Tutorial 7 (Week 9)

Dongho Kang (kangd@ethz.ch)

November 9, 2020

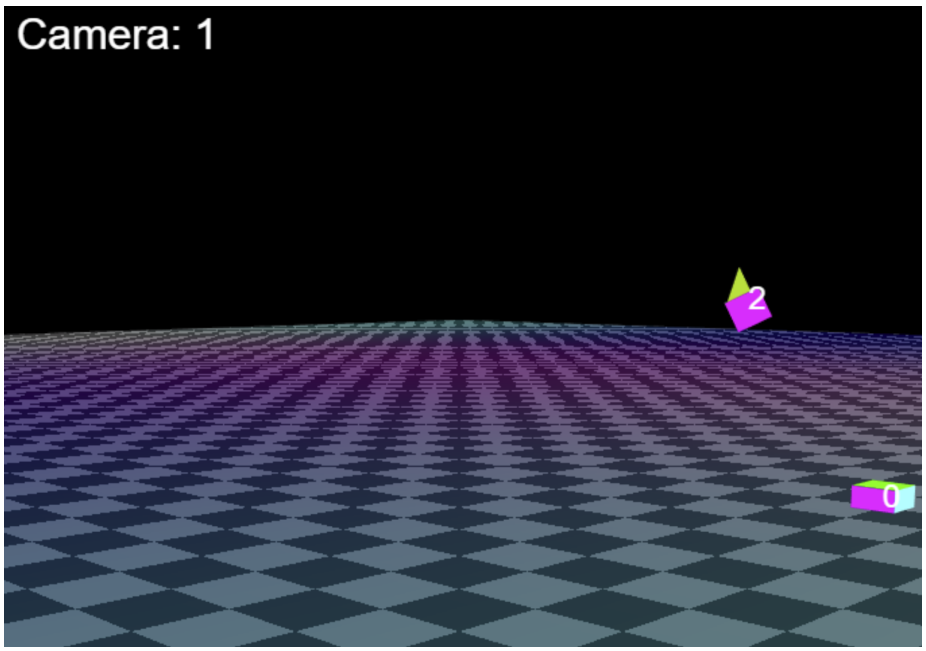

Figure 1: For extra credit make that checkerboard procedurally anti-aliased: `http://www.iquilezles.org/www/articles/checkerfiltering/checkerfiltering.htm`

Acknowledgement: Many thanks to James M. Bern who firstly wrote this tutorial instruction, slides and exercise code!

Introduction

The goal of this tutorial is to get real, real comfortable with transformations. NOTE: This code does **NOT** represent WebGL best practices.

Transformations in a nutshell

When in doubt, recall that for our purposes this is all just matrix multiplication.

Given point x, the equation

$$PVMx$$

means we first apply the model matrix M, then the view matrix V, then finally the projection matrix P. Note that the so-called *model-view matrix* is just the matrix VM.

Given camera matrix $C = TR$, where T is the camera's origin, and R is its orientation, the view matrix is

$$V = C^{-1},$$

where if you really want to be an overachiever, you can note that

$$C^{-1} = (TR)^{-1} = R^{-1}T^{-1} = R^{T}T^{-1},$$

since matrix rules and from the fact that R matrix is an orthogonal matrices (which means $R^{-1} = R^{T}$).

1 Basic transformations

First let's implement some basic transformations by hand (i.e. do **not** use **glmatrix**'s implementation of **rotate**, etc.)

So currently in addition to some other weird stuff, the code draws a triangle in the world's best shade of green. Modify the triangle's *model matrix* such that...

- The triangle is scaled $\cos(\text{globalTime})$ in the x-direction.
- The triangle is reflected across the x-axis.
- The triangle is rotated by globalTime about the origin.
- The triangle is rotated by globalTime about the point $(1, 1)$.

NOTE: Note that **mat4.multiply(C, A, B)** is equivalent to $C \leftarrow AB$ as you would (probably) expect.

NOTE2: However, I believe that e.g. **mat4.rotate(out, a, rad, axis)** means out $\leftarrow aR(\text{rad}, \text{axis})$, which to me at least was kind of unintuitive.

2 Hitch a ride on camera 2

By clicking you can switch which camera we're observing the scene from. (Note that all cameras are always looking at the origin.) Modify the triangle's model matrix such that instead of a big triangle at the world origin, we have a small triangle flying around on top of camera 2 as in Figure 1. Oh yeah, and make sure that the triangle spins in place while it flies on top of camera 2.

NOTE: Feel free to use the **mat4** calls for this section.

NOTE2: If you properly understand the code this section can be solved in around 5 lines.

3 Extra credit

- Watch this: `https://experiments.withgoogle.com/3-dreams-of-black`
- Implement the view matrix from scratch (gluLookAt)
- Check out pointerlock controls: `https://threejs.org/examples/misc_controls_pointerlock.html`
- Check out some cool video from YouTube channel 3Blue1Borwn:
  - `https://youtu.be/kYB8IZa5AuE`
  - `https://youtu.be/rHLEWRxRGiM`