Programming Assignment 2: User Interaction and Shaders
CMSC427: Computer Graphics, Fall 2020

Submission deadline: Friday October 16st[nd], 11:59pm

In this assignment you implement functionality for user interaction with your scene via a virtual trackball. Continue to build on the Java code that you got to know in the previous assignments. The deadline to submit your solution via ELMS/Canvas is Friday October 16st[nd], 11:59pm. Please submit only the *Java files that you modified*. Grading will take place in a meeting with the teaching assistant, for which you need to register via the list on ELMS/Canvas.

1 Virtual Trackball (4 Points)

Implement a virtual trackball that you can use to rotate an object with the mouse. Your solution should transform mouse movements with clicked mouse button (dragging) into a rotation matrix, which can then be applied to rotate the scene accordingly. Rotations around all three coordinate axes should be possible.

You can obtain information about mouse positions during dragging by implementing the `MouseMotionListener` interface, see also the documentation here. Follow the example of `SimpleMouseListener` in `simple.java` when implementing this interface.

The figure below shows how to derive a rotation axis and a rotation angle from a mouse movement. The symbols m_0 and m_1 denote two consecutive 2D mouse positions. First, you obtain two 3D points v and w by projecting these positions along the z-axis onto a "virtual sphere" that fills the rendering window. Use the cross product $a = v \times w$ as a rotation axis and the angle between v and w as a rotation angle. Note that the *Matrix4f* class in the *javax.vecmath* package provides functionality to obtain a rotation matrix from an axis-angle representation. In this way, horizontal movements in the center of the window should lead to a rotation around the y-axis. Vertical movements in the center of the window should lead to a rotation around the x-axis. Movements at the boundary of the window (horizontal and vertical) should lead to a rotation around the z-axis.

Do not forget to handle the following special cases:

- The mouse position lies outside the virtual trackball (in a corner of, or outside the rendering window).
- The rendering window is non-square.

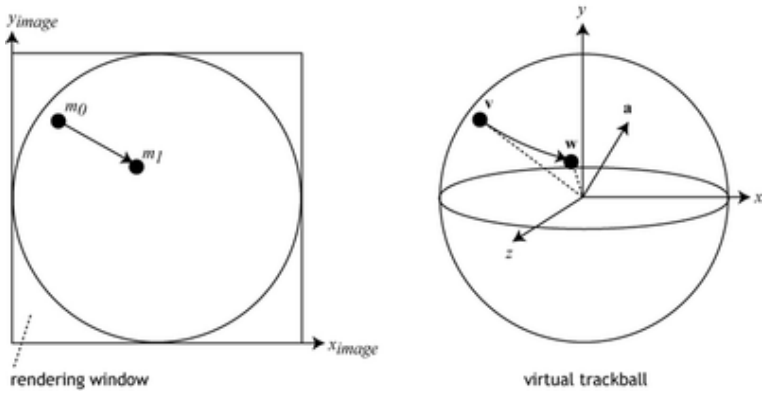

Figure 1: Visualization of the virtual trackball. While the mouse is dragged, two consecutive mouse positions m_0 and m_1 (left) are projected onto a virtual sphere (right) along the z-axis, resulting in vectors v and w that end on the sphere. These vectors define the rotation axis and angle.

Note that we also provide a more detailed description of a virtual trackball implementation from an external source with this assignment on ELMS/Canvas, which you may find helpful.

1.1 Test (1 Point)

You can run your virtual trackball with the objects and scenes from Project 1, but for this assignment please also test your implementation with triangle meshes that are read from files. You can use the class *ObjReader* that is included in the *jrtr* project. The object reader reads the *.obj* file format, a simple text-based file format to store polygon meshes. These *.obj* files basically stored a list of vertices (rows starting with v, one vertex per row) and a list of indices of the vertices of all polygons (rows starting with f, one polygon per row). Additionally, normals (rows starting with vn) and texture coordinates (rows starting with vt) can be stored. You can find more details concerning the *.obj* format on Wikipedia and in the obj format specification. The subfolder *obj* of the provided basecode contains some test files. It is recommended to test the trackball with the file *Teapot.obj*.