CS559 Lecture 19-20: More Texture

Part 5: Shadow Maps

Shadow Maps

Note: This is online one way to do shadows

The details on how it works change

Idea: Can the Light See the Object?

We know how to make a picture of what the **camera** sees

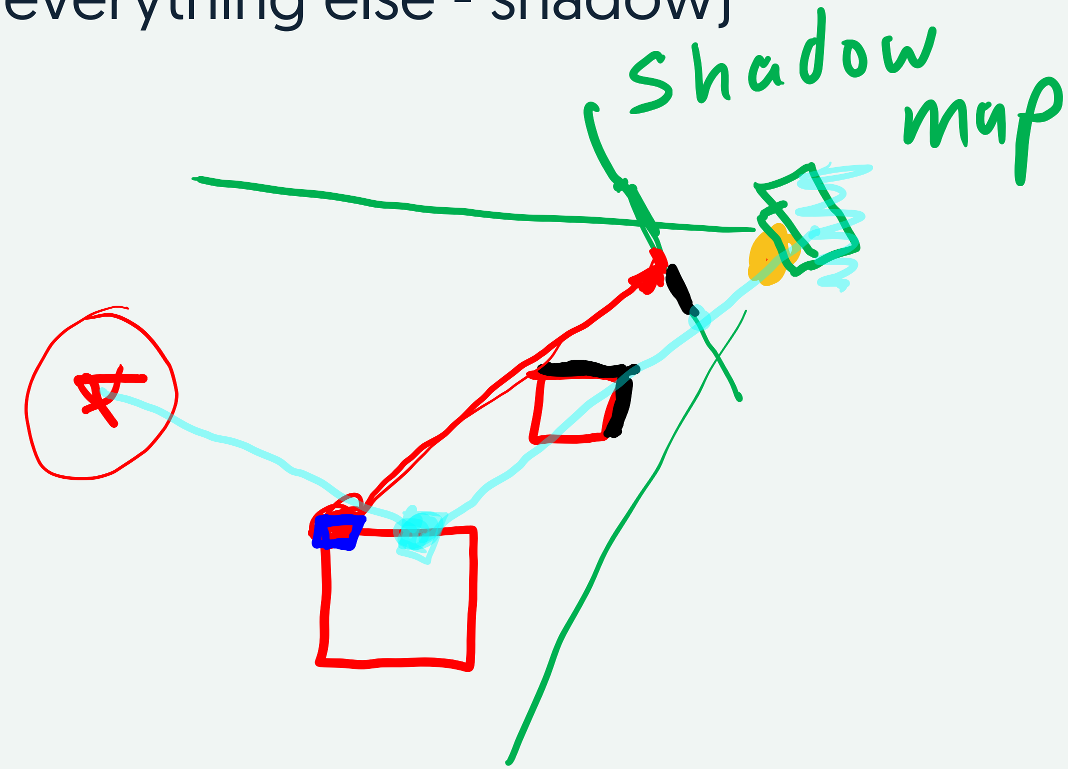
(warning - we didn't discuss how this works yet)

Use the same method to ask what the **light** sees

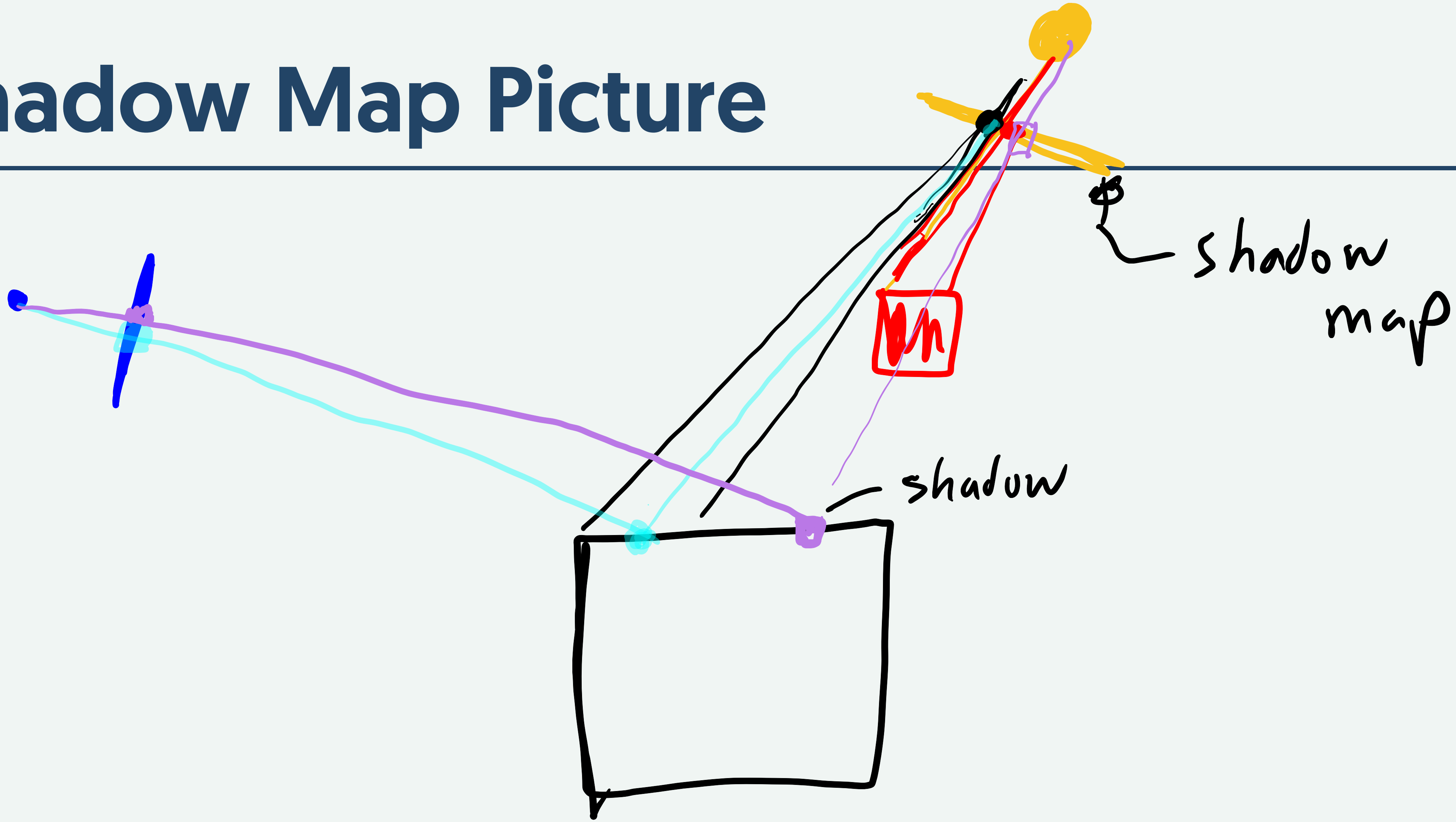
- put the "camera" where the light source is

Shadow mapping

1. Take a picture from the light position
    - camera at light source
    - what objects are visible to the light (everything else - shadow)
    - use this picture as the **shadow map**
2. Draw the "regular picture"
    - for each pixel on an object
    - see if pixel is visible in shadow map



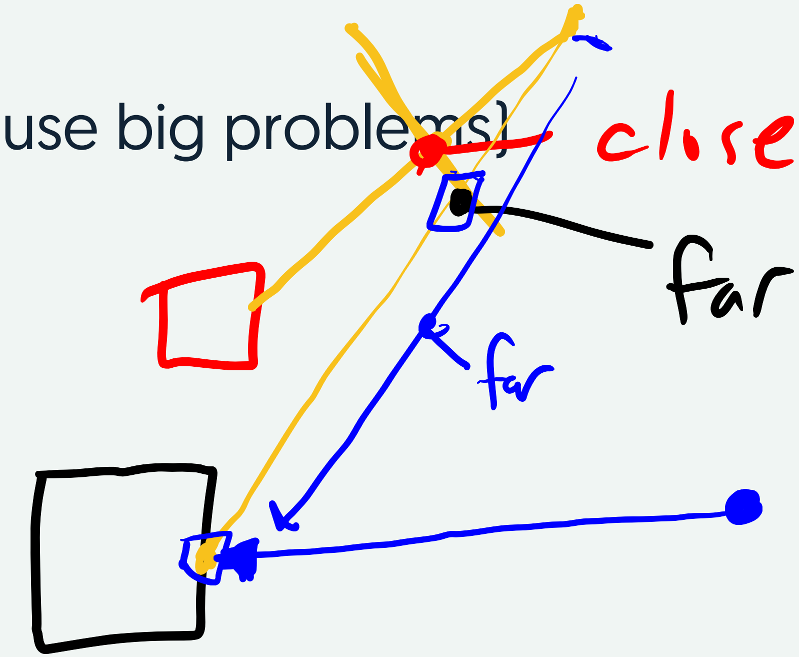
Shadow Map Picture



Shadow Map Test

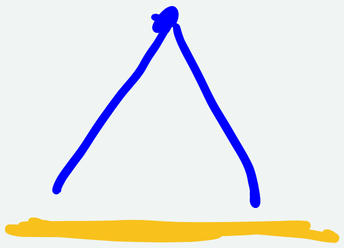
How do we know if the pixel is the same pixel in the shadow map?

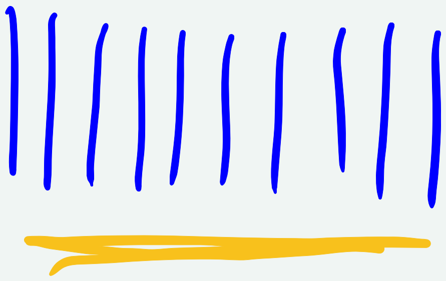
- check color (not a good idea)
- check depth (most common approach)
    - can be problematic (small errors cause big problems)

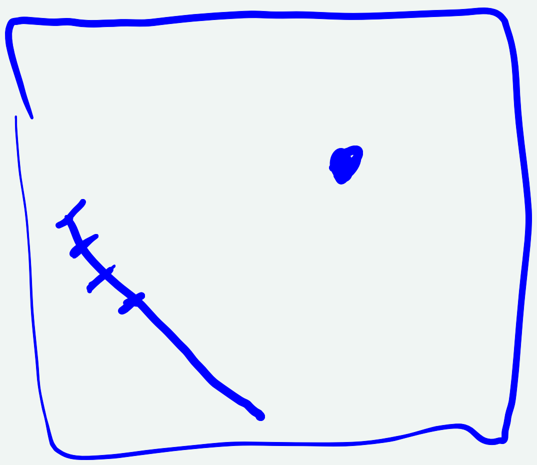


Shadow Map Resolution

- Size of Shadow Map Matters
  - spotlight (can be small)
  - directional light source (area)
  - point light? (needs to be a cube/sphere)







Shadow Maps in THREE

Of course it makes it easy!

- Tell the lights to cast shadows
  - they will make shadow maps
- Tell the objects to **cast** shadows
  - they will be rendered in all passes
- Tell the objects to **receive** shadows
  - their shaders will access the shadow maps
- Tell the renderer to do shadows
  - it will set up the multiple passes

Summary: Advanced Texture Hacks

- Normal and Bump Maps for surface details
- Layered Textures to mix effects

colors

- Lightmaps / Ambient occlusion for pre-computed lighting
- Environment Maps for Reflections (and lighting)
- Shadow Maps for Shadows

Hacks?

Why Hacks? Can't we do something more principled?

We use hacks because they are implemented efficiently in the **graphics hardware**.

(guess what we learn about next)