CS 419: Production Rendering

Stereoscopy

Eric Shaffer
Stereoscopy

- Stereoscopy is the phenomenon of seeing in 3D
- Human binocular vision provides two views of a scene
  - Left eye and Right eye views are slightly offset
  - Known as Binocular Disparity
- We will discuss how to generate stereo pairs of a scene
  - Suitable for viewing in VR headset
So...some people do not like VR

- Stereo viewing a VR environment can be unpleasant
  - Immersive but inconsistent with real-world viewing experience
  - For example...accommodation
    - Changing focus of eyes to a different depth
    - In VR differs from real-world viewing
  - Other cues missing as well
    - e.g. visual movement without physical movement
Parallax

- To generate stereo pairs we need left and right cameras
- Separated by a distance called camera separation
- Each camera has a view plane
  - When a point is projected onto each plane
  - The two projections are displaced from each other in the direction of the line that joins the cameras
  - This is called parallax
Parallax

- Point is behind view plane relative to the camera
- Positive parallax
  - Left projection is left of right projection in the view direction
  - When rendered point appears behind the screen
Parallax

- Point is on view plane
- Zero parallax
  - When rendered point appears at screen depth
Parallax

- Point is in front of view plane relative to the camera
- Negative parallax
  - Left projection is right of right projection in the view direction
  - When rendered point appears in front of the screen
Parallax

- Is there a bound on positive parallax?
- Is there a bound on negative parallax?
**Camera Arrangements**

- **Toe-in**
- Left and Right cameras share a common look-at point
- Views differ in that object will appear rotated through angle $\beta$
When used with perspective projection toe-in yields vertical parallax
Example: Height of triangle will differ in the two images
...as will angle the edges make with horizontal
Can make visually fusing the images more difficult
Alternative: Each camera has own look-at point

Arranged so view directions are parallel
- Look-at points are same distance apart as the eye-points

The view-planes are coincident
- As are the viewports or "windows"

Requires the camera to use asymmetric view frustrums
Asymmetric View Frustum

- z-axis does not go through center of the window
- Defined by eye point and 4 corners
- Translate each viewport in the $x_v$ direction
  - Left camera $\rightarrow$ translate viewport in positive $x_v$ by half the camera separation
  - Right camera $\rightarrow$ translate viewport in negative $x_v$ by half the camera separation
The Stereo Camera - Implementation

- Stereo camera has
  - Eye point
  - Look-at point
  - Up vector
  - Camera separation

- Builds left and right cameras to generate the stereo pairs
Instead of specifying camera separation, can specify angle $\beta$

\[ x = r \tan \left( \frac{\beta}{2} \right) \]
Results

- Left-Right-Left stereo images

Viewing options:
- Synchronized viewing with shutter glasses
- Polarized projection and glasses (3D Imax)
- Parallel (Google Cardboard or Oculus Rift)