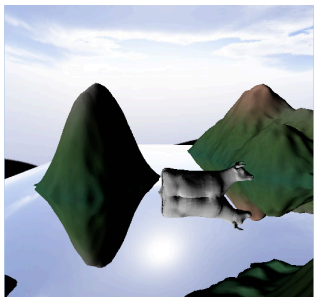
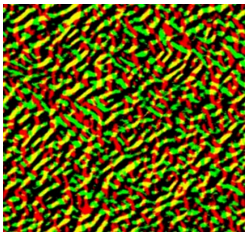
Realistic Water Rendering

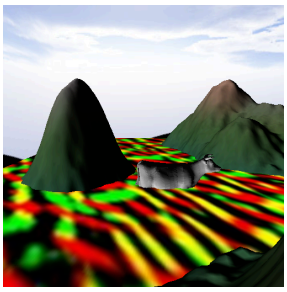
Xin Chen



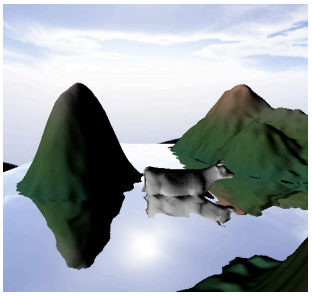
Phase 1: Water with reflection texture
I used a frame buffer object to draw the reverse scene as seen by the user, then bound the image to a texture unit. This image was then set as the texture of the water surface. To make the water look correct, special attention was taken to determine the appropriate texture coordinate, namely matching the texture using the fragment coordinate in the fragment shader. Namely, the texture of the water at the fragment coordinate was given the texture vector of the same fragment coordinate, while the rest of the water not seen by the user are left alone.



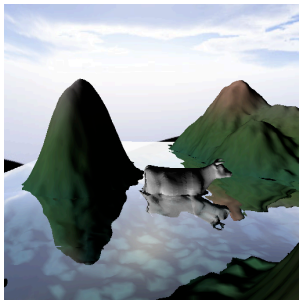
dudv.jpg
This image of green and red pixels was used to disturb the normal vector of the water surface. The intensity of red in each pixel was used to disturb the x component of the normal vector of the water at the same texture location, and the intensity of green was used to disturb the y-component. This image was also later used in phase 4 to act the same way in disturbing the normal vector of the water, only this time for the purpose of changing the texture look up of the refracted image underneath the water.



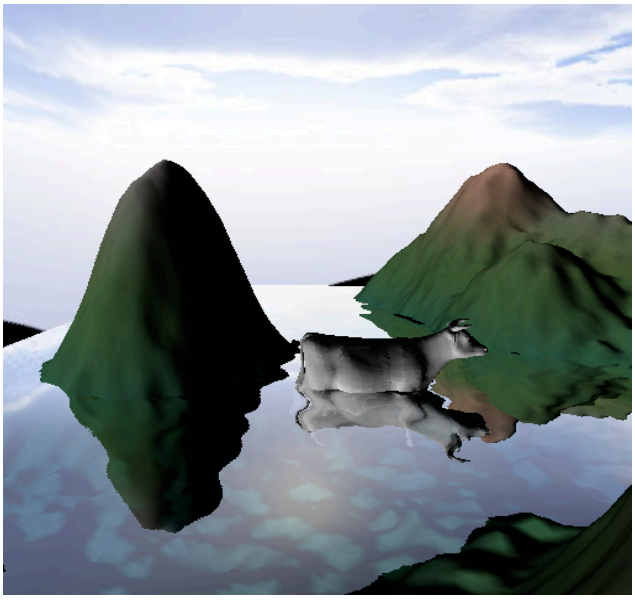
dudv.jpg on water texture
This is an illustration of dudv.jpg being used as a texture on the water plane to disturb the normal vectors. Note that what cannot be seen is that the texture is being scrolled in the direction of movement to simulate moving water.



Phase 2: Scene with ripple effect to disturb the reflection texture coordinate selection
While the vertex location of the water was not changed, the disturbance of the reflected image gives an impression that the water has a changing height.



Phase 3: Changing the alpha value to simulate light passing through to the bottom
The alpha value was changed in the fragment shader using the following factors:
Alpha(water) =
distance_factor + angle_factor + depth_factor
To do such calculations, the user position as well as the terrain height map were passed into the fragment shader for the distance/angle and depth factors, respectively.



Phase 4: Scene with refraction
To create the refracted scene underneath the water, I used another frame buffer object to draw to and bind the scene below the water line to a texture. This was drawn on a quad right below the water, and a fragment shader was used to alter the image using the ripple texture to simulate the difference of normal vector of the surface as described earlier. In addition, the darkness of the refracted image was scaled based on the depth and distance to simulate the attenuation of light through water. In other words, the refracted scene which passes through more water before hitting the land will be darker than the refracted water which passes through less.