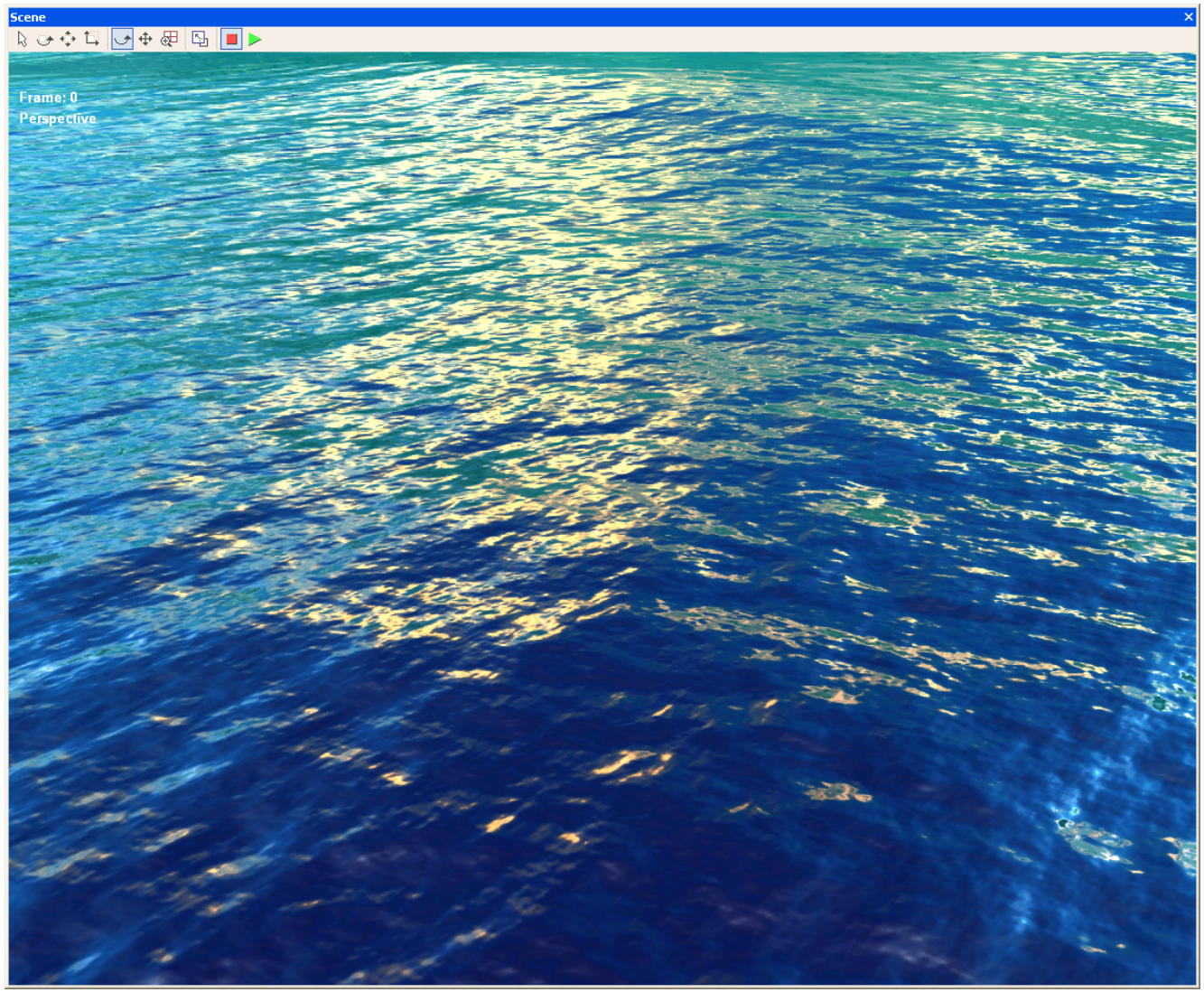
Water Effect

Advanced Shading and Rendering

2006

Aim

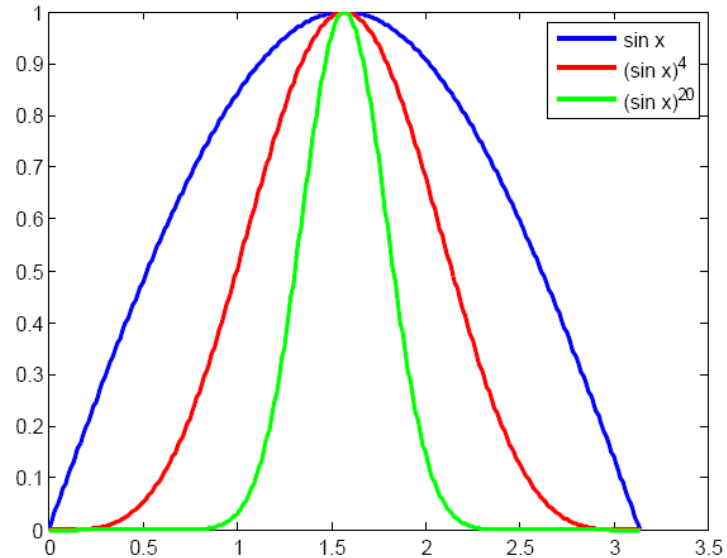


Preparations

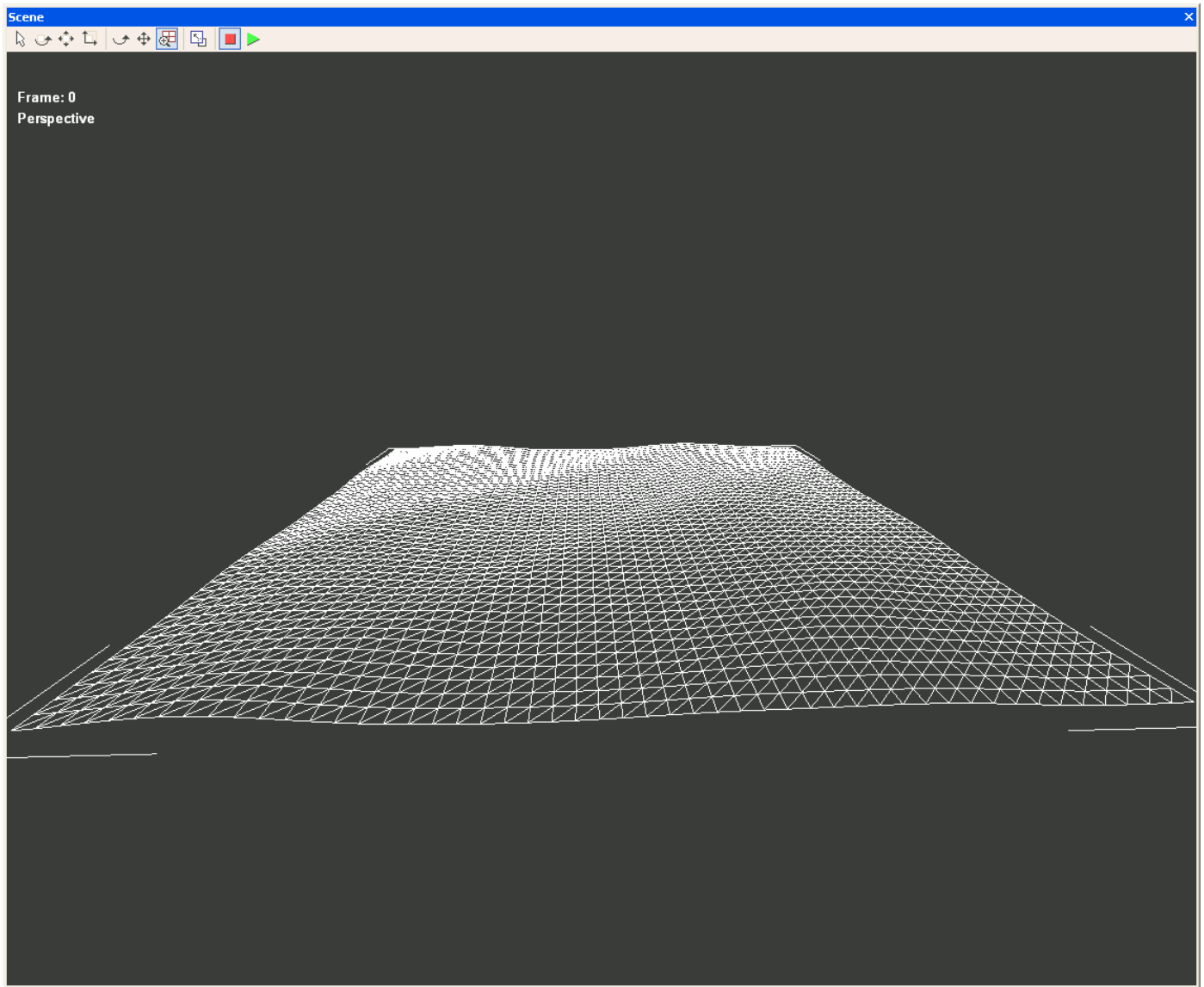
- Finch, M., “Effective Water Simulation from Physical Models”, excerpt from GPU Gems book
- Pelzer, K., “Advanced Water Effects”, excerpt from ShaderX2 book
- Download ”Textures.zip” from the course home page

Waves

- Sum of sines
- Form sharper crest by raising each sine wave to an exponent k (sharpness)
- sin(x) -> (sin(x))^k
- Gerstner Waves is a better approximation, as they move vertices towards crests. See GPU GEMS article for details



Waves



Waves cont.

- A = amplitude
- D = (Dx,Dy) = direction of travel
- f = frequency
- p = phase
- k = sharpness
- t = time
- (x,z) = position on the plane of the water surface
- Gi = one wave with a unique set of parameters
- H = A sum of waves

$$H(x,z,t) = \sum G_i$$

$$\frac{dH}{dx} = \sum (\frac{dG_i}{dx})$$

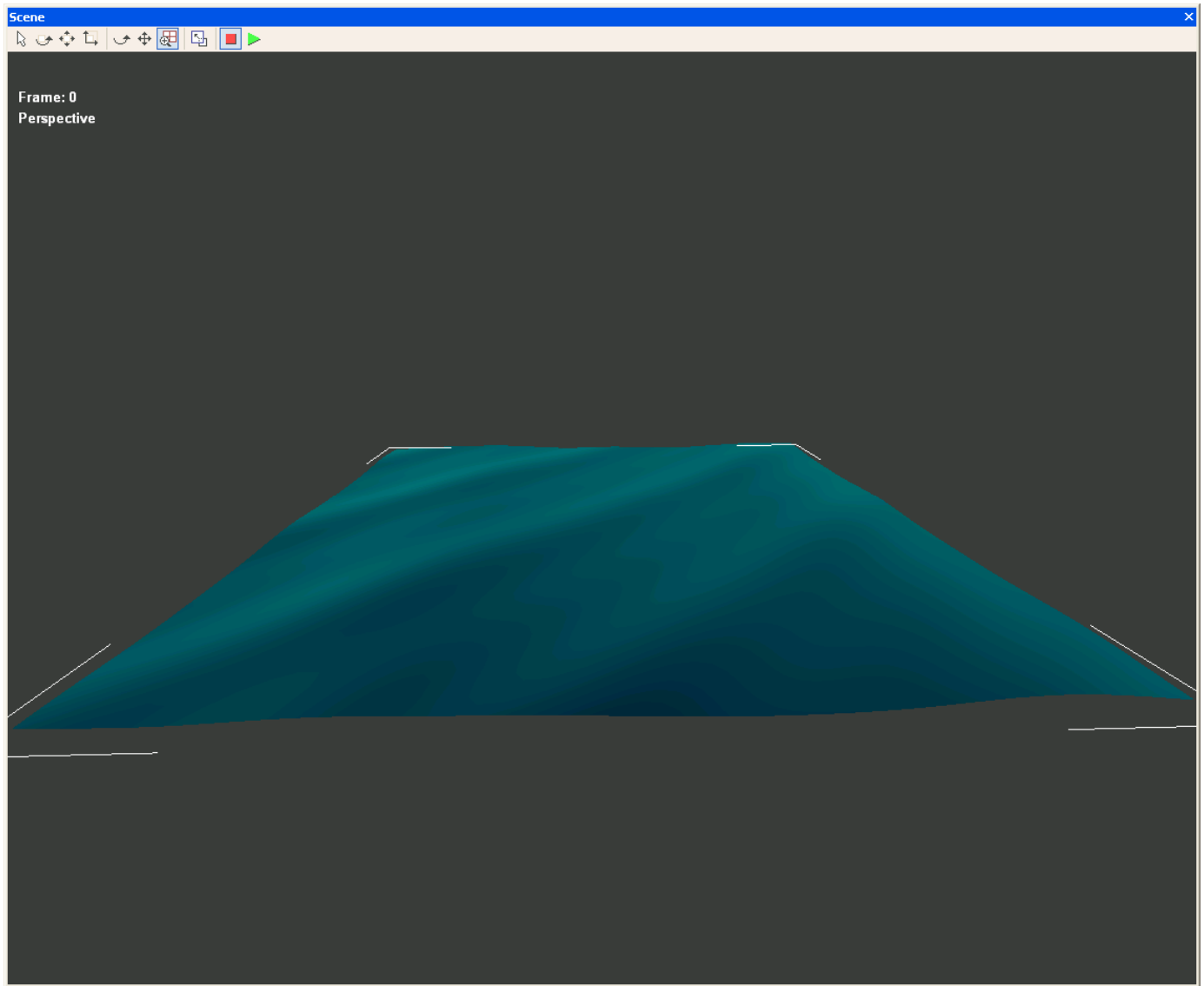
$$\frac{dH}{dz} = \sum (\frac{dG_i}{dz})$$

$$y = G(x,z,t) = A(sin((D_x \cdot x + D_z \cdot z) \cdot f + tp) \cdot 0.5 + 0.5)^k$$

$$\frac{dG}{dx} = 0.5kfA(sin((D_x \cdot x + D_z \cdot z) \cdot f + tp) \cdot 0.5 + 0.5)^{k-1} \cdot cos((D_x \cdot x + D_z \cdot z) \cdot f + tp) \cdot D_x$$

$$\frac{dG}{dz} = 0.5kfA(sin((D_x \cdot x + D_z \cdot z) \cdot f + tp) \cdot 0.5 + 0.5)^{k-1} \cdot cos((D_x \cdot x + D_z \cdot z) \cdot f + tp) \cdot D_z$$

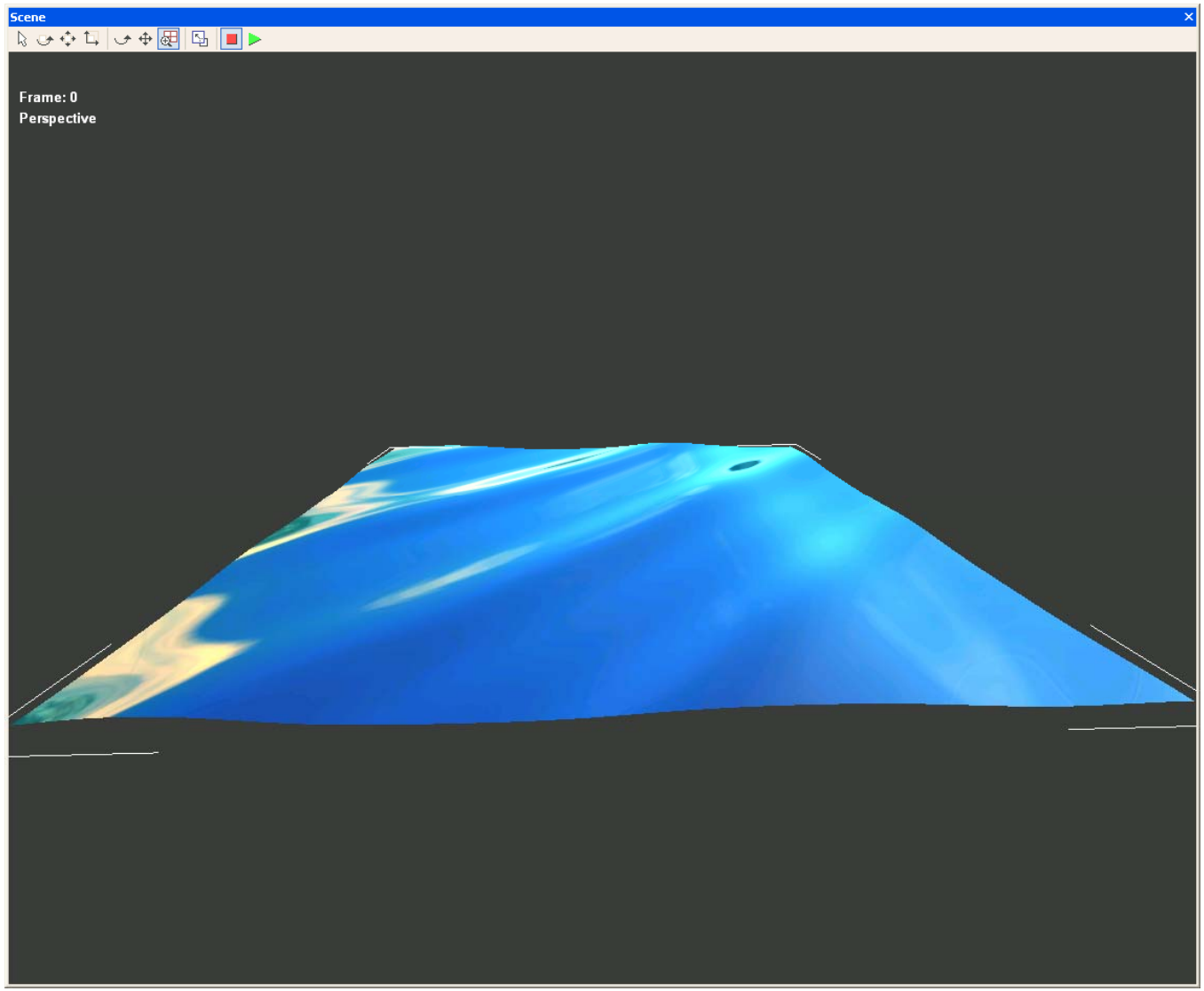
Water Color



Water Color cont.

- Deep color = {0.0f, 0.0f, 0.1f, 1.0f}
- Shallow color = {0.0f, 0.5f, 0.5f, 1.0f}
- facing = 1.0 - max(dot(world eye, world normal), 0)
- normal (expressed in object space) = (-dH/dx, 1, -dH/dz)
- waterColor = lerp(deep color, shallow color, facing)

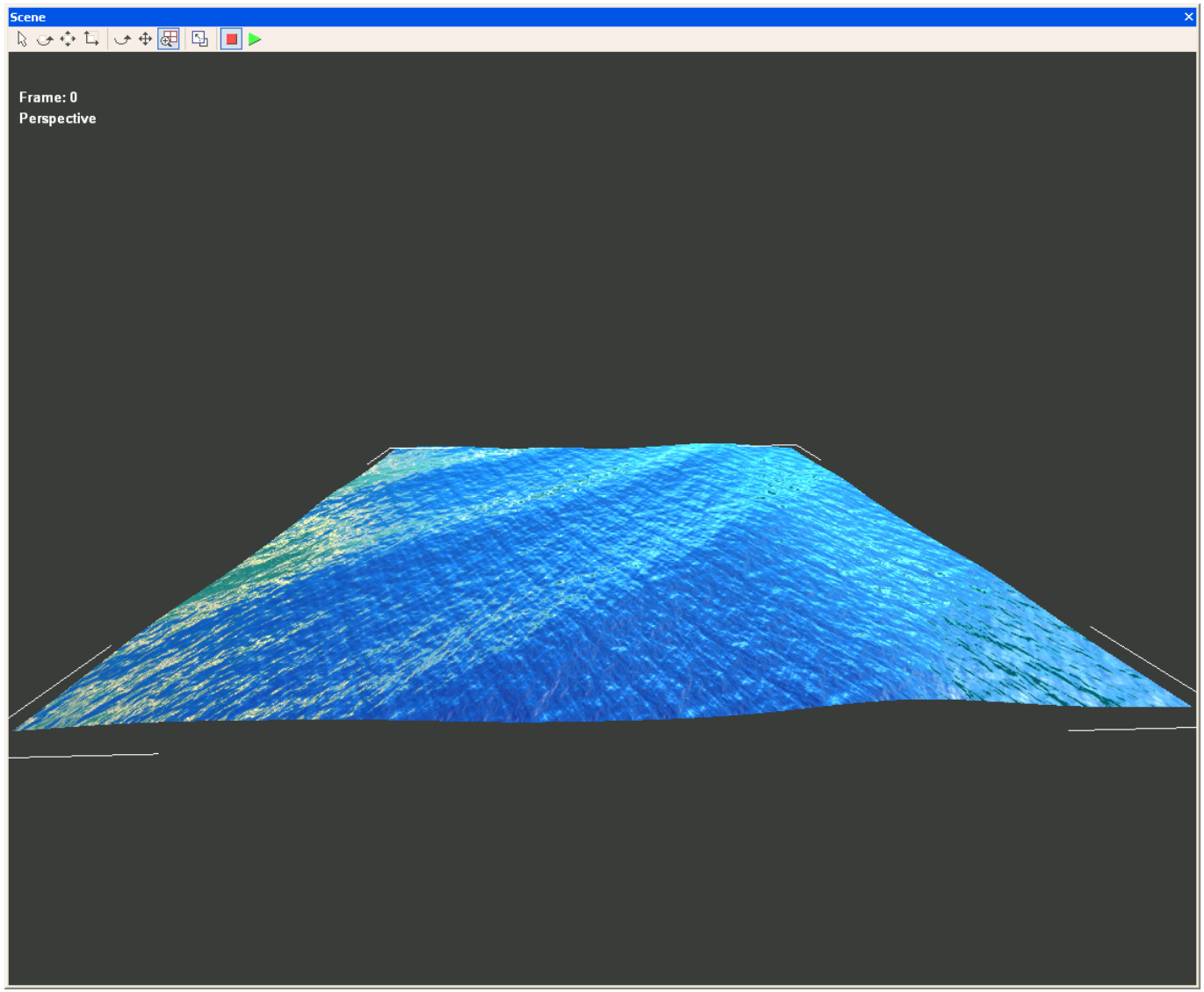
Reflection Mapping



Reflection Mapping

- As in assignment 3
- Normal = (-dH/dx, 1, -dH/dz)
- Used as light source
  - CloudyHillsCubemap2.dds
- color = water color + reflection

Animated Bump Mapping



Animated Bump Mapping

- Use file : waves2.dds

– sampler2D normalMapSampler = sampler_state {

```
Texture = <normalMap>;
MagFilter = Linear;
MinFilter = Linear;
MipFilter = Linear;
};
```

- Use point sampling for sparkles -HACK
  - MagFilter = Linear;
  - MinFilter = Point;
  - MipFilter = None;

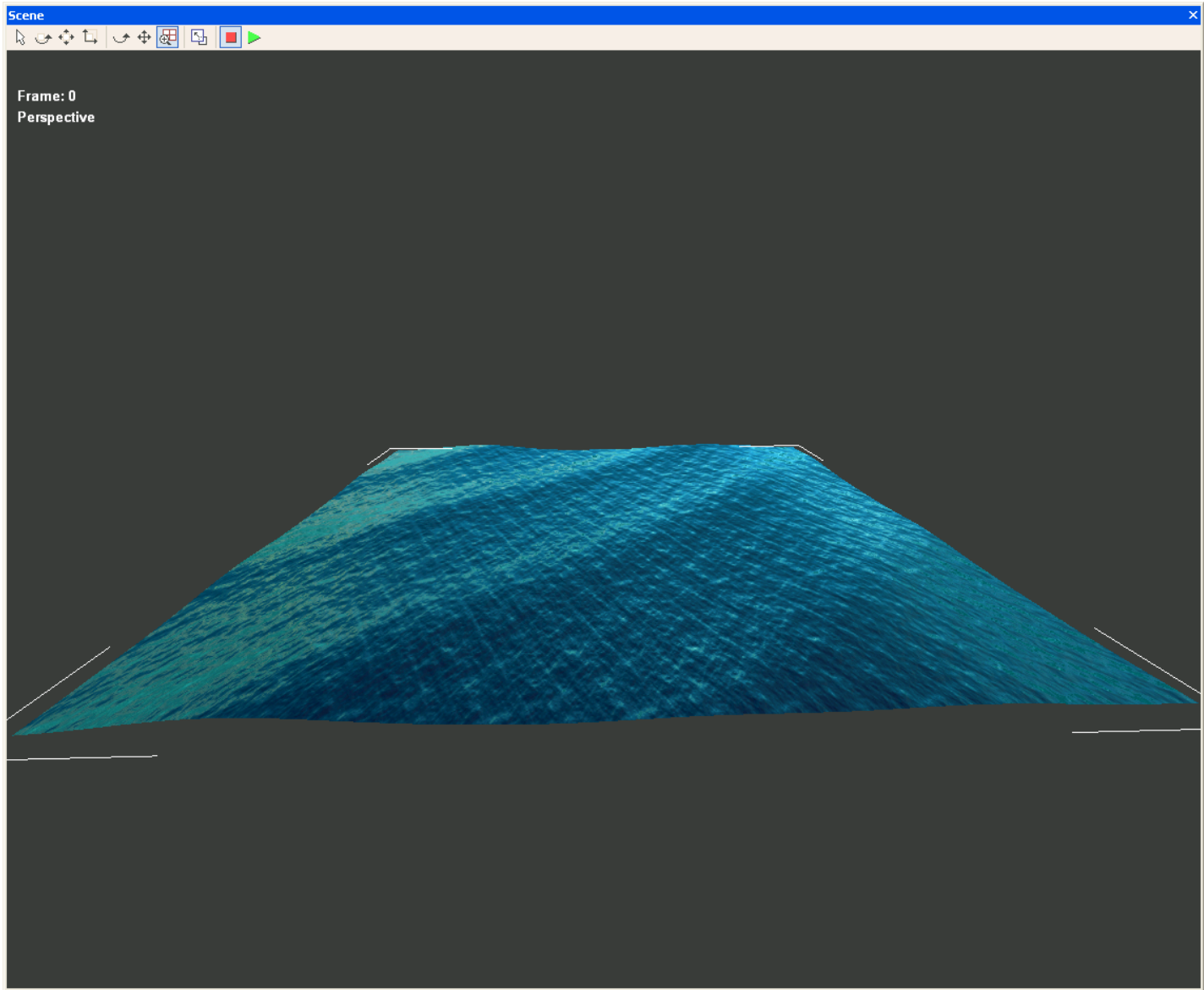
Animated Bump Mapping

- Sliding windows
  - bumptime = fmod(time, 100.0)
  - textureScale = (8, 4)
  - bumpspeed = (-0.05, 0)
  - bumpCoord0.xy = TexCoord.xy*textureScale +bumptime*bumpSpeed
  - bumpCoord1.xy = TexCoord.xy*textureScale*2 + bumptime*bumpSpeed*4
  - bumpCoord2.xy = TexCoord.xy*textureScale*4 + bumptime*bumpSpeed*8

Animated Bump Mapping

- Tangent base coordinate system
  - B = (1, dH/dx, 0)
  - T = (0, dH/dz, 1)
  - N = (-dH/dx, 1, -dH/dz)
- Superposition bump coordinates
  - t_i = tex2d(bumpsampler, bumpcoord_i) - (0.5,0.5,0.5)
- c.xyz = sum(t_i.xyz)

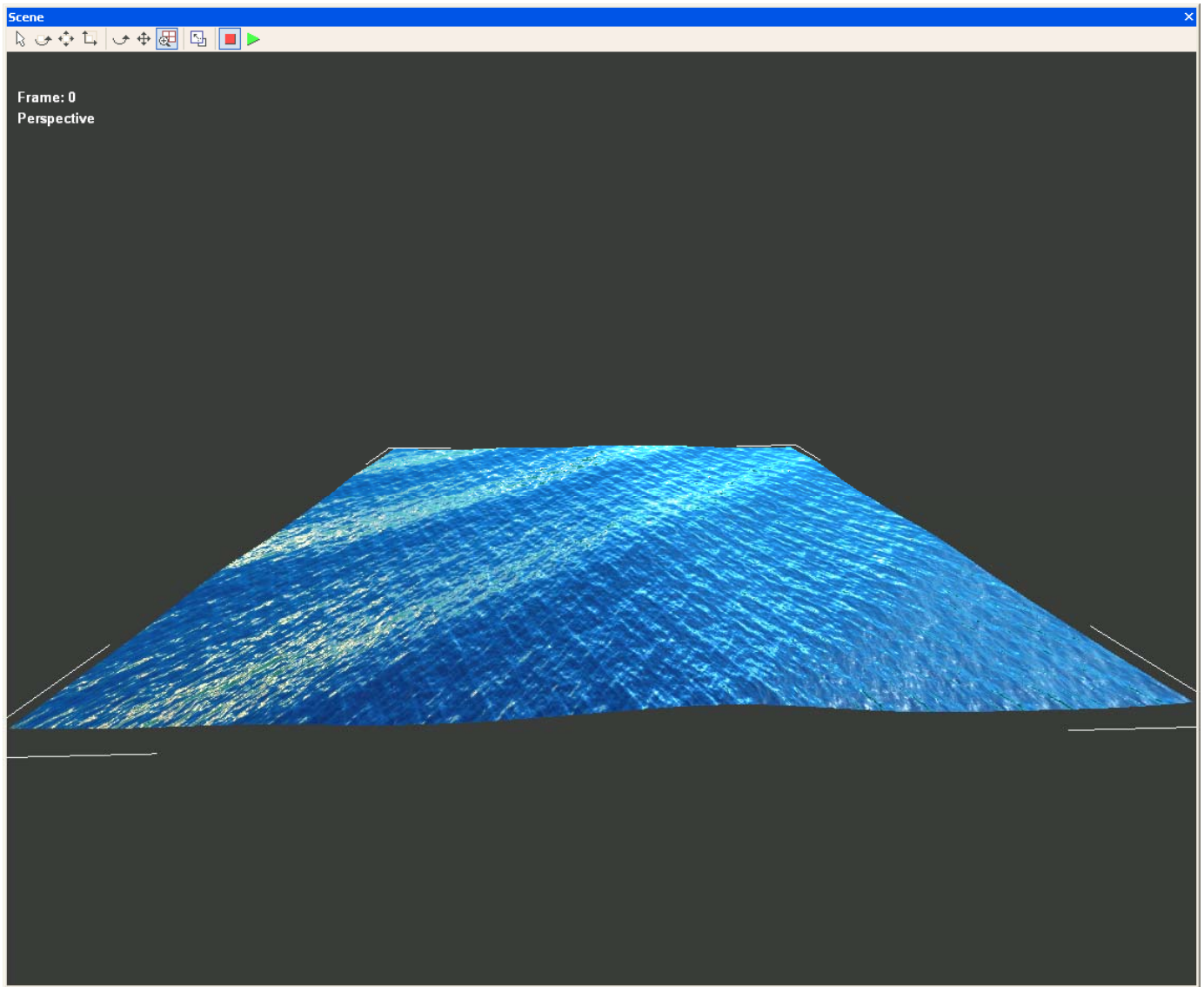
Fresnel Reflection



Fresnel terms

- How much light reflects at a material boundary (and ~ 1 - how much refracts)
  - fastFresnel = R(0) + (1-R(0))*pow(1.0 - dot(viewerDirection, normal), 5.0)
  - R(0) = ((n1 – n2)/(n1+n2))^2
  - Air to water boundary: R(0) = 0.02037
- color = water color + reflection*fresnel

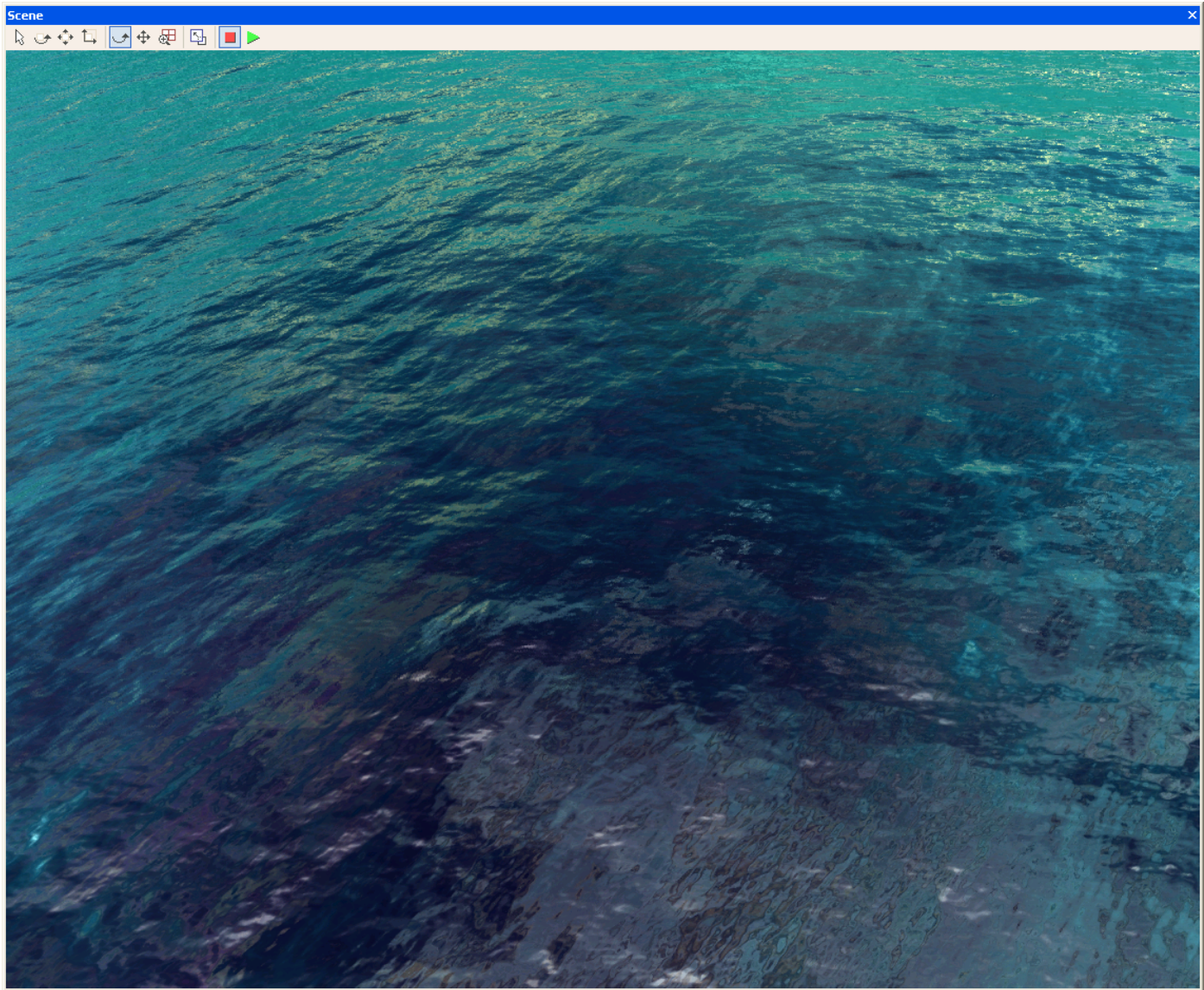
HDR lighting



HDR lighting

- Real light has large dynamic range
- Extra lighting encoded in the cube map’s alpha channel
- hdrMul ~ 3
- reflection.rgb *= (1.0 + reflection.a*hdrMul)

Refraction



Refraction

- color = waterColor + reflection*fresnel + refraction*(1-fresnel)
- Refraction index for water is 1.33 (thin->dense) or 1/1.33 (dense->thin)
- Use the intrinsic “refract”

Final Result