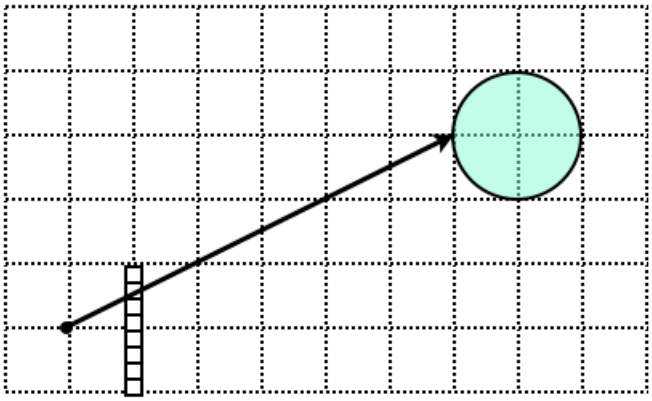
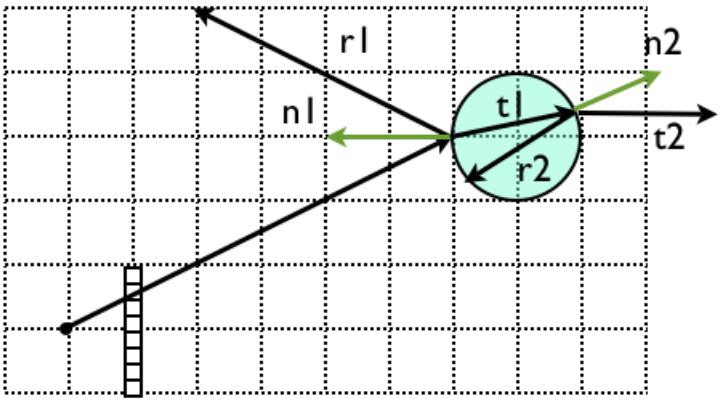
Week 10 Tutorial Solutions

Question 1:

The image below shows an eye ray from the camera hitting a glass sphere.

Draw the **reflected** and **transmitted** rays. Use Snell's law to compute refraction angles, assuming the speed of light in the sphere is 50% of the speed in air.





$$
\begin{aligned}
\tan\theta_1 &= \frac{1}{2} \\
\theta_1 &= 26.6^\circ \\
\sin\theta_1 &= 0.447 \\
\sin\theta_2 &= \frac{c_2}{c_1}\sin\theta_1 \\
&= 0.5 * 0.447 \\
&= 0.223 \\
\theta_2 &= 12.9^\circ
\end{aligned}
$$

The normals are shown in green. The first collision creates reflected ray r1 and transmitted ray t1. If the angle of the incoming ray to the normal is 26.6 degrees, as shown, then the angle of the reflected ray is also 26.6 degrees. The angle of the transmitted ray can be computed by Snell's law to be 12.9 degrees to the normal as shown.

The transmitted ray will branch again as it exits the sphere. The angle of incoming ray t1 to the normal n2 also happens to be 12.9 degrees (due to the symmetry of the circle) so the angle of the outgoing transmitted ray t2 is 26.6 degrees to n2.

There is also an internally reflected ray r2.

Question 2:

We can rearrange the terms in Snell's Law to get:

$$\sin\theta_1 = \frac{c_1}{c_2}\sin\theta_2$$

What happens when this value is greater than one? Under what circumstances will this occur?

This is a case of total internal reflection. All the light is reflected internally and none is transmitted. Note that it can only occur when moving from a slow medium to a faster one (eg, from water to air). In case you are interested, the general formulae for the amount of light reflected vs transmitted is the Fresnel equation . It depends on the polarisation of the light source. A simpler approximation is Schlick's Approximation .

Question 4:

Consider a cloud represented as the array of densities below:

0	0	0	0.1	0.1	0
0	0.1	0.1	0.1	0.1	0
0.1	0.1	0.2	0.1	0	0
0	0.1	0.1	0	0	0
0	0	0.1	0.1	0.1	0
0	0	0	0.1	0.1	0.1

The cloud has uniform colour C(x,y) = (1,1,1) and transparency alpha(x,y) equal to the density values above. The sky behind the cloud is blue C = (0,0,1).

Suppose we are looking up at the cloud from below. Assume the cloud is far enough away that we can render it with an orthographic camera (parallel rays) with little distortion.

Cast six equally spaced vertical rays through the cloud, taking samples distance 1 apart. Calculate the pixel values for each ray.

What happens to the result if we take samples at twice the frequency? or half? Is this behaviour correct? How could we adapt the algortihm from lectures to make it less sensitive to such changes?

An example for one ray, as illustrated, using back-to-front compositing:

$$
\begin{aligned}
C_6 &= (0,0,1) \\
C_5 &= 0.0 * (1,1,1) + 1.0 * (0,0,1) \\
&= (0,0,1) \\
C_4 &= 0.1 * (1.1.1) + 0.9 * (0,0,1) \\
&= (0.1,0.1,1) \\
C_3 &= 0.2 * (1,1,1) + 0.8 * (0.1,0.1,1) \\
&= (0.28,0.28,1) \\
C_2 &= 0.1 * (1,1,1) + 0.9 * (0.28,0.28,1) \\
&= (0.35,0.35,1) \\
C_1 &= 0.1 * (1,1,1) + 0.9 * (0.35,0.35,1) \\
&= (0.42,0.42,1) \\
C_0 &= 0 * (1,1,1) + 1.0 * (0.42,0.42,1) \\
&= (0.42,0.42,1)
\end{aligned}
$$

What happens to the result if we take samples at twice the frequency? or half? Is this behaviour correct? How could we adapt the algortihm from lectures to make it less sensitive to such changes?

If the same method is applied with more samples along the ray, the transparency of the cloud is reduced because the algorithm as shown does not take sample density into account. If the cells are spaced every d units and sample points on the ray are r units apart then the alpha value should be:

$$\alpha_i = 1 - (1 - \alpha(P_i))^{\frac{\Delta r}{\Delta d}}$$

Question 4: Sample Exam Questions

a and b are similar in style to Part B of the final exam which are short answer questions. c,d,e and are similar in style to Part C in the final exam which are design questions

a. Explain the advantages and disadvantages of normal mapping (a.k.a. bump mapping) over adding extra polygons to represent detail.
   - **Advantages: Every polygon we add to the scene creates extra computation at every step of the pipeline: transforming, illuminating, clipping, rasterising, etc. Having a large number of polygons is therefore very computationally expensive. Normal mapping allows us to represent rough surfaces with far fewer polygons.**
   - **Costs: Extra work has to be done in the texturing stage. Normals must be computed for each pixel on the surface and the illumination equation recalculated to include data from the normal map. The map itself must be stored in memory, doubling the amount of texture memory required.**
   - **Disadvantages: One problem with using normal maps is that they do not affect the outline of the object. So if the surface is viewed on an angle, it will appear flat. Also normal mapping only works for minor perturbances in the surface. Larger bumps should occlude other parts of the suface when viewed at an angle. Normal maps do not support this occlusion.**

b. We have looked at three methods for simulating shadows: shadow mapping, ray-tracing and radiosity. What are the pros and cons of each technique? When might they be appropriate to employ?
   - **Shadow mapping: is relatively fast and can usually be done in real time. It requires two extra rendering passes per light source, one to compute the shadow buffer and one to apply the shadows to the scene. The quality of the shadows is limited by the resolution of the shadow buffer. A higher resolution buffer gives better looking shadows but is more costly in terms of time and memory. Shadow edges are hard and appear jagged at low buffer resolutions.**
   - **Ray-tracing: is relatively slow and can only be done in real time on very high-end machines. For every pixel in the rendered image a ray is cast from the object hit towards each light source in turn to decide whether the source is**

occluded or not. This avoids the quality prolems created by the shadow buffer, but it much more computationally expensive. The advantage is that it can take into account reflected and refracted light and so produces much more realistic lighting. Nevertheless the shadow edges are still typically hard.

- **Radiosity: takes into account light from all objects in the room, not just light sources. This means that shadows are much softer and realistic because it takes into account light from very many angles. As a result, the computational cost is much higher and this approach is typically not suited for real-time rendering. However the results of radiosity calculations can be baked into static light-maps and used in real time situations if the lights and objects in the scene are not moving.**

c. Below is an image of M. C. Escher's Three Worlds. We want to render a short animation of this scene. What techniques would we employ?

- **Assuming the scene is not intended to be rendered in real time (i.e. it is a static animation, not an interactive systems) then we are free to use more expensive rendering techniques to produce high-quality images. One of the crucial elements of this scene is the water: it reflects the images of the trees in the distance but it is also transparent and allows the underwater fish to be seen. This calls for a ray-tracing renderer, which is able to handle reflections and transparency.**
- **The water itself could be rendered using volumetric ray-tracing so that the light diminishes as rays pass deeper underwater. Snells law could be used to compute refraction as light passes between the water and the air. If you look carefully, you will notice the the water is more transparent in the foreground and more reflective at the back. The Fresnel equations (not covered in this course) describe this behaviour, the proportion of light that is reflected or transmitted depends on the viewing angle.**
- **The trees in the background could be procedurally generated using an L-System. The leaves on the water surface could also be generated using an L-System (although this hasn't been discussed in the course). The fish would probably need to be modelled by hand.**
- **If this were to be animated it would be nice to have the trees move with the wind and add ripples on the surface of the pond.**
- **Finally, the entire image could be rendered with a non-photorealistic filter to imitate Escher's original lithograph.**

Note: This is a much longer answer than I would expect from you in the exam. I provide it here to show the variety of things to think about in rendering a simple image like this. If this were an exam question, it would focus more on the particular use of ray-tracing to render this scene.

d. For an art project you need to render a polished wooden bowl like the one below. How would you generate this mesh? What method would you use to texture it? What would its material properties be for lighting?

- **A mesh for this shape could fairly easily be extruded by taking a vertical cross section through the middle of the bowl and rotating it around the Y-axis (creating a surface of revolution). To create the effect of the bowl having been carved out of a solid piece of wood, it would be more appropriate to use a 3D texture of woodgrain rather than try to wrap a 2D texture around this surface.**
- **The bowl shows both specular and diffuse illumination (notice the specular highlights on the rim and inside the bowl). The specular highlights are broad, suggesting a low shininess value would be appropriate. Phong shading would probably be appropriate to ensure that the surface appears curved and highlights are rendered well. A normal map could be added to give the surface some roughness so that it doesn't look too glossy.**

e. You want to implement a smoky fire in a 3D game. Name (at least) two different approaches to implementing this. What are the pros and cons of each?

- **Volumetric objects like fire and smoke are usually implemented as either particle systems or using volumetric ray tracing. A particle system represents the volume as a collection of moving particles. A volumetric system represents the volume as a grid of cells with varying density.**
- **In broad terms, particle systems are easier to implement in polygon-rendering systems like OpenGL as they can be represented simply as collections of quads or point sprites, which are generally supported with little extra coding. Grid-based representations will required more specific ray-marching code to be written. On the other hand, grid-based representations are easier to implement in ray-tracing systems, where particles require lots of individual collisions to be calculated.**
- **The pros and cons of particle vs grid-based simulations are actually well beyond this course. I put it in here as a challenge question to get you thinking. To learn more about this I recommend the Fluid Simulation for Video Games articles by Dr. Michael J. Gourlay.**

If there is any more time left, please go over material from previous weeks tutorials that were not finished or discuss last minute assignment issues.