Review from last lecture

- Texture, Bump and Normal mapping
- Three types of optical distortion?
- How does texture mipmapping work?
Improving Latency and Frame Rates

- Example:
- Does a perfect system exist?
- Historical problems?
Reduction Methods

1. Lower the complexity
2. Improve rendering pipeline performance
3. Remove the delay from the path from the rendered image to switch pixels
4. Use prediction to estimate future viewpoint and world states
5. Shift and distort the rendered image to compensate for last-moment viewpoint, errors and missing frames
1. Simplify the virtual world

- Mesh simplification algorithms
- Lower the computational demands on the VWG (Virtual World Generator)
2. Improving rendering performance

- Better shading algorithms
- Perform rasterization for both eyes in parallel
- VR specialized GPU
3. From rendered image to switching pixels

Problem: Tearing

Vertical Blanking:

the time for the beam to move from lower right to top left.
Solutions to tearing

- **VSYNC**: sync VBLANK and rendered image loading time
- **Buffering**: Preload the frames into memory that is not being scanned for output to the display
- **Single buffering, Double buffering, Triple buffering, etc.**
Buffering
4. Prediction

Three questions:

1. At what future time will the pixels be switched to the next image?

2. Where are the objects (position & orientation) at that time?

3. Where will the user be looking at that time?
Render the future

Suppose we already know time when pixels will be switched, $t_s$

Ask the VWG to produce a frame for time $t_s$ and then perform visual rendering for the viewport at time $t_s$.

- The stimulus will be presented to the user at the exact time and place it is expected.
- “Zero effective latency” 😎😎😎😎

(All operations need access to a common clock!)
Predicting movement

- Static world vs Dynamic world
- Object’s movement in dynamic world is easy to predict. But human motion and the object's movement caused by human motion is hard to predict
- Use Momentum and inertia: Head has more inertia, finger has less inertia and harder to predict but head movement is predictable on the scale of 20ms.
5. Post-rendering image warp

- Rendered image is slightly larger than the display

- A simple transformation because of no time to perform complicated shading operations.
Post-rendering image warp
Flaws in the warped image

- Orientation changes (rotate head) are OK
- But positional changes (translate head) are incorrect!
- Visibility event: Part of an object may become visible only in the new viewpoint
Visibility event

Before image warp

After image warp
Solutions to the flaw

- Data structures: aspect graph & visibility complex
- Parallel processing to sample several future viewpoints and render image for all of them
Increasing the frame rate

- Inbetweening and tweening: insert warped frames for every rasterized image that is properly rendered

Example:

- 1 rasterized image per 100ms -> 10 FPS
- (1 rasterized image + 9 warped frames) per 100 ms -> 100 FPS
Texture mapping onto a virtual screen

- Each frame is treated as a photo that is texture-mapped to the mesh
Capturing a wider field of view

1. Taking multiple images by using one camera but in different directions
   - Image stitching (panorama)

2. Taking synchronized images by using multiple camera with fixed direction
   - Trade off: Numbers of camera vs optical distortion
Mapping onto a sphere

- From cube to sphere
- Six-sided CAVE projection system (Cube)
- Spherical projection theater (Sphere)
Perceptual issues

- What if different viewpoints are presented to left and right eyes?
- Can limited depth data improve the parallax and depth perception a lot?
- How do we perceive virtual objects (menu, avatar, etc.) inside of the sphere?
Review from today

- What are five solutions to increase frame rate?
- What is tearing? And how to solve it?
- Which one have less inertia? Head or finger?
- What are flaws of post-rendering image warp?
Announcements

● See you next week and keep your final project on track!

● Read Chapter 8.1 & 8.2