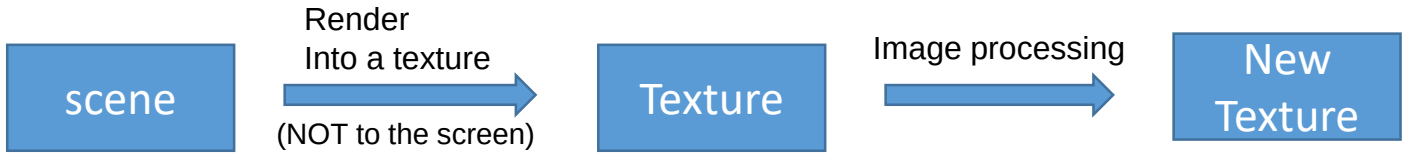
Post Processing

What Is Post Processing

- Idea



- **Image processing** is a well studied, theoretically rich subject in computer science
  - a huge number of image processing techniques can be used
    - to further **improve** the rendering quality of a scene
      - Smoothing, sharpening, denoising, …
    - or to alter the rendering result to achieve new visual effects

Applications of Post Processing

- Image transformation
- Image smoothing and sharpening
- Glowing effect
- God’s Rays
- Edge detection
- Deferred rendering
- Soft shadows
- Antialiasing
- … …

Image Deformation Using a Normal Map

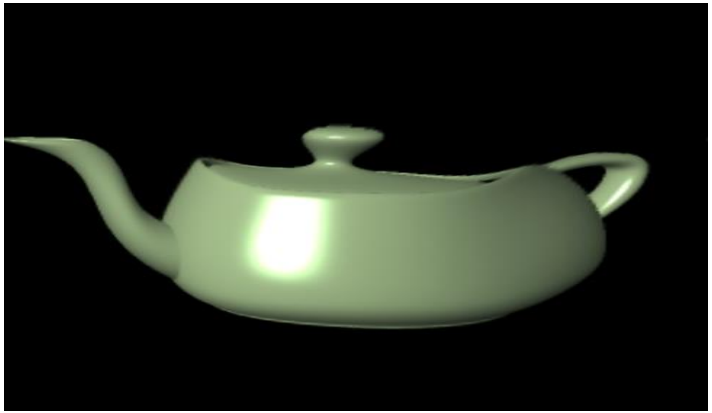
- Imagine that you are viewing the scene through a transparent object with patterned surface, such as a patterned glass or water
- Depending on the patterns and the medium properties of the translucent object, your view will be distorted somewhat
- Such a phenomenon can be animated based on the surface normal of the translucent object
  1. Render the scene into a texture
  2. Load normal map representing the surface normal of the translucent object
     - Remark: an ordinary image can be used to calculate the surface normal if a premade normal map is not available
  3. Distort texture coordinates using the normal vector read from the normal map

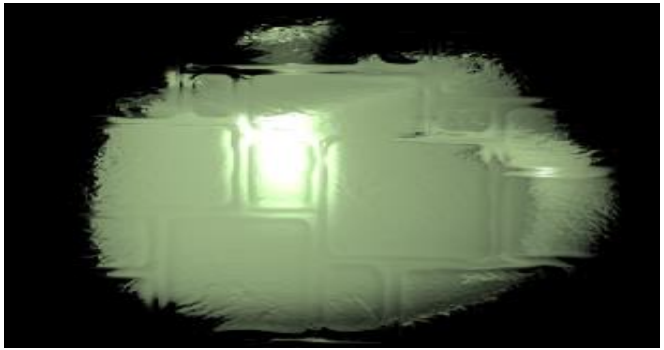
Image Deformation Using a Normal Map

```
void main(void)
{
   … …
   vec2 Translt = texture2D( NormalMap, Density*TexCoord ).xy;
    Translt  - =0.5;

    vec2 newTex = TexCoord + Scale* Translt.xy;
    gl_FragColor = texture2D( Base, newTex );
}
```

Image Deformation Using a Normal Map





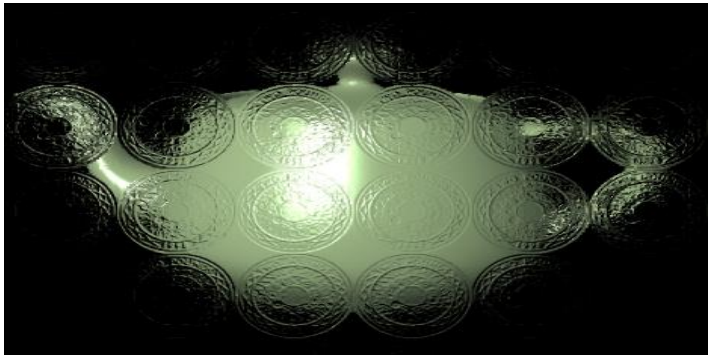
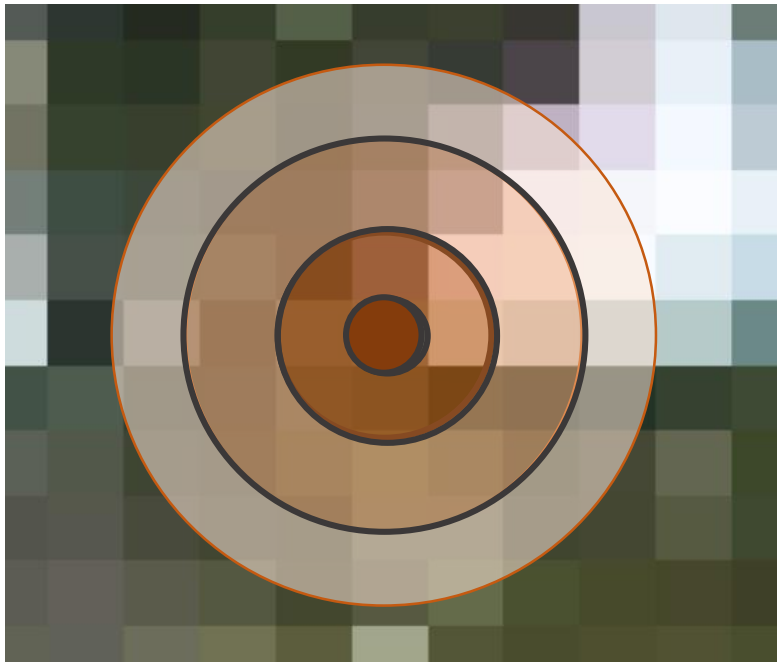
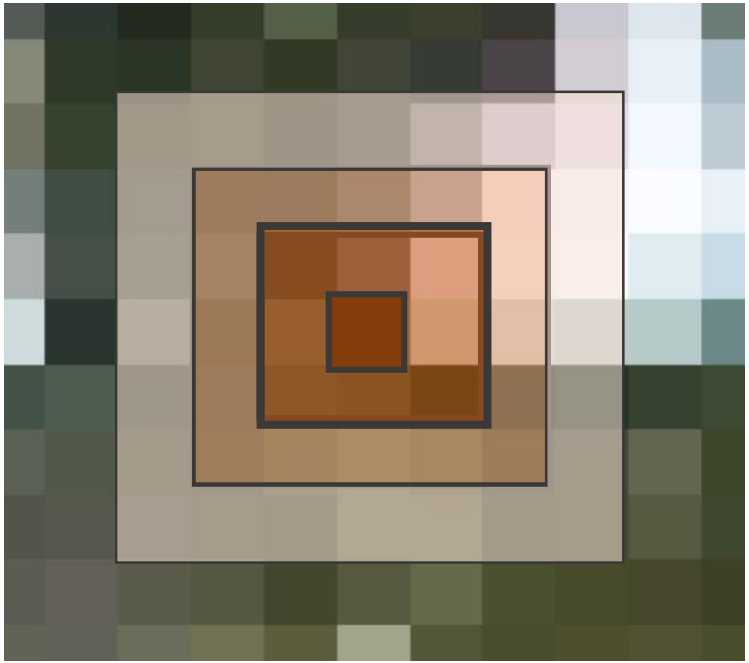


Image Smoothing

- Very useful for
  - Antialiasing
  - Generating soft shadows
  - Creating glowing and burning effects
- Various methods
  - Using averaged colour of neighbour pixels
    - Squared regions
      - 2x2, 3x3, 4x4, …
    - Circular regions
      - Various radii
  - Different neighbouring colours may be weighed differently
    - Eg, using Gausssian function

Circular Regions vs. Squared Regions





Texture Smoothing

- For example, an image smoothing method based on squared region of 5x5 with equal weights can be easily implemented in the following way:

```
vec4 texSmoothing(sampler2D Texture0, vec2 uv)
{
  vec4 smoothedCol=vec4(0.0);
  for (int i=-2; i<=2; i++){
    for (int j=-2; j<=2; j++){
      smoothedCol += texture2D(Texture0,  uv + vec2(i, j)*pixelSize );
    }
  }

  return smoothedCol/25.0;
}
```

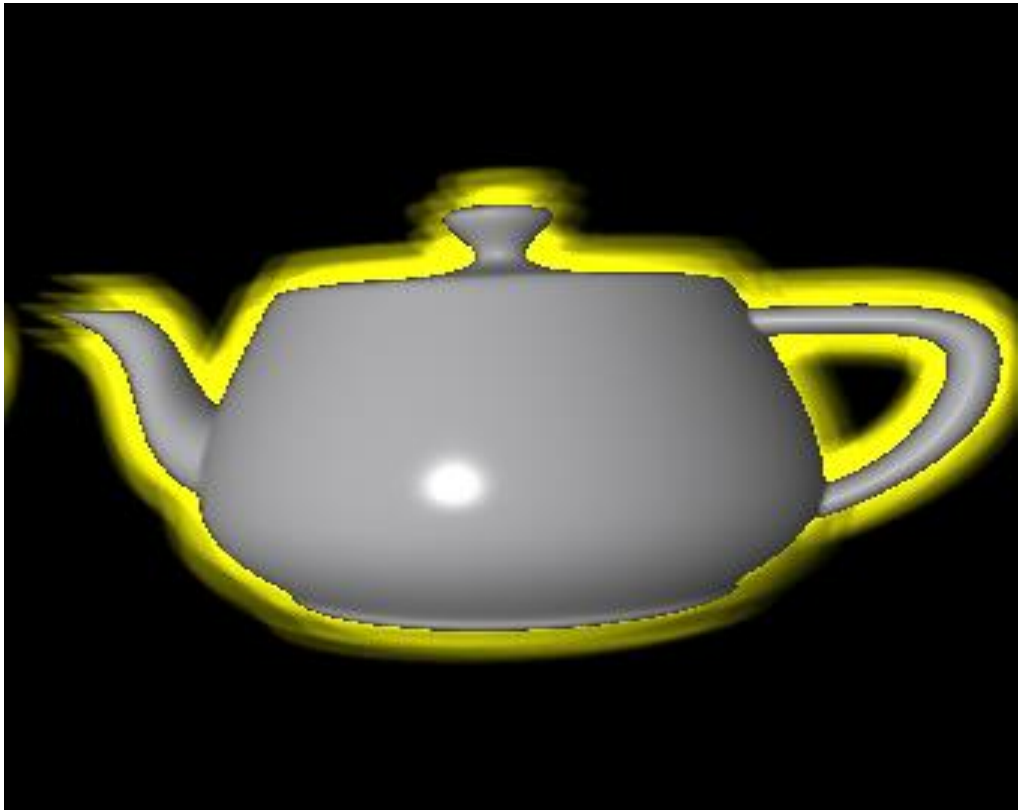
Texture Smoothing

Application: Glowing Effect

- Render the effect into a texture
- Apply a texture smoothing technique to the texture
- Blend the original image and the smoothed image. For instance,

```
vec4 teapot=texture2D (teapotMap, Texcoord);
vec4 teapotSmoothed=texSmoothing(teapotMap, Texcoord);
if(teapot.r>0.1)
    gl_FragColor=teapot;
else
    gl_FragColor = teapotSmoothed*GlowColor*Intensity;
```

Make an Objects Glow

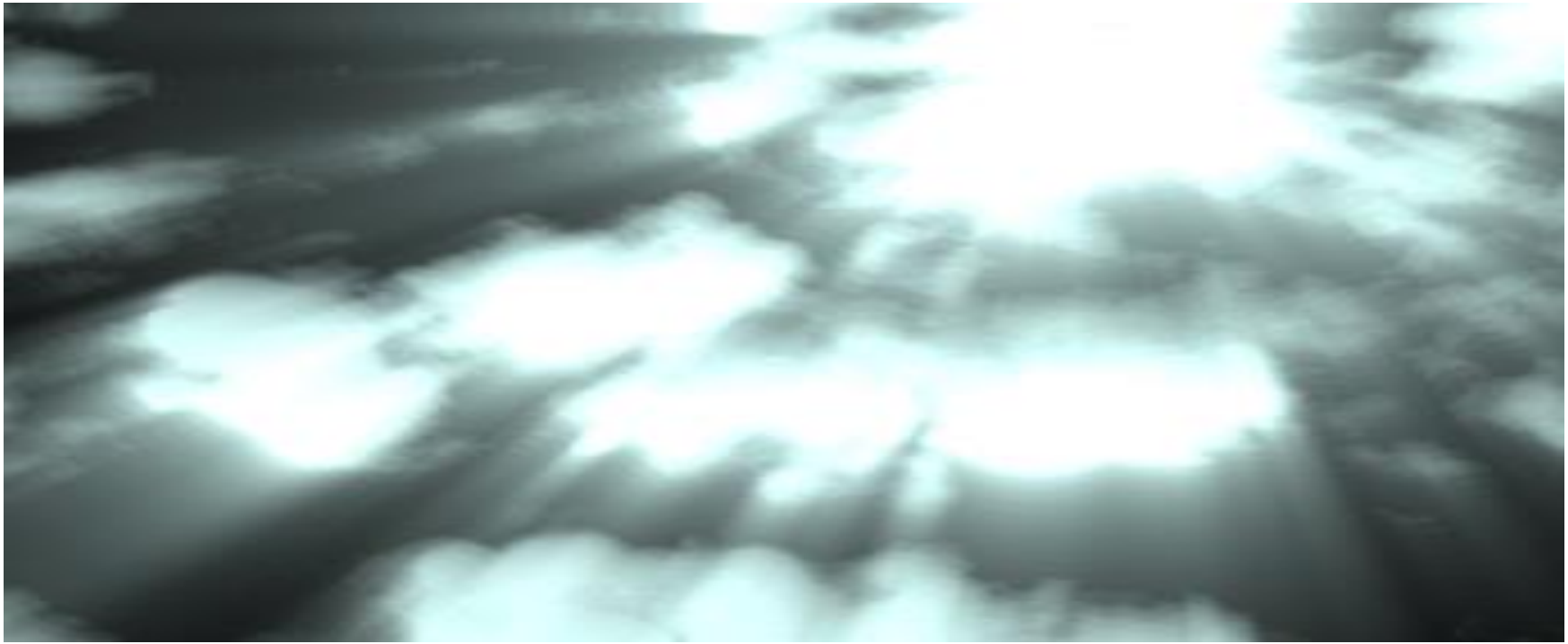


Make an Objects Burning

Put it Together

- Map the original and processed texture to a screen aligned quad model
- Reset the quad size to make it have the same aspect ratio with the viewport size
- Configure render states to combine the glowing object with other objects in the scene properly

God’s Rays



Light Scattering Model

- Light energy decal model

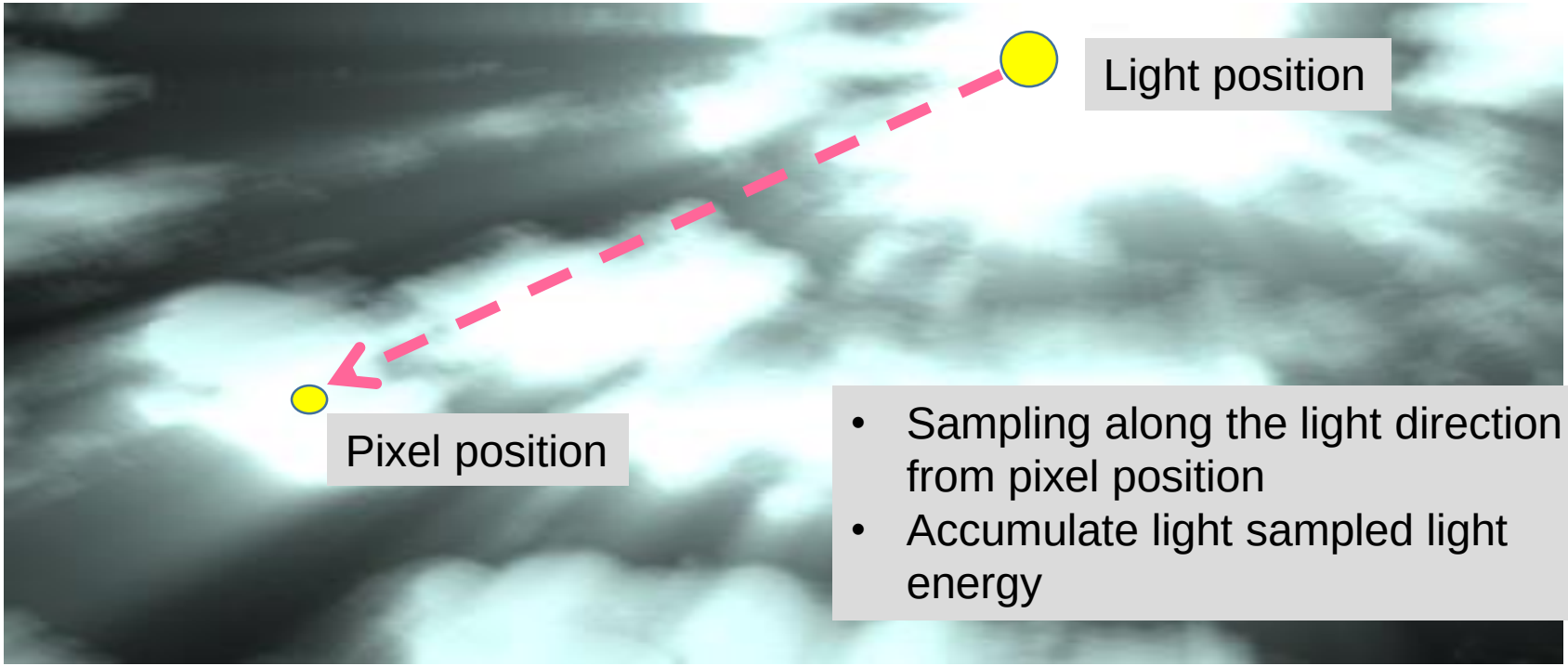
$$\frac{dE(t)}{dt} = -\lambda E(\mathrm{t})$$

- The solution to this equation is

$$E(t) = E_0\, e^{-\lambda t}$$

  - Here $E(t)$ is the light energy at time *t*, and $E_0 = E(0)$ is the light energy at t=0, the initial light intensity

God’s Rays



- Sampling along the light direction from pixel position
- Accumulate light sampled light energy

God’s Rays

```
vec4 GodRays(vec2 texCoords,
                        vec2 pos) {
   vec2 st = texCoords.xy;
   vec2 rayDir = normalize(st - pos.xy);
   vec4 color=vec4(0.0);
   vec4 sampleColor;
```

```
 for (int i = 0; i < NUM_SAMPLES; i++)
    {
        st -= delta*rayDir;
        sampleColor = texture2D(Texture, st);
        sampleColor *= exp(-decay*length(st-pos.xy));
        color += weight*sampleColor;
    }
     return color*E0;
}
```

Questions?