

LECTURE 8: VISUAL RENDERING

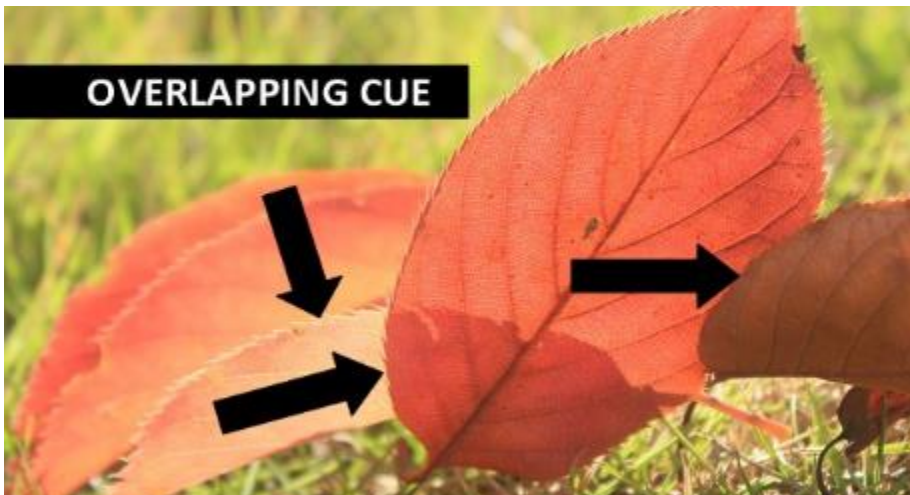
Ehsan Aryafar

earyafar@pdx.edu

<http://web.cecs.pdx.edu/~aryafare/VR.html>

Recall: Perception of Depth

- How do humans judge distance
 - In meters or in order (one object close than other)
- Depth/distance cues used by brain
 - **Cue: a piece of information derived from sensory stimulation and used for perception**
 - A depth cue by only a single eye movement is called monocular depth cue
 - Stereo depth cue: both eyes are required



Recall: Monocular cues may be enough for VR!

- We are bombarded by stereo and 3D displays for better depth perception
 - These technologies essentially feed a different picture to each eye
- **Drawback of feeding different left/right images: rendering cost increases**
- You do not need stereo images to perceive the world as 3D
- Billions of images/video on Google and YouTube, which give you immersive 3D experience by using Google Cardboard and other VR headsets
 - **Panoramic images and videos are already 3D enough due to wide field of view and sufficient monocular depth cues**

Recall: Perception of Motion

24: Sound and frame needed to be synchronized

FPS	Occurrence	
2	Stroboscopic apparent motion starts	
10	Ability to distinguish individual frames is lost	
16	Old home movies; early silent films	
24	Hollywood classic standard	
25	PAL television before interlacing	Black in between frames, caused flicker
30	NTSC television before interlacing	
48	Two-blade shutter; proposed new Hollywood standard	
50	Interlaced PAL television	
60	Interlaced NTSC television; perceived flicker in some displays	
72	Three-blade shutter; minimum CRT refresh rate for comfort	72: People sat closer to monitors
90	Modern VR headsets; no more discomfort from flicker	

Modern LCD and LED displays in smartphones, TVs, monitors support 60, 120, and even 240 FPS

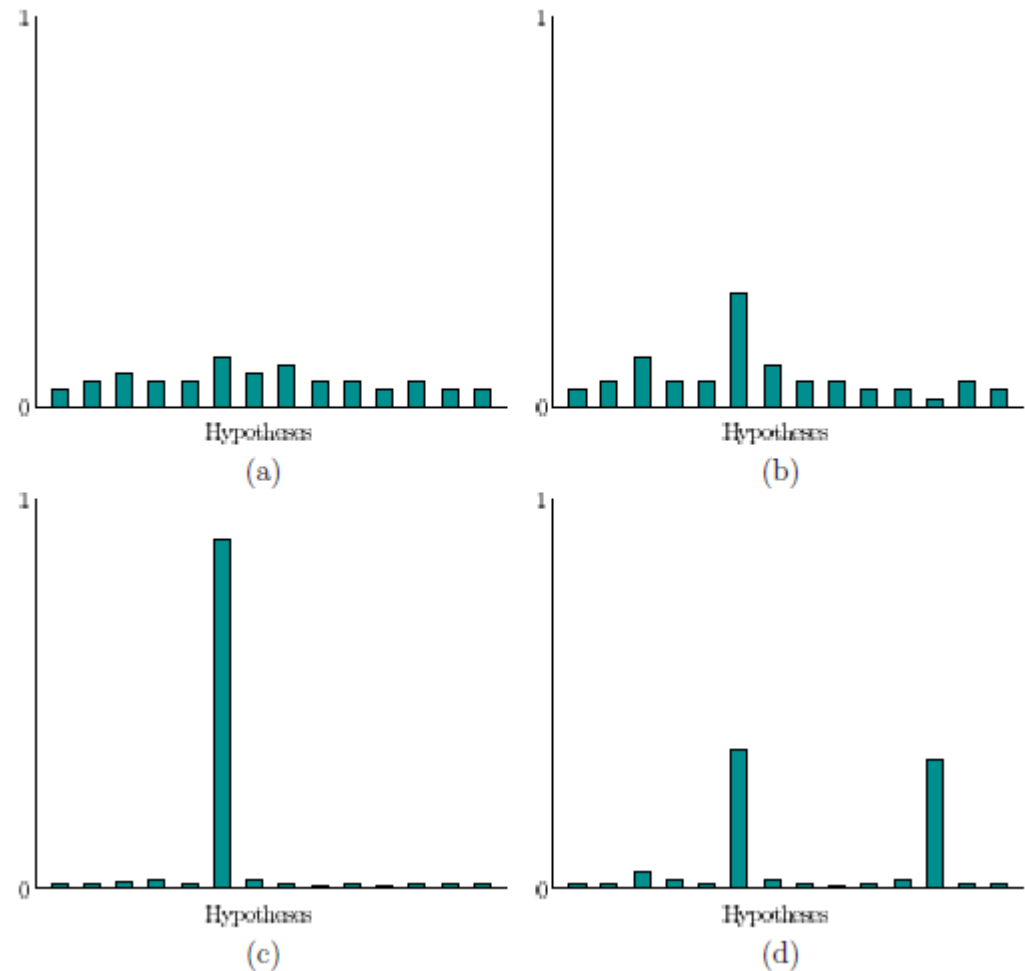
Figure 6.17: Various frame rates and comments on the corresponding stroboscopic apparent motion. Units are in Frames Per Second (FPS).

2 or 3 blade: show a single picture across 2 or 3 frames

Interlacing: show half of the image in one frame, and the other half in another

Recall: Combining Multiple Sources of Information

- Our perception system works by combining multiple cues
 - We also take into account prior cues
- Let Hypothesis be for example face of a person as we see the image
 - Cues could be for example, color of the eye, etc.
 - Ambiguity arises if two or more hypotheses have the same probabilities



This Lecture

- Visual rendering: what should the visual display show
 - Geometric transformations discussed where objects should appear based on rigid body, eye, canonical, and view port transformations
 - This lecture: how should objects appears based on light propagation, visual physiology, and visual perception

Rendering is the process of generating an image from a 2D or 3D model by means of a computer program!

Outline

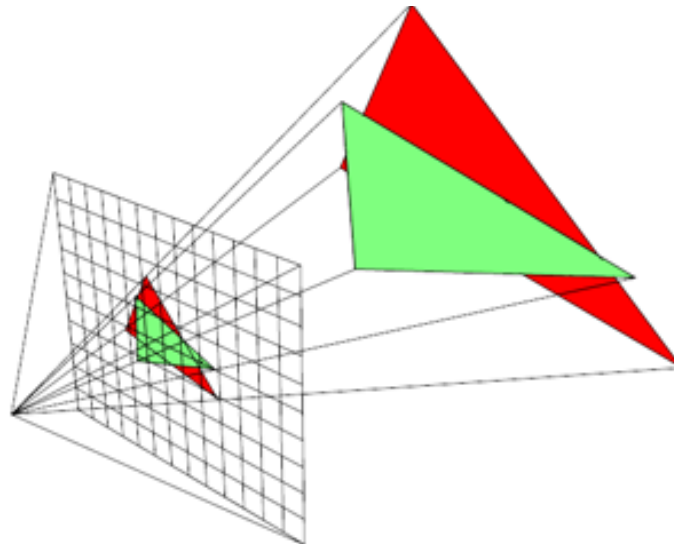
- What should be the light for each pixel
 - Based on light sources in the VR world and environment reflections
- Rasterization methods for solving rendering problems
 - Widely used in GPUs
- Rendering latency reduction
 - Important for great VR experience
- Rendering for captured virtual worlds
 - Going beyond synthetic VR worlds and objects
 - Example: panoramic photos and videos

What should be the light for each Pixel?

- Suppose the full chain of transformations is done
 - The location of every triangle is determined on the virtual screen
- Next step
 - Which screen pixels are covered by the transformed triangle(s)?
 - How to illuminate the pixels according to the VR world physics

Ordering Between Objects

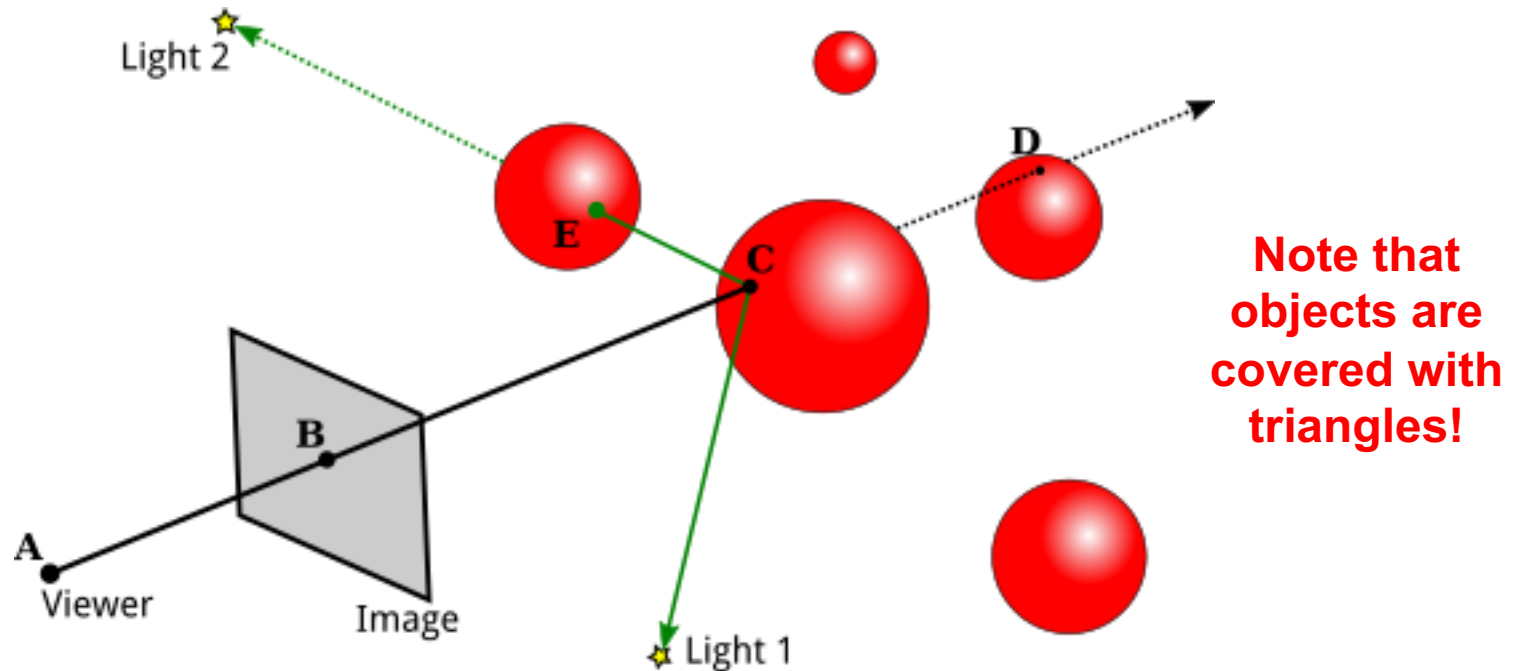
- For each pixel we need to answer:
 - Is the triangle even visible to the eye, or is it blocked by part of another triangle?
- Determine for any pair of points, whether the line segment that connects them intersects with any triangles
 - When intersection occurs, the line-of-sight (LoS) visibility between the two points is blocked



Rendering Loop

- Object-order rendering
 - Iterate over the list of all triangles and attempt to render each one-by-one onto screen
 - For each triangle on the screen FoV (field-of-view), iterate over the corresponding pixels
 - The pixels are updated only if the corresponding part of the triangle is closer to the eye than any triangles that have been rendered so far
- Image order rendering
 - Iterate over the pixels
 - Determine which triangle should affect its RGB values
 - Use Ray tracing to solve this problem

Ray Tracing

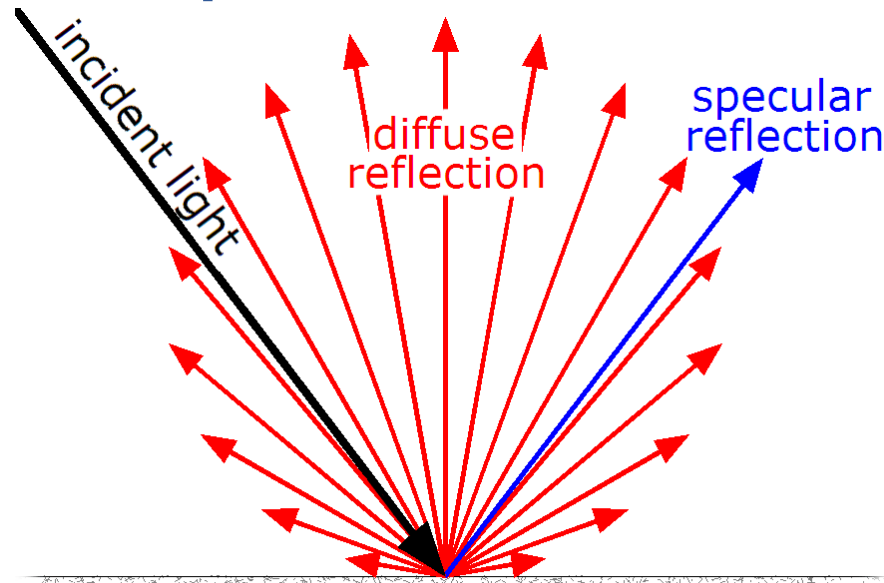


- **Ray casting:** the viewing ray is defined and its nearest point of intersection among all triangles in the virtual world is calculated
- **Shading:** the pixel RGB values are calculated based on lighting conditions and material properties of the intersection point

Ray Tracing

- Input: the eye location, pixel, and triangles' data structure
- Iterate over triangles
 - Determine if the ray intersects it
 - If it does, consider it as a candidate solution is closer than any other triangle considered so far
- Several optimizations to speed up the process
 - Algorithms for quickly finding intersection points
 - Store triangles in a 3D data structure that allows for quick elimination of some triangles, e.g.: Binary Space Partitioning (BSP) trees

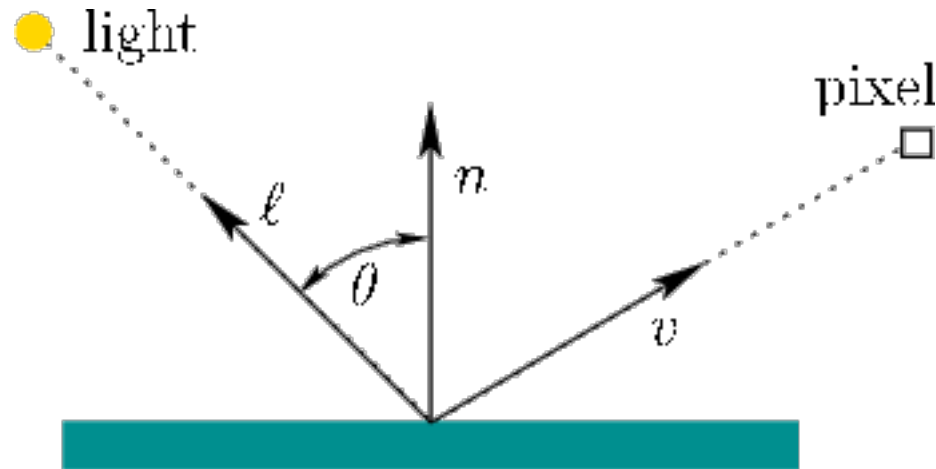
Diffuse vs Specular Reflection



Specular reflection is mirror-like reflection of waves

- Diffuse reflection is the reflection of light or other waves from a surface such that a ray incident on the surface is scattered at many angles rather than at just one angle as in the case of specular reflection. An ideal diffuse reflecting surface is said to exhibit Lambertian reflection, **meaning that there is equal luminance when viewed from all directions.**
- Diffuse scattering is due to roughness of the surface, specular reflection is due to the surface shininess

Lambertian Shading



- Light source simulates the real world light
 - Spectral reflection function: how obstacles reflect R, B, and G lights
- Assumptions about the spectral reflection model are captured by a shading model
- In the case of Lambertian shading, the angle that the viewing ray strikes the surface is independent of the resulting R, B, and G values
 - What matters is the angle light hits the source

Illumination Under Lambertian Shading

- A pixel under Lambertian shading is illuminated as

$$R = d_R I_R \max(0, n \cdot \ell)$$

$$G = d_G I_G \max(0, n \cdot \ell)$$

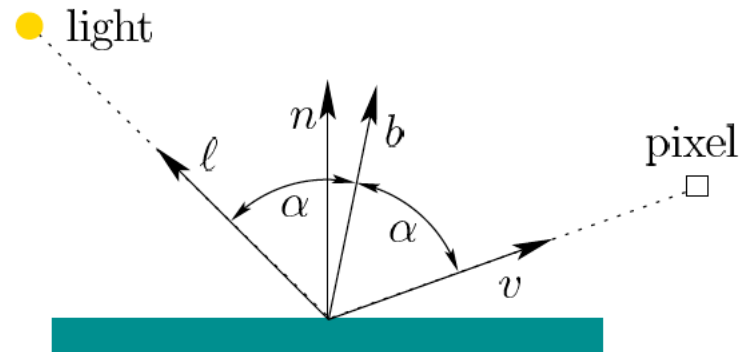
$$B = d_B I_B \max(0, n \cdot \ell)$$

$$n \cdot \ell = \cos \theta$$

Note: n and l are unit size vectors!

- d and I are the spectral reflectance (diffuse component based on the material) and light spectral power distribution (based on light)

Blinn-Phong Shading



- Models reflection due to object shininess
 - Second term of the formula
 - Assuming equal d and I values for R, G, and B
 - Shininess captured through an x parameter
 - Higher x (e.g., 1000) makes the object mirror-like
 - s is the spectral shininess (specular component)

$$L = dI \max(0, n \cdot l) + sI \max(0, n \cdot b)^x$$

Assume white object and light!

$$b = \frac{l + v}{\|l + v\|}$$

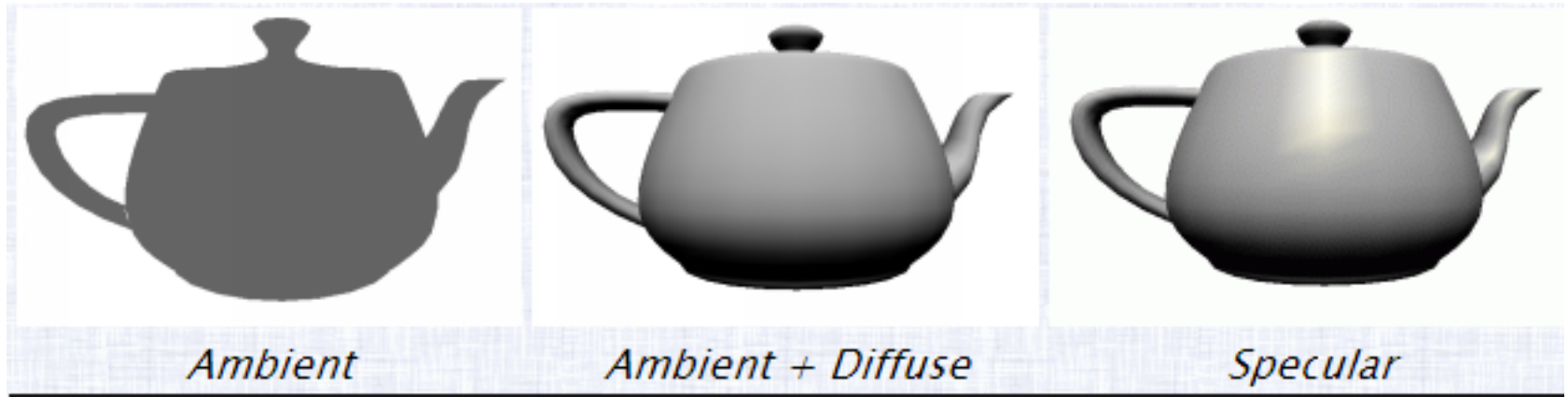
Ambient Shading

$$L = dI \max(0, n \cdot \ell) + sI \max(0, n \cdot b)^x + L_a$$

- Allow the object to glow without light source
 - Otherwise, shadows of all lights would be black
 - In real world, light can be reflected from objects multiple times
- L_a is the ambient light component
- In case the VR world has multiple light sources, all lights will add up at each pixel

$$L = L_a + \sum_{i=1}^N dI_i \max(0, n \cdot \ell_i) + sI_i \max(0, n \cdot b_i)^x$$

Ambient + Diffuse + Specular



Texture Mapping

- **Texture mapping** is a graphic design process in which a two-dimensional (2-D) surface, called a **texture map**, is "wrapped around" a three-dimensional (3-D) object. Thus, the 3-D object acquires a surface **texture** similar to that of the 2-D surface.
 - There are many types of texture maps: color, bump, displacement, normal

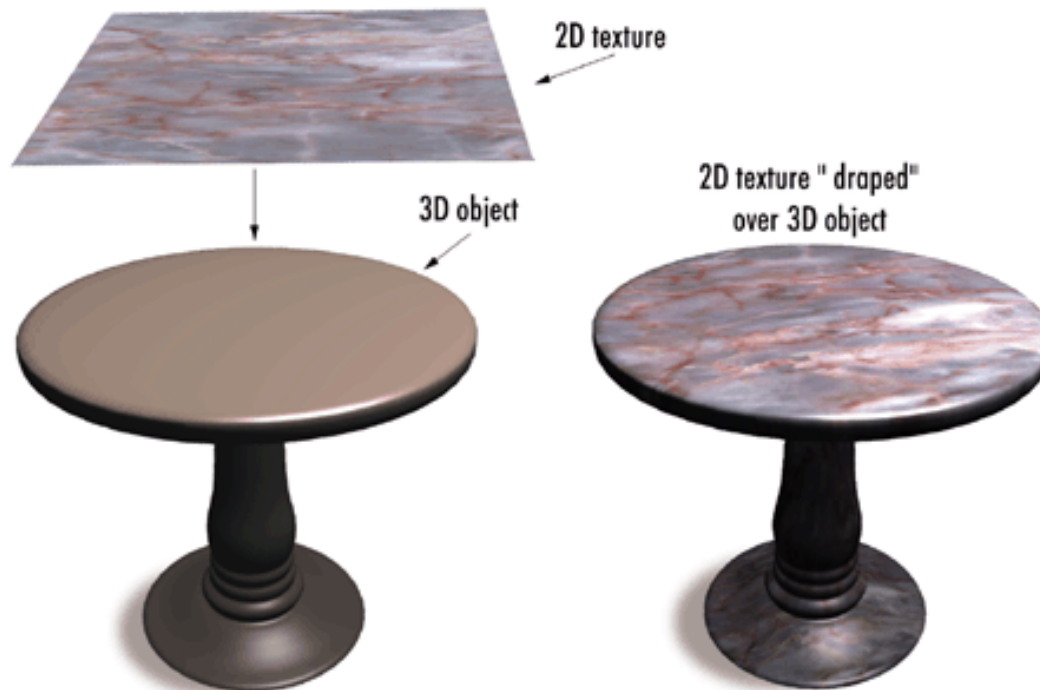


Image-Order Rendering Pseudo-Code

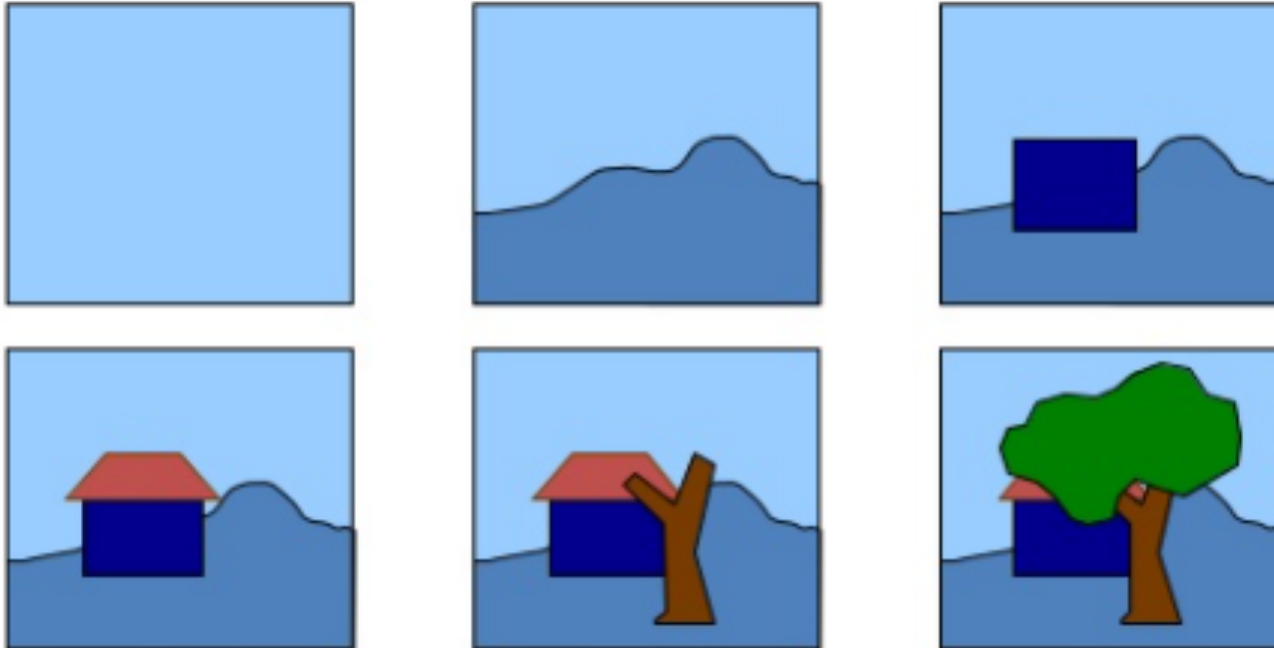
```
001  for (each pixel in image) {
002      Ray R = computeRayPassingThroughPixel(x,y);
003      float tclosest = INFINITY;
004      Triangle triangleClosest = NULL;
005      for (each triangle in scene) {
006          float thit;
007          if (intersect(R, object, thit)) {
008              if (thit < closest) {
009                  triangleClosest = triangle;
010              }
011          }
012      }
013      if (triangleClosest) {
014          imageAtPixel(x,y) = triangleColorAtHitPoint(triangle, tclosest);
015      }
016  }
```

Object-Order Rendering

- Ray casting can quickly become a bottleneck
 - More efficient to switch from image-order to object-order rendering
- For object-order rendering, we iterate over every triangle and attempt to color every pixel where the triangle lands
 - The process is called rasterization
- Formally, rasterization is the task of taking an image in a vector graphics format (shapes) and converting it into a raster image (a series of pixels or lines, which when displayed together, create the image originally represented as shapes)
 - The objects in our case are triangles
 - Rasterization is a key function of GPUs

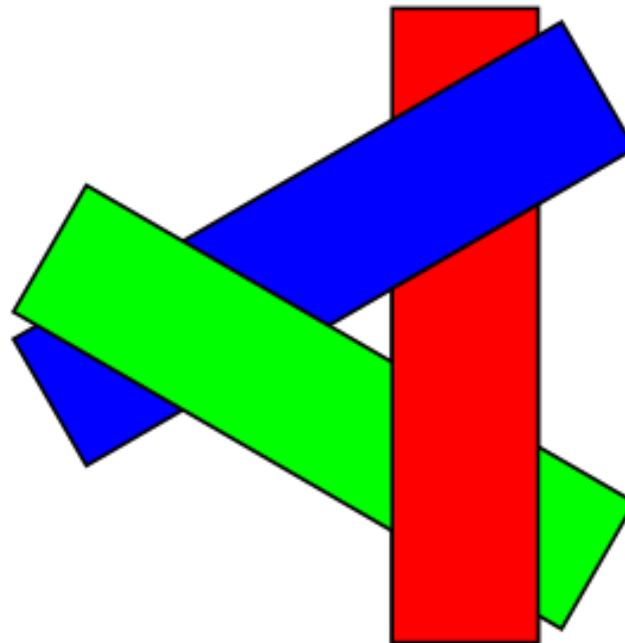
Painter's Algorithm

- Draw triangles from back (farthest away) to front (closest)
 - Sort triangles by their depths (z value)
 - Draw objects in order (farthest to closest)
 - Closer objects paint over the top of farther away objects



Depth Cycles

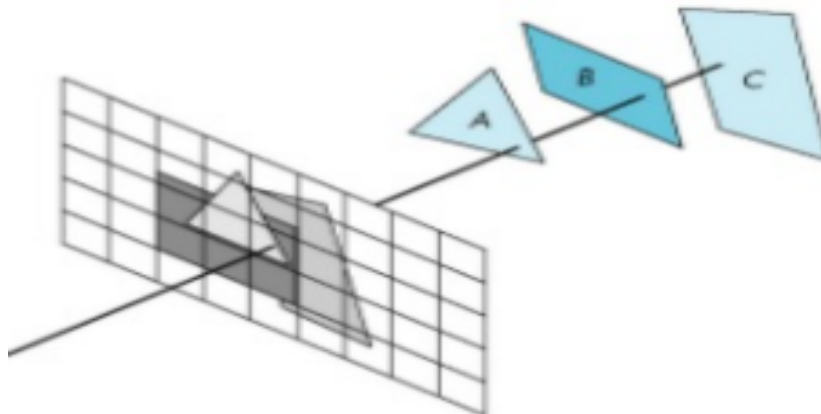
- One possible drawback of painter's algorithm, is existence of depth cycles



The solution is to store the depth on a pixel-by-pixel basis, by maintaining a depth buffer (z-buffer), which is typically normalized to be a value between 0 and 1

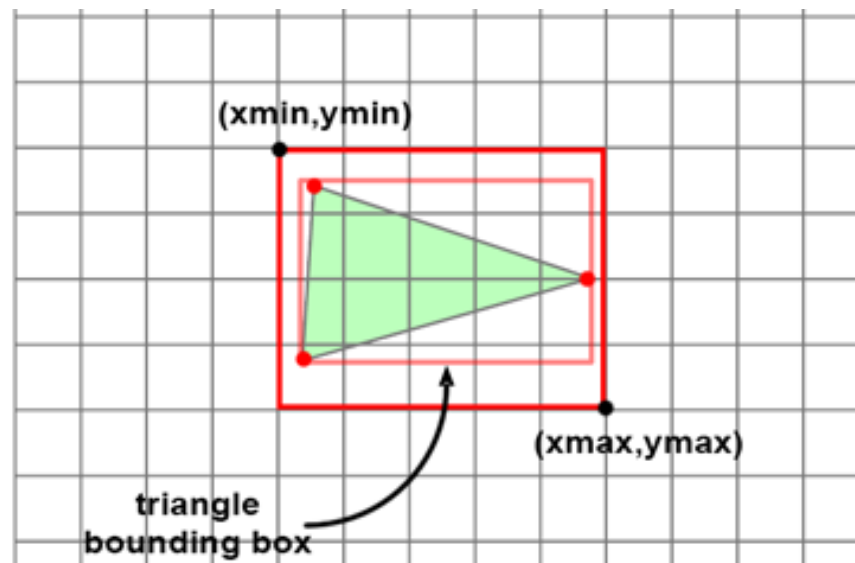
Depth-Buffer Algorithm

- Initialize the depth buffer and frame buffer such that for all pixel positions (x,y)
 - $\text{depthBuff}(x,y) = 1$, $\text{frameBuff}(x,y) = \text{BackgroundColor}$
- Process each triangle one at a time
 - **For each projected (x,y) pixel position of a triangle**, calculate the depth z
 - If $z < \text{depthBuff}(x,y)$, compute the surface color at that position and set
 - $\text{depthBuff}(x,y) = z$, $\text{frameBuff}(x,y) = \text{surfCol}(x,y)$



Rendering Triangles and Pixels

- Each triangle is rendered by calculating a rectangular part of the image that fully contains it: called a bounding box
 - The box is determined by transforming triangle vertices and finding the minimum i and j values (row and column indices)
 - A simple set of calculations is then conducted using barycentric coordinates to determine if a pixel is inside the triangle or not



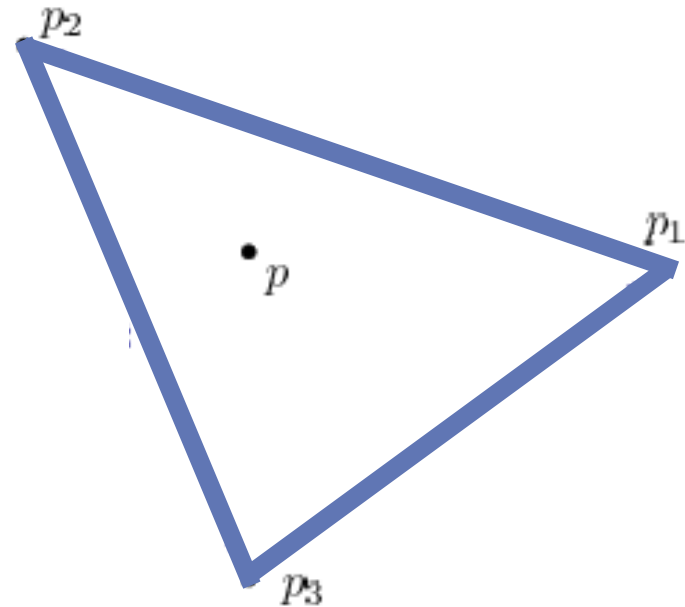
Barycentric Coordinates

- Barycentric coordinates are triples of numbers $(\alpha_1, \alpha_2, \alpha_3)$ corresponding to masses placed at vertices of a reference triangle. These masses then determine a point P , which is identified with coordinates $(\alpha_1, \alpha_2, \alpha_3)$. The vertices of triangles are given by $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$.

$$e_1 = p_3 - p$$

$$e_2 = p_2 - p$$

$$e_3 = p_1 - p$$



The same barycentric coordinates may be applied to the points on the model in \mathbb{R}^3 , or on the resulting 2D projected points. However, rasterization typically uses perspective projection to work on 2D (i.e., screen projected) triangles.

Barycentric Coordinates Continued

- Let for each combination of i and j we have

$$d_{ij} = e_i \cdot e_j \quad \text{dot product}$$

$$s = 1/(d_{11}d_{22} - d_{12}d_{12})$$

- Then, the barycentric coordinates are given by

$$\alpha_1 = s(d_{22}d_{31} - d_{12}d_{32})$$

$$\alpha_2 = s(d_{11}d_{32} - d_{12}d_{31})$$

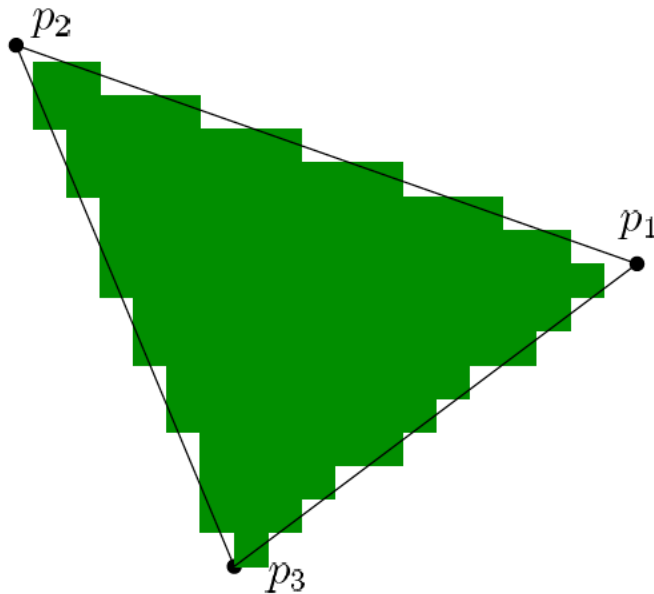
$$\alpha_3 = 1 - \alpha_1 - \alpha_2.$$

- A point P is inside the triangle if all conditions below are met:

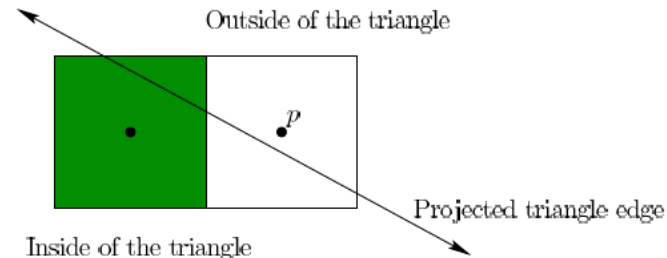
$$\alpha_1 \geq 0, \quad \alpha_2 \geq 0, \quad \text{and} \quad \alpha_3 \geq 0$$

Aliasing

- Staircase effect happens when only part of the triangle is inside a pixel (note we color based on the triangle center)
 - Due to insufficient pixel density



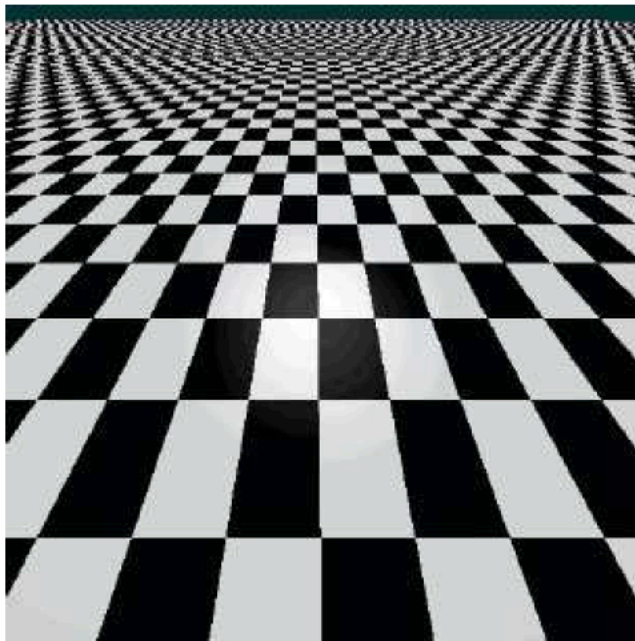
(a)



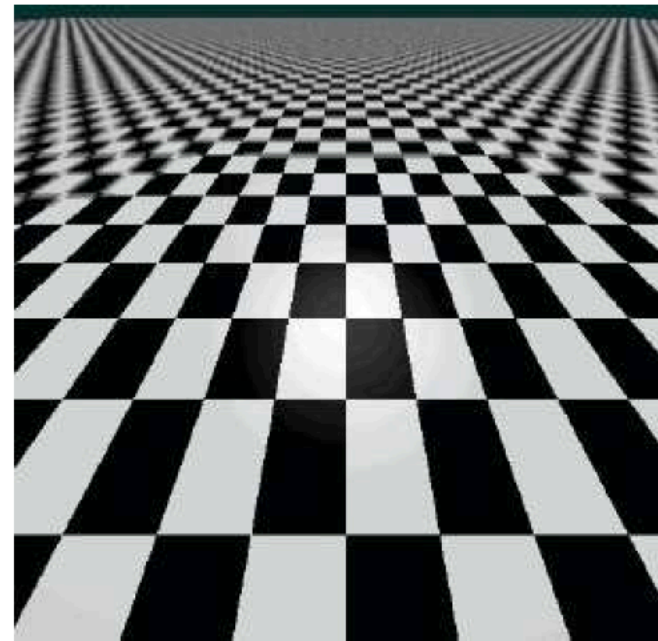
(b)

Super Sampling

- Combat aliasing by rendering a pixel based on the fraction of the pixel region covered by triangle
 - Color of the pixel can also be blended from multiple pixels



(a)



(b)

Object-Order Rendering Pseudo-Code

```
001 // A z-buffer is just an 2D array of floats
002 float buffer = new float [imageWidth * imageHeight];
003 // initialize the distance for each pixel to a very large number
004 for (uint32_t i = 0; i < imageWidth * imageHeight; ++i)
005     buffer[i] = INFINITY;
006
007 for (each triangle in scene) {
008     // project vertices
009     ...
010     // compute bbox of the projected triangle
011     ...
012     for (y = ymin; y <= ymax; ++y) {
013         for (x = xmin; x <= xmax; ++x) {
014             // check of if current pixel lies in triangle
015             float z; // distance from the camera to the triangle
016             if (pixelContainedIn2DTriangle(v0, v1, v2, x, y, z)) {
017                 // If the distance to that triangle is lower than the distance
018                 // z-buffer, update the z-buffer and update the image at pixel
019                 // with the color of that triangle
020                 if (z < zbuffer(x,y)) {
021                     zbuffer(x,y) = z;
022                     image(x,y) = triangle[i].color;
023                 }
024             }
025         }
026     }
027 }
```

Improving Latency and Frame Rates

- Motion to photon latency in a VR headset
 - The amount of time it takes to update the display in response to a change in head orientation and position
 - Head tracking delay
 - Rendering delay
 - If latency is too high, stationary objects would appear moving, and world would appear out of sync from what user expects
 - Motion sickness and VR fatigue
- In 1990s, 60 ms latency was considered acceptable
 - Latency was well over 100 msec
- Today, latency can be less than 20 ms
 - Even zero with predictive methods
 - Other sources of VR sickness: mismatches,vection, optical aberrations

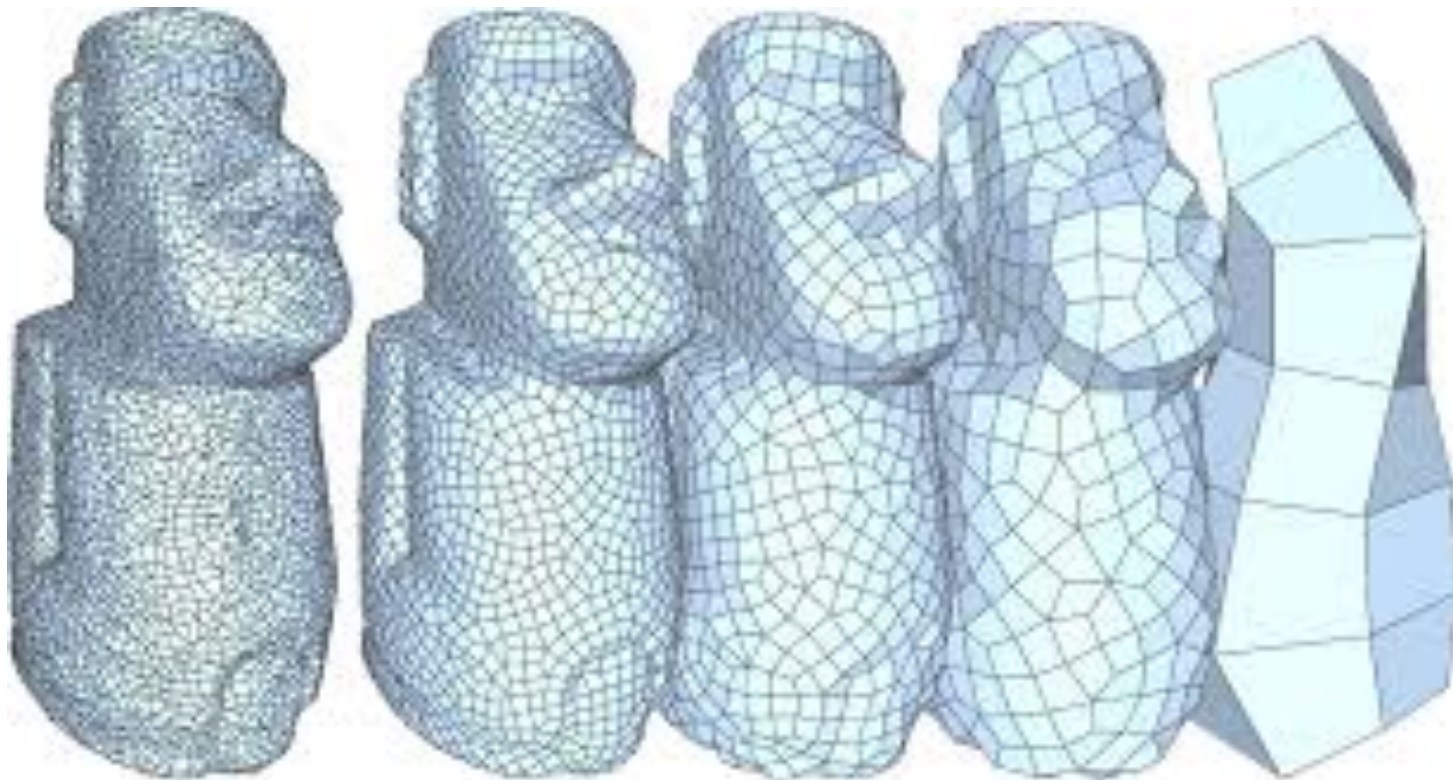
Overview of Latency Reduction Methods

- Reduce the complexity of the VR world
- Improve rendering pipeline performance
- Remove delays along the path from rendered image to switching pixels
- Use prediction to estimate future viewpoints and world states
- Shift or distort the rendered image to compensate for last-moment viewport errors

Simplify the Virtual World

- The chain of transformations and rasterization apply to each triangle, which increases the computation cost
 - Reduce the model by using less triangles!
- Why are models too big in the first place?
 - 3D scan model of the real world, has highly dense data
 - Similar to cameras, focused on dense and high quality representation
 - In Synthetic worlds, tools (such as Maya) automatically create a highly accurate mesh of triangles over a curved surface
- We can reduce model size by mesh simplification algorithms
 - Model should have sufficient details for all viewpoints in VR
 - In Unity 3D, you can use less number of material properties

Mesh Simplification



Virtual World Generator (VWG) Frame Rate

- VWG maintains a simulation of the virtual world
 - Moves all mobile geometric bodies and satisfies physical laws
 - Handles motions of avatars, object collisions, falling objects
 - VWG must maintain a coherent snapshot of the VR world each time a rendering request is made
 - **VWG has a frame rate similar to a display frame rate**
- Ideally we want the VWG frame rate to be as fast or faster than the rendering process, **and the to to be synchronized**
 - Whenever a new rendering request is made, a fresh VWG frame is available for rendering

GPUs Targeted for Improved VR Rendering Performance

Latest NVIDIA GPUs Get Foveated Supersampling Feature for Sharper VR Games


By Ben Lang - Jan 6, 2020  28

Image courtesy NVIDIA

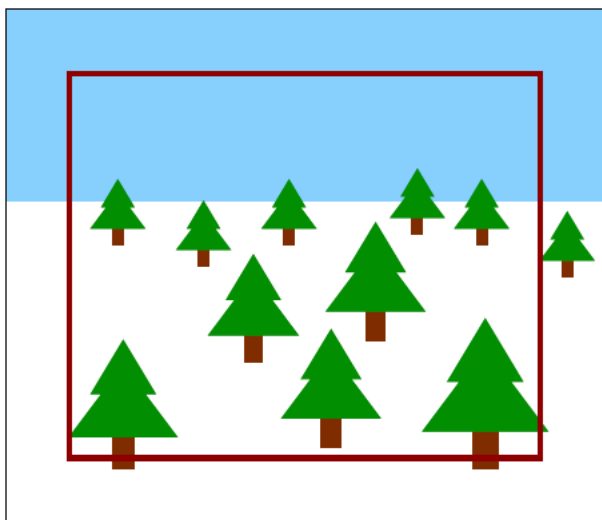
- VR gamers running NVIDIA's newest RTX graphics cards will be able to take advantage of a new 'Variable Rate Supersampling' (VRSS) feature designed to increase the sharpness of VR games without reducing performance. The feature uses a foveated rendering approach which focuses sharpness toward the center of the lens without wasting extra processing power toward the edges where the image will be blurred by the lens anyway.

Power of Prediction

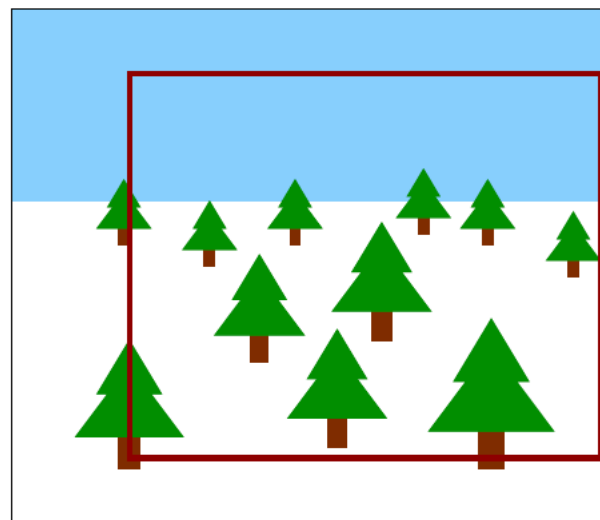
- If you can accurately predict the future (e.g., user look), you can eliminate the remaining latency
- It is possible to predict where your head will be in 20 ms
 - **The motions are tracked (methods will be covered later)**
 - You have no control over 20 ms granularity
 - **High momentum and inertia for head movement**
 - **Momentum:** Force that something has when it is moving
 - **Inertia:** Tendency to do nothing or remain unchanged
- Can be used to determine the future direction user would be looking at, and therefore the position and orientation of the virtual world

Post-Rendering Image Warp

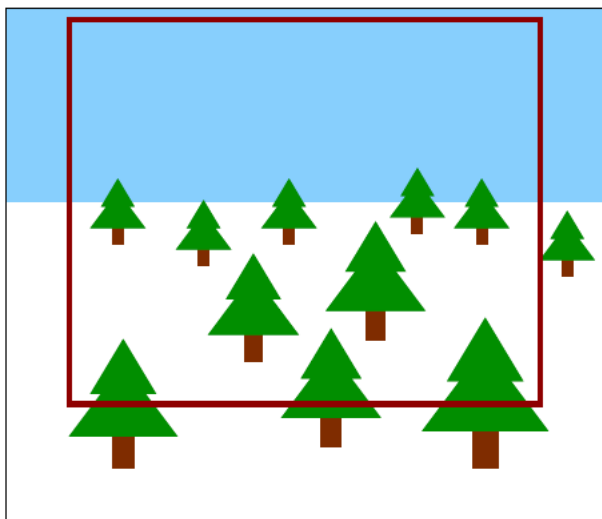
- Due to both latency and prediction imperfections, a last moment adjustment may be needed before the frame is scanned out to display
 - This is called post-rendering image warp
- No time for complicated operations, but a simple transformation
- Solution
 - Render a larger image
 - Once error in prediction is detected, move the image that should be sent to the display to correct the error



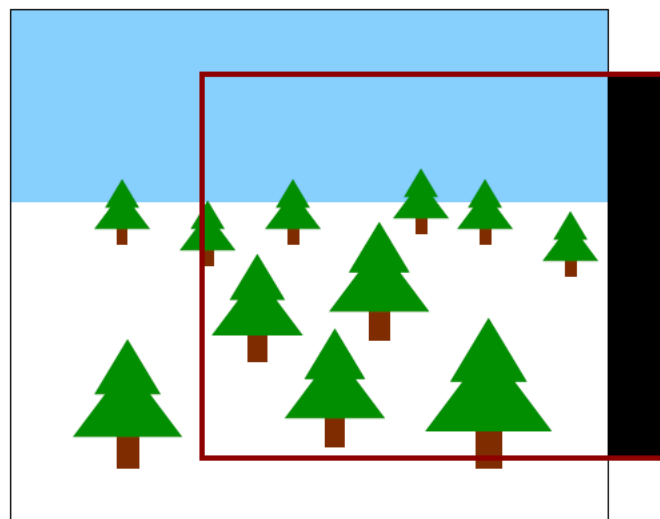
(a)



(b)



(c)



(d)

Immersive Photos and Videos

- New trend: capturing real-world photos and videos and turning them into VR experiences
 - Millions of people carrying high resolution cameras (smartphones)
 - 3D cameras: distance information + light/color
 - Panoramas: capture image data from all possible viewing directions
 - **Current challenge:** how to mimic all possible viewing directions and orientations, e.g., if the user was able to move?

Mapping into a Rectangle

- Easy to map pictures from captures to a rectangular screen
 - Typical image is rectangular
 - VR world can be mimicked as a rectangular virtual screen
 - The size of the virtual screen can be increased as needed!

Increasing the Field-of-View

- To fill the user field of view, desirable to curve the virtual screen and put the user in the middle
 - In the limit, a panoramic photo turns into photosphere
 - A sequence of photosphere is a panoramic movie called moviesphere



Two Methods to Capture Photospheres

- Take multiple images from a single camera, but tune it to different directions over time until all angles covered
- Use multiple cameras, pointing in various directions
- The photosphere can be directly mapped into a spherical virtual screen with the user at the center
- The image can be rendered only once (i.e., the entire rasterization process can be performed only once), while the image rendered to the display gets adjusted based on the viewing direction



**360Heros Pro10 HD is
a rig that mounts 10
GoPro cameras!**