CS-AD-216: Foundations of Computer Graphics

Assignment 5, Due: October 18

Instructions:

- Assignments can be submitted in groups of at most three. The purpose of groups is to learn from each other, not to divide work. Each member should participate in solving the problems and have a complete understanding of the solutions submitted.
- Submit your assignments as a zip file (one per group).

Problem 1 (Matrix for rotation around an axis)**.** [10 points]
We have proved that if a vector $\vec{v}$ is rotated around the axis $t\hat{u}$ (where $\hat{u}$ is a unit vector) in the counter-clockwise direction (as seen from a point $o + s\hat{u}$, $s > 0$ where o is the origin) by an angle θ, then $\vec{v}$ is transformed to the vector $\vec{v}' = \vec{v}\, cos\theta + (1 - cos\theta)(\vec{v} \cdot \hat{u})\hat{u} + sin\theta\,(\hat{u} \times \vec{v})$. Derive the 3×3 transformation matrix corresponding to this linear transformation. You may assume that $\hat{u} = (u_1, u_2, u_3)$. The entries of the matrix should depend only on u_1, u_2, u_3 and θ, not on the components on v.

Problem 2 (Virtual Trackball)**.** [10 points]
The goal of this assignment is to implement a virtual trackball interface. For this we imagine an invisible ball of radius r (which can be set to 2 for example) around the origin and we map any point with coordinates (x, y) (recall that each of the coordinates are in $[-1, 1]$), to a point $p = (x, y, \sqrt{r^2 - x^2 - r^2})$ on the hemisphere with non-negative z-coordinate. When the user clicks the left mouse button we start tracking the mouse and if between two successive moments the mouse has moved so that on the hemisphere we have moved from a point p_1 to a point p_2, then we do a rotation that corresponds to moving p_1 to p_2 along a great circle while keeping the center of the ball fixed.

A skeleton code for the assignment is attached. The only function that you need to modify is `mousemove`. However, you do need to study the rest of the code carefully. There is a variable `vtM` which represents the current transformation matrix. It is initially set to a 4×4 identity matrix. The function `mousemove` is called every time the mouse moves. Between successive calls to this function, the mouse moves a little bit. This gives you the two points p_1 and p_2 from which you need to derive a rotation matrix R for this tiny movement. You then need to modify `vtM` appropriately and send it to the shaders. You can use the functions for computing the matrix for rotation around an axis and cross product from `MV.js`.

Problem 3 (10 points)**.** Prove that in any rotation matrix, the column vectors are of unit length and pairwise orthogonal.

Hint: consider how the matrix transforms the standard basis vectors.