HOMEWORK #5

POOLS, RIPPLES, REFRACTIONS AND SHADERS
DUE MAY 1

(1) In this homework you need to render a pool full of water. The pool is cube-shaped — all angles are 90°, and all edges have the same length. Assume the center of the water surface is at coordinate $(0, 0, 0)$, and the center of the bottom of the pool is in $(0, 0, -10)$.
(2) The viewer p_{viewer} is located above the center of the pool and high above the pool. So for all practical purposes, the rays emerging from p_{viewer} towards the pool are parallel.
(3) You can assume that only ambient light exists.
(4) On the bottom of the pool, a picture is painted. We refer to it as the input image I, and its pixels are $I[x, y]$. This image must contain a sufficient number of pixels, with a sufficient number of colors so that all the resulting effects described next are clearly noticeable. For example, an image of 20×20 pixels, all black, is a very bad input image. As in previous homeworks, the name of the input image file can be passed as a parameter to the program.
(5) When starting the program, or after pushing the 'r' (restart) key, the water surface is completely smooth.
(6) The color of the pools walls is up to you, but is the same for all the walls.
(7) Since all the phenomena we discuss have radial symmetry around the center c, it would be convenient to describe points on the surface of the water by the coordinate (ρ, θ, z). If you need to switch to the Cartesian coordinate system, recall that $x = \rho \cos\theta$ and $y = \rho \sin\theta$. We denote by $h(\rho, \theta)$ the height of the water (above/below the edge of the pool) at the point (ρ, θ). So when no wave exists, then $h(\rho, \theta) = 0$ for every (ρ, θ). Note that it could be either a positive or negative term.
(8) Triggered by hitting the 'd' (drop) key, you will need to simulate the effect of a small rock dropped exactly at the center of the pool. This causes waves to spread around the point where the rock hit the surface. Let v the speed of the wave, and assume the rock was dropped at time $t = 0$. Note that if $\rho > t \cdot v$ then $h(\rho, \theta) = 0$, because the wave did not reach this point yet. (ρ is the distance from the center c to the point (ρ, θ).)
(9) For points (ρ, θ) for which $\rho \leq t \cdot v$, we assume a sinusoidal wave. This means that

$$h(\rho, \theta) = A \sin((vt - \rho)2\pi)$$

. Here A is the *Amplitude*, and is a user controlled parameter.

Hints

(10) You want to allocate a shader to each pixel on the surface of the water. This shader traces a ray r_1 emerging from the viewer, hits the surface of the water at a point q, changes its orientation and continues along a ray r_2 till it reaches some point (x, y) on the bottom of the pool. The color of p is the color of the pixel $I[x, y]$. The change of direction from r_1 to r_2 is computed according to Snell's law (http://en.wikipedia.org/wiki/Snell's_law). To implement it, we need to know the angle between the normal $\vec{n}_p$ to the water surface at p, and the direction $\vec{v}_p$ from p to the viewer. Since the viewer is located very high above the ground, we assume that $\vec{v}_p$ is $(0, 0, 1)$. That is, the direction from p to the viewer is vertically upward.

(11) To find $\vec{n}_p$, again it is convenient to use polar coordinates. Note that $\frac{\partial}{\partial\theta}h(\rho, \theta, t) = 0$ for every time t (that is, the water height $h(,)$ is always the same along every circle around c. Note also that

$$\frac{\partial}{\partial\rho}h(\rho, \theta, t) = \frac{\partial}{\partial\rho}A\sin(2\pi(vt - \rho)) = -2A\pi\cos(2\pi(vt - \rho)).$$

To help you visualize this function, you might want to think first about the behavior of the ripples along the x-coordinate only (replace ρ by x), and then realize that h is radially symmetric around the z-axis, and the center of the pool.

(12) Note that shaders cannot share large arrays. In particular, they cannot share the input image $I[x, y]$. Instead, once a shader computes the point (x, y) where its 'own' ray hits the bottom of the pool, it uses a *sampler* (see class notes) to find the color of the pixel $I[x, y]$.

Bonuses (15 points each)

(13) Allow the viewpoint to be in an arbitrary location — close to the surface of the water, and not necessarily above the center of the pool.

(14) Write a different program that assumes a point source of light (in addition to the ambient light), arbitrary location of the viewer, and assume the pool contain *mercury* rather than water. In this case the bottom is not seen, but instead there are interesting pattern creating by the light, including specular shading.

Guidelines

(15) Submit your program when all unspecified parameters are set so the ripple effect is clear visible.

(16) Assume the input image is of size 100×100. Pick an image for which the ripple effect is visible, but recall that the image filename is passes as a parameter, enabling changing the image.
(17) Hitting 'A' or 'a' will increase/decrease the amplitude by %30. Hitting 'V' or 'v' will increase/decrease the wave velocity by %30.
(18) Hitting 's' will return all parameters to their original setting. In particular, the face of the water returns to be flat. Hitting 'd' restarts a new wave, but with the new A, v values.
(19) You can ignore the effect of a ray hitting the water more than once. Assume after first hitting point the ray contains along a straight line.