the course will be to implement a fully featured anatomy viewer.
To prepare for this, we need to allow the viewer to load meshes from multiple OBJ files.
In more detail:

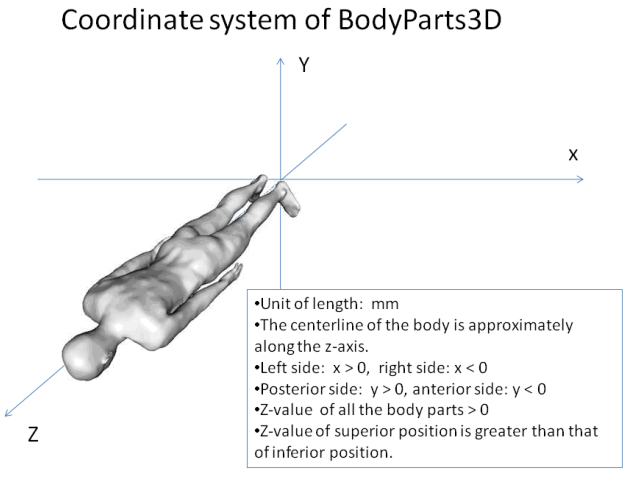
- Modify your `Open File` dialog to allow the user to select multiple OBJ files.
- If the user uses the `Open File` dialog to open more OBJ files, then create and create and show one mesh view per OBJ file and add them all to the scene.
- Implement a `Clear` menu item and tool bar button to remove all currently showing mesh views.

To debug this, use the two provided files `earth.obj` and `moon.obj`.



5 Centering the group of mesh views (3 points)

The anatomy data that we are going to use in the project is based on the following coordinate system:



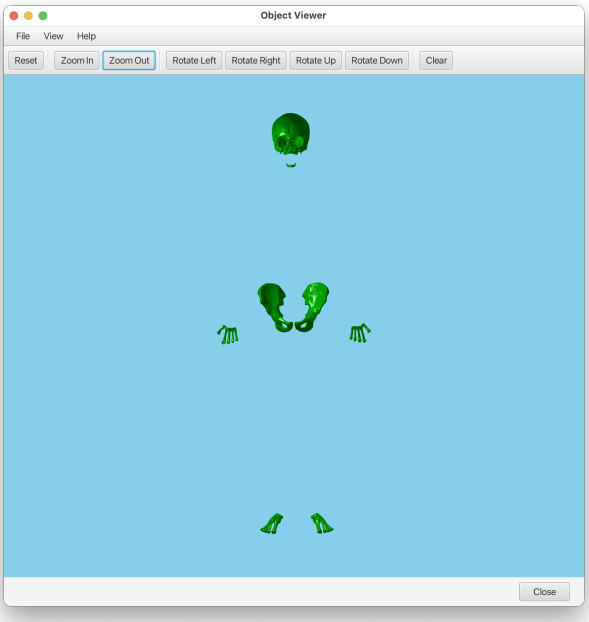
For the viewer to display things well, we need to center the set of created mesh views as a group.
To address this, introduce a new group called `innerGroup`. Place it inside your `contentGroup`. Now, when adding mesh views to the scene graph, place them into the `innerGroup`, not the `contentGroup`.
(It is important that your controls for rotating the figure continue to transform the `contentGroup`, not the new `innerGroup`.)

When the list of children of the `innerGroup` changes, we want to apply a translation to the `innerGrouop` so that the group is centered, using code like this:

```
innerGroup.getChildren().addListener((InvalidationListener)e->{
  var bounds = innerGroup.getBoundsInLocal();
  var centerX = (bounds.getMinX() + bounds.getMaxX()) / 2.0;
  var centerY = (bounds.getMinY() + bounds.getMaxY()) / 2.0;
  var centerZ = (bounds.getMinZ() + bounds.getMaxZ()) / 2.0;
  innerGroup.getTransforms().setAll(new Translate(-centerX, -centerY, -centerZ));
});
```

Explain how this works.

Apply your program to the collection of objects in the `bones` directory to get an image as shown here:



Please note: The contents of Assignment 5 and Assignment 6 are prerequisites for the project, so it is important that you get things to work and understand how they work.