CSE 167: Projection

Lecturer: Jurgen Schulze

Scribed by Owen Jow on October 11, 2018

1 The Future

Imagine a world with *layers*, toggleable layers, like Photoshop or HUDs, all existing on top of our physical reality. Imagine a world where you are the djinni Bartimaeus, switching between planes to see the wonders and monstrosities that others cannot. *Magic.*

Or – Magic *Leap*. An AR[1] goggles device, rendering on two planes in front of the scene. (Narrow FoV.) Pointer controllers to manipulate the augmented information. And imbued with the ultimate goal of rolling out such an AR layer system, networked so it can be shared by everyone, there or not there depending on whether you have the power to *see*. For the 4G network standard, video was the biggest driver. For 5G in the future, AR might take that mantle, with all of the 3D animations and lack of latency it will require when pulling information down to devices.

Let's do it! Let's augment the world everywhere. And when you have the goggles on, and you're on the right layer (tuned into the right frequencies), you will see your Pikachus or your AR OASIS. Just like everyone else, mind – everyone will be able to see the same things in the same places (unlike Pokemon Go, which is AR-wise currently an individual experience). Over the next couple of decades, worlds of layers might come up and AR-driven infrastructure might be built. All so we can tune into this virtual world existing parallel to the real world.

This is the future. At this very minute, a community is planning it all out...

2 Quaternions

We can describe 3D rotations using Euler angles, i.e. rotation about each of three coordinate axes, which is simple but suffers from potential Gimbal lock (where in certain configurations, rotation axes line up and thus do the same rotation, meaning an axis of rotation has been lost). This is bad e.g. in VR, where a hand/controller should be able to rotate however it wants.

One way to solve this is by rotating around an arbitrary axis, but this makes interpolation between orientations difficult.

So we turn to quaternions. A quaternion incorporates (1) angle a and (2) normalized axis of rotation nx, ny, nz into four coefficients w, x, y, z:

$$\begin{aligned} w &= \cos(a/2) \qquad & x &= \sin(a/2) \cdot nx \\ y &= \sin(a/2) \cdot ny \qquad & z &= \sin(a/2) \cdot nz \end{aligned}$$

[1]Augmenting the current world. By contrast, VR is "putting someone in another world."

For proper behavior, quaternions should be normalized. If normalized, the only source of redundancy in quaternions comes from positive and negative quaternions, which are the same.

The identity quaternion is $[w, x, y, z] = [1, 0, 0, 0]$.

On the whole, quaternions generally make everything to do with 3D rotations much simpler.

3 Projection

As far as we're concerned, projection refers to the act of transforming 3D points into 2D points. Typically, we will use either orthographic (parallel) projection, *e.g. when looking at CAD models for which we want parallel lines to stay parallel*, or perspective projection, *e.g. for increased photorealism.*

For orthographic projection, just drop the z-coordinate.

For perspective projection, we use a pinhole camera model in which we project points along rays that converge in the center of projection (i.e. the pinhole).

3.1 View Volumes

A view volume is the volume in the scene that is seen by the camera. It tells us which parts of the scene show up in the image (e.g. we might zoom out, making the viewable region larger, or zoom in, making the viewable region smaller).

Why? *In theory we could project everything in the world, and whatever ends up in the image ends up in the image. But in practice, it's an enormous waste to deal with all the objects that aren't visible. To avoid this, we need to know ahead of time what things could potentially appear.*

For perspective, we typically represent the view volume as a pyramid with angles dependent on FoV. In theory this pyramid could have infinite depth to look infinitely in the distance, but we don't do this, and limit the viewable depth using a *far plane*, because we want to have a reasonable accuracy for layering objects in the scene using the z-buffer. Similarly, we ignore close objects according to a *near plane*. And so our view pyramid becomes a view frustum.[2]

- A **general perspective view volume** has six parameters (in camera coordinates): left/right/top/bottom boundaries, and near/far clipping depths.
- If symmetrical, this reduces to four parameters: vertical FoV (basically the zoom angle: it's smaller when zoomed in and greater when zoomed out), image aspect ratio (determined by target window), and near/far clipping depths.

The fate of a camera view volume is to become a canonical view volume which extends from $[-1, 1]$ in each dimension. This serves as a unification of all projection methods; everything ends up as part of this "distorted 3D" cube (i.e. in NDC). Later, NDC are transformed into pixel coordinates according to a viewport transformation.

[2]**Frustum:** a pyramid with the top cut off.