1 Colours, shaders and rendering

Describe the following four terms gamut, hue, saturation, refraction

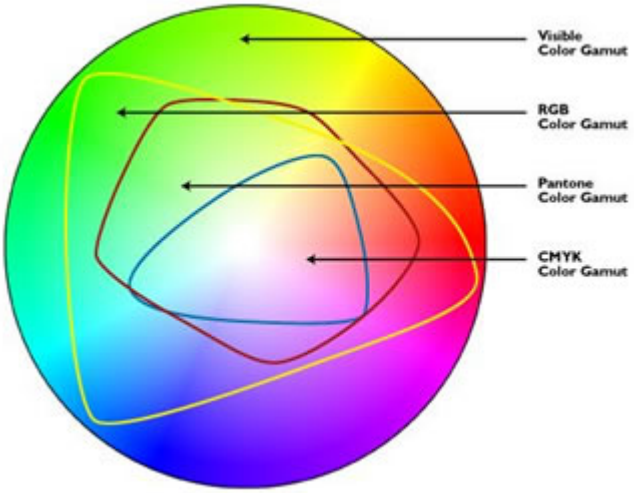

Figure 1: **Gamut** - The set of colours that can be represented in a certain colour space or by a certain device.

-

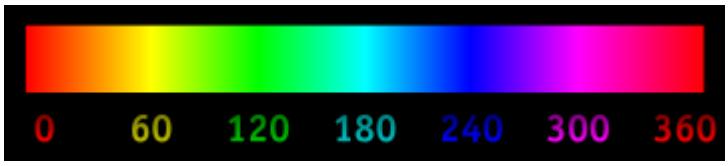

Figure 2: **Hue** - E.g. blue, orange, red. The appearance of a colour that is independent of brightness or saturation, and is associated with the dominant frequency, or combination of frequencies, of light.

-

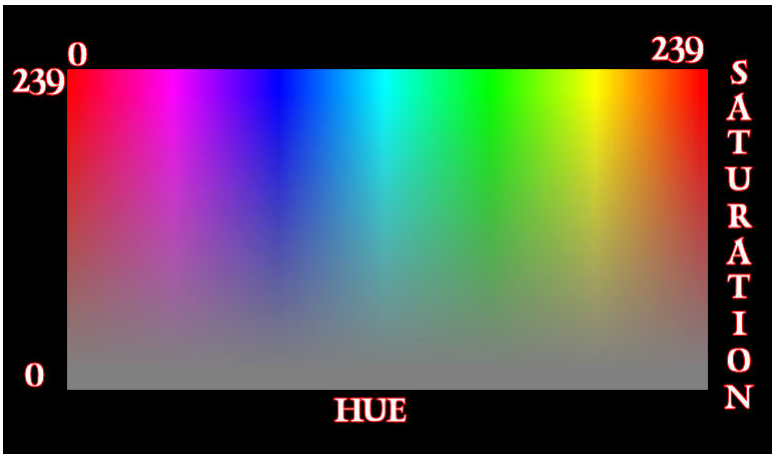

Figure 3: **Saturation** - The relative colour purity, or how little white light is mixed in. Monochromatic light has full saturation, while white light as zero saturation.

-

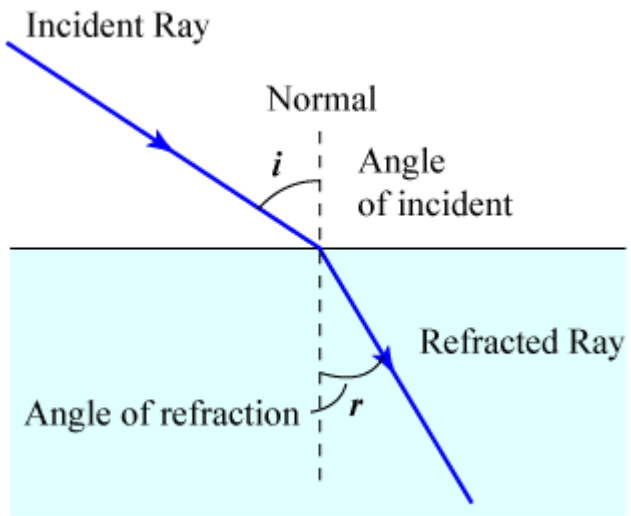

Figure 4: **Refraction** - Bending of light as it passes the boundary between two materials in which the speed of light is different. (Index of refraction)

-

A common colour space is sRGB, where colours to be displayed are modified as weighted sum of three specific R, G, B components. Would extending sRGB to sRGBC colour space, that also includes a cyan component, affect the gamut? Yes, it would increase the gamut. The base colours of sRGB form a triangle in the Chromaticity diagram. If the C base colour is chosen outside of this triangle, the gamut of the sRGBC colour space will be bigger. However, if the cyan base colour is chosen to be within the sRGB triangle, then it would not affect the gamut.

List advantages and disadvantages of a single pin-hole camera model with zero exposure when rendering computer graphics.

- Pin-hole camera gives no depth of field (out-of-focus blur)
- Zero exposure time gives no motion blur
- Zero aperture (pin hole) and zero exposure time means there zero photons, so no image at all, if the image were rendered in a physically correct way
- Single camera means no depth perception

Difference between Phong or Gouraud shading and the Phong reflectance model

- **Phong shading** is a way to interpolate the rendered colour over a polygon face, which interpolates the normals, and computes the colour using a per-pixel normal.
- **Gouraud shading** evaluates the colour of the corner pixels of the polygon and lineraly intepolates the colour over the face.
- **Phong's reflection model** is the traditional model that computes the reflected light as a sum of an ambient, specular and diffuse term.

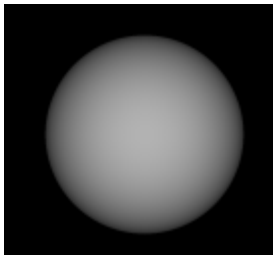

Figure 5: Lambertian sphere

Looking at figure 5. The BRDF of a Lambertian surface is a constant value. Why does the surface not appear flat white, but instead is

darker towards the silhouette? It does not appear flat white because of **Lambert's cosine law**. The amount of reflected light is multiplied by the cosine of the angle between the surface normals and the direction of the incoming light.

Figure 6: Full moon

We want to model the moon as a smooth sphere. Suggest a lighting model more suited than purely Lambertian scattering. Disregard the texture.

- Phong reflection with a large ambient (or emissive) term would be an easy solution.
- More physically correct would be to model the surface using the Cook-Torrance model. Cook-Torrance models the rough surface of the moon as a collection of small mirrors.
- Even better would be using Oren-Nayar model. Oren-Nayar models the rough surface of the moon as a collection of small Lambertian reflectors.

Figure 7:

$$L(P, \omega_o) = L^e(P, \omega_o) + \int_{\omega_i \in S^2(P)} L(P, -\omega_i) f(P, \omega_i, \omega_o)(|\omega_i \cdot \mathbf{n}_P|)\, d\omega_i \quad (1)$$

$$L(P, \omega_o) = L^e(P, \omega_o) + \int_{\omega_i \in S^2_+(P)} L(P, -\omega_i) f(P, \omega_i, \omega_o)(\omega_i \cdot \mathbf{n}_P)\, d\omega_i \quad (2)$$

Figure 8: Two rendering equations

The rendering equation can be written in the following two forms. Which materials can they model? Describe the implications of the difference in the limits used in the integral and use of an absolute value in the last factor

- Equation (1) uses the **reflectance equation**. It integrates incoming light over all the full sphere, so it can also render transparent materials, with incoming light from below the surface.
- Equation (2) uses the **scattering equation**. It integrates over the positive hemisphere (see s^2_+) - only light from outside the surface. It cannot render light that has passed through the object, which means it can model only opaque surfaces.

2 Ray casting

Figure 9: Path tracing room

Consider figure 9, it shows an indoor scene with two light sources, rendered with path tracing

Difference between ray tracing and path tracing when rendering indirect illumination. How would it affect the appearance of the figure above.

- Ray tracing traces one ray per pixel to its first intersection with a surface and emits one recursive ray from the intersection point towards each light source to determine whether it is shadowed or not. If the material has a reflective component, ray tracing emits a recursive ray and another if it has a translucent component. Stochastic ray tracing emits several rays with perturbations to mitigate aliasing artifacts (several rays through a pixel).
- Path tracing traces many rays per pixel, where each ray is traced until its "end" - the whole path of light. (Is it not possible to trace the path until it ends up at the light source, so the path can be terminated with probability proportional to the reflectance of the material at each bounce). At each bounce, secondary rays are traced to all light sources in order to determine whether the point is shadowed or not.

With ray tracing only surfaces that are directily lit by a light source would be non-black in the figure. These includes, the corner of the ceiling above the floor lamp, and part of the table and the left wall, and the glass egg.

Path tracing implementations typically make use of Monte Carlo integration. Use Monte Carlo integration, with one sample, to compute the integral $\int_0^1 x^2 dx$ Monte Carlo integration works as follows: **Given n samples, a function $g(x)$ to integrate, and limits $i_1$and i_2:**

- For each sample:
  - select a point $p \in [i_1, i_2]$
  - compute $s = g(p)$
  - multiply by the size of the region: $s \leftarrow s * (i_2 - i_1)$
  - add it to the total sum $S \leftarrow S + s$
- Return $\frac{S}{n}$

In this case, we have $n = 1$, $g(x) = x^2$, $i_2 = 1$. If we pick $p = 0.5$, we have $g(0.5) = 0.25$, multiplied by one, so the approximation of the integral is $\int_0^1 x^2 dx \approx 0.25$. (The true value is $\frac{1}{3}$.

Some rendering methods produce a biased value of each pixel and some produce an unbiased value, but with larger variance. Explain where such bias and variance come from, and how they affect the appearance of the rendered image. The variance comes from stochastic sampling used in path tracing. Each sample is offset from the currect value but the mean of all samples is at the correct value. The variance describes the spread around the mean.

The bias on ray tracing comes from the fact that not all information is used, which means that there is a systematic error (bias) between the computed value and the physically correct value.

- An image with large variance will be noisy (grainy-looking).
- An image with large bias but small variance will be smooth, but have incorrect colours (compared to a more correct physical model).

3 Shadows

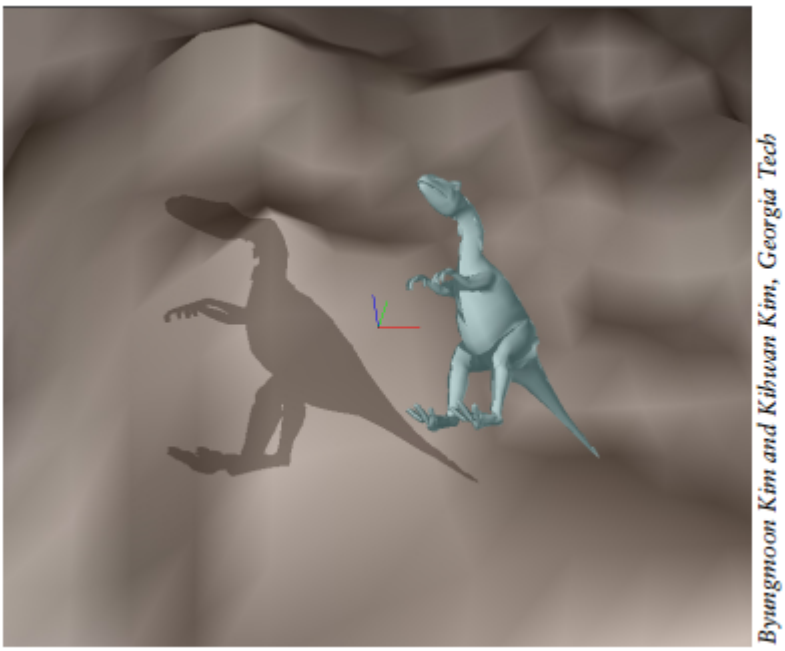

Figure 10: Hard shadow rendered with shadow volume

Consider figure 10. It shows an example of a hard shadow rendered in real-time using shadow volumes

Outline the steps of a naive shadow volume algorithm for rendering hard shadows cast by a polygon model onto a terrain object, given a point light source, represented by the figure.

- Generate shadow volumes (by creating shadow triangles that extend from the edges of the actual geometry and away from the light source).
- For each pixel, count how many shadow volumes a ray through this pixel enters
- Count how many shadow volumes the ray exits
- If it enters more volumes than it exits, then the pixel is in shadow, otherwise not.

A simple shadow volume algorithm that uses the stencil buffer might look like this:

1. (Optional) First render ambient light only

2. Turn off frame buffer and depth buffer
3. Generate shadow volumes
4. Render only the forward-facing shadow triangles, incrementing the stencil buffer with 1 for each triangle
5. Render only the backward-facing shadow triangles, decrementing the stencil buffer with 1 for each triangle
6. Turn on the frame and depth buffer, and render the scene as usual, using the stencil mask and rendering only those pixels where the mask = 0

Discuss performance issues of your algorithm and give details of how it can be amended to be more efficient. Alternatively, give details of what measures you took in the previous algorithm. A naive shadow volume algorithm generates more shadow triangles than necessary. One step to decrease the amount of extra geometry generated is to first extract the silhouette edges of the objects and only generate shadow volumes from them.

A naive algorithm might also do ray tracing for each pixel. Implementing it using the GPU stencil buffer makes it more efficient.

Shadow volumes are typically used for rendering hard shadows. Discuss how your shadow volume algorithm could be amended to render soft shadows One could sample many point lights over the area light's surface and render shadow volumes for all of them. This is a brute-force way of rendering soft shadows, and would be slow, since it would require generating a lot of geometry.

Another idea would be to generate two sets of shadow volumes from the same point light (one inside the other), where points that are inside the outermost volume only are in the penumbra (see figure 11). This might be faster than the above option, but would require additional work to compute the distance to the inner and outer shadow volume and be susceptibe to artefacts (aliasing) when a surface is shaded by two nearby or overlapping objects.

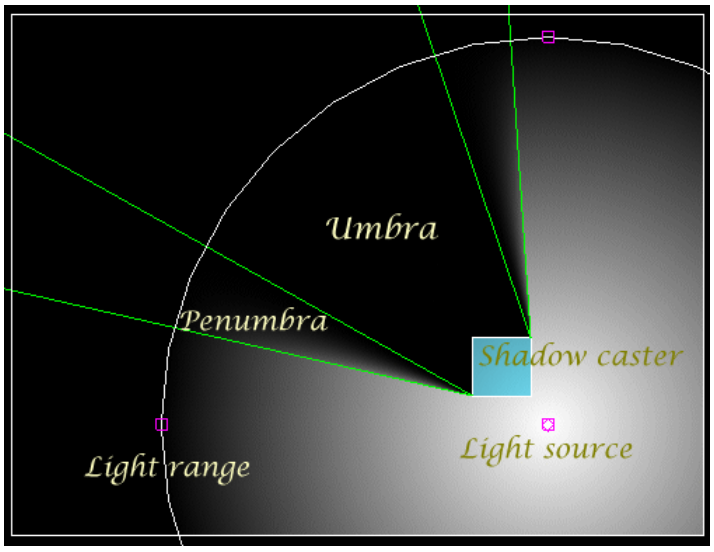

Figure 11: Penumbra

Discuss pros and cons relative to implementing soft shadows with shadow mapping instead With shadow mapping, plausible (but incorrect) soft shadows can be generated quickly with, for example, percentage closer filtering.

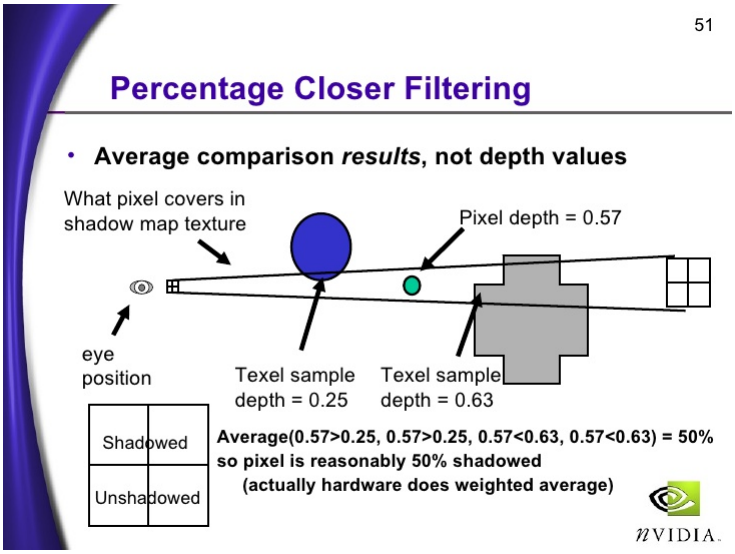

Figure 12: PCF