During CS148 we were fascinated by learning how light flows through a scene. The images that caught our attention most were dramatically lit scenes that channeled light to teach your eye something new—so we set out to create such a scene of our own.

Description of Assets

Our scene consists of a unique angular highball glass made of flint glass ("lead crystal") set atop the lip of a glossy black bar. A lit display of bottles sits behind it, refracting light through the glass and its contents. The glass is waiting to have a drink poured into it and contains three ice cubes and a brown
plastic straw that softly complements the warm glow of the bottles in the background.

We modeled the straw ourselves in Blender, using a Boolean modifier on cylinders and applying a bevel modifier to the ends. The first straw we modeled was made out of brass and had a lower polygon count. However, we found that a brown plastic straw did a better job directing the eye without being obtrusive, and we needed to remodel the straw to get smooth curvature at 4k. We came up with the final straw material ourselves by tweaking a bronze material until it felt best with the scene. The dimensions of the straw were taken from measurements Christopher took of a real straw with calipers.

The straw and the vertical black bar at the first third of the image help guide your eye to the centerpiece of the image: the glass and the refractions it contains.

Our original mesh for the angular highball glass was found online, but due to issues with the original mesh, we had to heavily modify it. The original mesh had misconnected faces, small gashes through the sides, and mismatched normals. This lead us to learn about non-manifold geometry in Blender, as we filled its gaps, removed nonsensical faces, and marshaled the model into something usable. We also slimmed and streamlined the glass’s features to make the reflections less wavy so you could see the contents of the glass.

Since the refraction is what makes the glass a part of its environment, we looked up the index of refraction for lead crystal in order to properly model its behavior. We then adjusted the other components of a glass material we looked up online until they matched the dramatic theme of the scene.

The ice cubes were also found online, but they were initially too low poly to be used at 4k. We subdivided and smoothed them in Blender. We looked up the index of refraction for ice (1.3) and modified the glass material to look like ice. The ice cubes are clear without bubbles—like the kind made from boiled water that you would find at a bar.

We made the black bar top in Blender by beveling the edges of a rectilinear solid. It is important that the edge be semi-smooth so it realistically reflects whisps of the ceiling. We designed the material from scratch, tweaking the reflection and specular color so that it gained depth by picking up subtle reflections from the scene while being muted enough that it still was a restful region for the eye. We also made sure that it was just off-black enough (slightly red-brown) that it felt distinct from the black in the backdrop and called attention back to the straw and, through it, the cup. If you look closely, at the front of the bar you can faintly see the feet of a bar table and the gold and black pattern of the carpet. We had initially sought to make a slate table top; however we decided on a glossy black to show off subtle details through reflection.

Because we depended heavily on refraction and reflection to light up our scene, we caged our scene within a box of image backgrounds. We did this to ensure no redirected rays would fly out without collision. The background billboard image is from a bar in Edinburgh, Scotland, that Christopher’s girlfriend visited 18 months ago. We had to get the image online rather than use one of hers, since we
needed about 16MP to get enough pixels at the angle we wanted. It is rotated and perspective warped slightly from the original to be as close as possible to our camera angle. We tried differentially blurring the image to create a depth of field effect, but found that having the image is also reflected around the sides to create a box that continues to refract warm colors, though we modified the image a bit to silhouette the top ice cube. The back wall is of a bar scene to give the correct reflections off the front edge of the bar and help give the bar depth. The ceiling is an ornate black tiled pattern and gets picked up in a few small reflections.

## Layout and Lighting

We had been bothered by details in other scenes we had seen in which textures or objects seemed to be mismatched in size, so we were careful to start by mathematically basing all the sizes in ours, before adjusting things to taste.

The initial position of the background was calculated by looking at the camera information in the image header and looking up the sizes of the bottles in the image to double check. We then put the camera at the focal point of that image—again using the camera settings—so it would not look distorted.

We settled on a camera angle of 25 degrees. Sixty degrees looked a bit too flat, so we calculated our angle based on how far we tended to sit from our screens while working (as an approximation of where you will sit while viewing). We chose 3:4 as the aspect ratio because it felt right, given the image.

We put the center of the glass through the first vertical rule-of-thirds line. We also tweaked the image so the two middle vertical bars align with the vertical rule-of-thirds lines. The bar lip was positioned so that it caught the bottom horizontal rule-of-thirds line.

Finally, we adjusted the image slightly back to make the second row of bottles reflect off the bar top. This helps give a sense of depth.

Overall, we aimed to create a balanced image that lets your eye explore, while guiding it gently back toward the glass with leading lines. The straw and image placements help with this, as do the lights and a number of effects we implemented (described in the next section). Briefly, we created a new material that turns the billboard images into lights for reflection and refraction, and we ended up placing only two additional point lights. There is a front light that is a warm yellow, as if from the well-lit bar. In addition, there is a stiffer orange light coming from behind that softens the shadow behind the glass, as well as developing a richer table color, and adds a leading line to the front of the glass. We had considered switching these to area lights, but the harder shadows went better with the dramatic themes in our image and with the hard black lines in the background. So much for realism.

## Technical Contributions
We’ll talk about the lighting effects first:

We implemented a new material for creating billboard images that essentially turns them into giant light sources for reflected and refracted rays. Any ray that hits one of the backdrop or side images takes on the color at that (interpolated) texture position. This prevents additional shadowing and speeds runtime, since we did not have to cast further light rays for those rays.

When we first rendered our image, it was hard for the eye to stay and explore the frame because of the bright leading lines converging to the right. To fix this problem, we approximated lens vignetting, wherein the edges are darkened because the aperture is effectively smaller for light coming in at an angle. We implemented this as a final pass over the rendered image, dimming pixels by the squared cosine of their distance from the center over the max distance times a tweakable constant that simulates the relative size of the lens and the image sensor. This made quite an effect in terms of being able to keep your eye exploring the details of the image.

It was still somewhat hard to keep your eye focused on the glass within the image because using realistic transmittance values made it darker than the background. Therefore we modified the raytracer to desaturate rays that directly hit billboard images by 15%. This was subtle enough to not look unnatural while helping the glass pop out as the focal point.

Other changes:

Because runtime is always an issue, we parallelized the raytracer using OpenMP with collapsed loops and guided scheduling. Since the time to compute pixels is heavily unbalanced between the background and the glass, guided scheduling is key to getting the largest speedup. Similarly, we parallelized the vignetting effect.

We also modified the diagnostic logger to not crash under races, as the original tree map based implementation sometimes did, by switching the backing to an array indexed by the enum values. We still left out locks for speed, since we’re fine with an approximation, Hogwild!-style.

We also added logging for culling of reflection and refraction rays, which helped us realize that lots of our render time was being taken up by rays reflecting back and forth forever inside the glass, continuously spawning refraction rays. Since the reflections quickly got dim, we clipped the number of them heavily for speed, without having any visual difference.

Finally, for speed of adjustment, we modified the camera and image placement interfaces so we could directly copy in values from Blender.

A note on starter code issues for future students:

In early versions of our scene, we noticed that most objects had hard noise scattered across their concave or flat surfaces (see images, below). Further, we noticed that the properties of the noise
changed when we disabled acceleration structures on the objects. Even with lots of samples per pixel, ring shaped patterns remained on the background.

![A very early version of our scene. Note the black noise.](image1.png)

![Symmetric noise tipped us off to the $\epsilon$ issue.](image2.png)

After much debugging, we realized that the ring shaped patterns had to be the result of spurious light ray intersections. Changing LARGE_EPSILON to a value five times larger (5e-5f) fixed the issue on all objects, and allowed us to render our image noise-free with only 16 samples per pixel.

If you’d like our code for any of these for the starter codebase (epsilon fix, diagnostic logger that doesn’t crash under parallelism, vignette effect, etc.) we’d be delighted to send it over. Just shoot us an email at cpsauer@cs.stanford.edu and krake@cs.stanford.edu.

**Work Breakdown**

Since we were both trying to learn as much about the whole process as possible, we worked almost exclusively though everything in tandem. Since Christopher had used C++ more recently, he took point on coding tasks, and Kevin, on placement and materials, but we thought through both together.