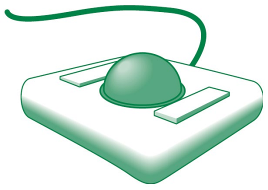
Worksheet 10: Virtual trackball

Reading	Henriksen et al.: Virtual Trackballs Revisited. *IEEE TVCG*, 2004. RTR: 4.2-4.3
Purpose 	In this worksheet, we return to user interaction. Our goal is to implement interactive orbiting, dollying, and panning of the camera. Orbiting is moving spherically around the scene center. Dollying is moving radially to have the camera closer to the center or further away from it. Panning is moving the camera in a plane parallel to the image plane. These three interaction modes are typical for a trackball, but this is not a commonly available tool, so we implement a virtual trackball and attach it to the mouse. The virtual trackball can equally well be attached to a touch interface.
Part 1 Simple orbiting	Pick one of your previous solutions where you draw in 3D. Set mouse events that modify your view matrix (by rotating the eye point) when the mouse is moving while a mouse button is down.
Part 2 Quaternion rotation	Switch to using quaternions for the orbit rotation instead of Euler angles. In this way, you can avoid the gimbal lock. Get x- and y-coordinates for your mouse position that are in $[-1,1]$. Project these coordinates to a spherical surface (blended with a hyperbolic surface). Normalize the resulting vector and build a quaternion that rotates from this vector to the previous one to find the camera orbit rotation corresponding to the mouse movement. To solve this part, you need a quaternion math library. This is available in `quaternion.js`, which is available on DTU Learn.
Part 3 Dolly and panning	Set up interaction modes (using either webpage buttons or mouse buttons) for orbiting, dollying, and panning. Store distance to the eye point from the look-at point and the xy-displacement of the look-at point together with the quaternions used for orbiting. In dolly mode, the difference in the y-coordinate of the mouse position is used to update the distance to the eye from the look-at point. In panning mode, the differences in x- and y-coordinates of the mouse position are used for displacement of both look-at and eye points along the world space basis vectors of the image plane.
Part 4 Spinning	To do spinning, continue to update the quaternion rotation of the view. Do this using another quaternion representing the last incremental rotation recorded in the `mousemove` function. Stop the spinning by resetting the incremental rotation to an identity quaternion when the mouse is released without a movement recorded in the `mousemove` function for more than 20 milliseconds.