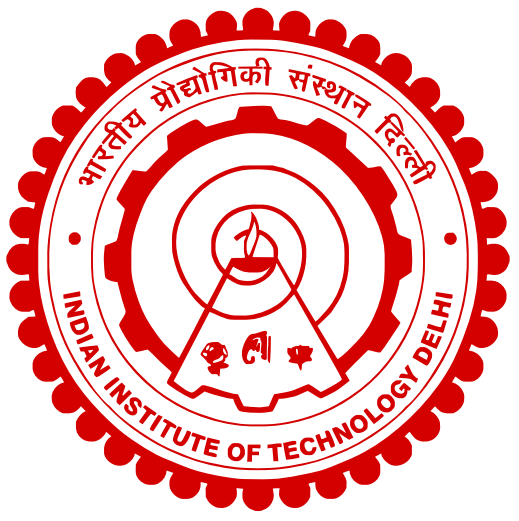


Computer Graphics

Textures

Subodh Kumar, Dept of Computer Sc. & Engg., IIT Delhi



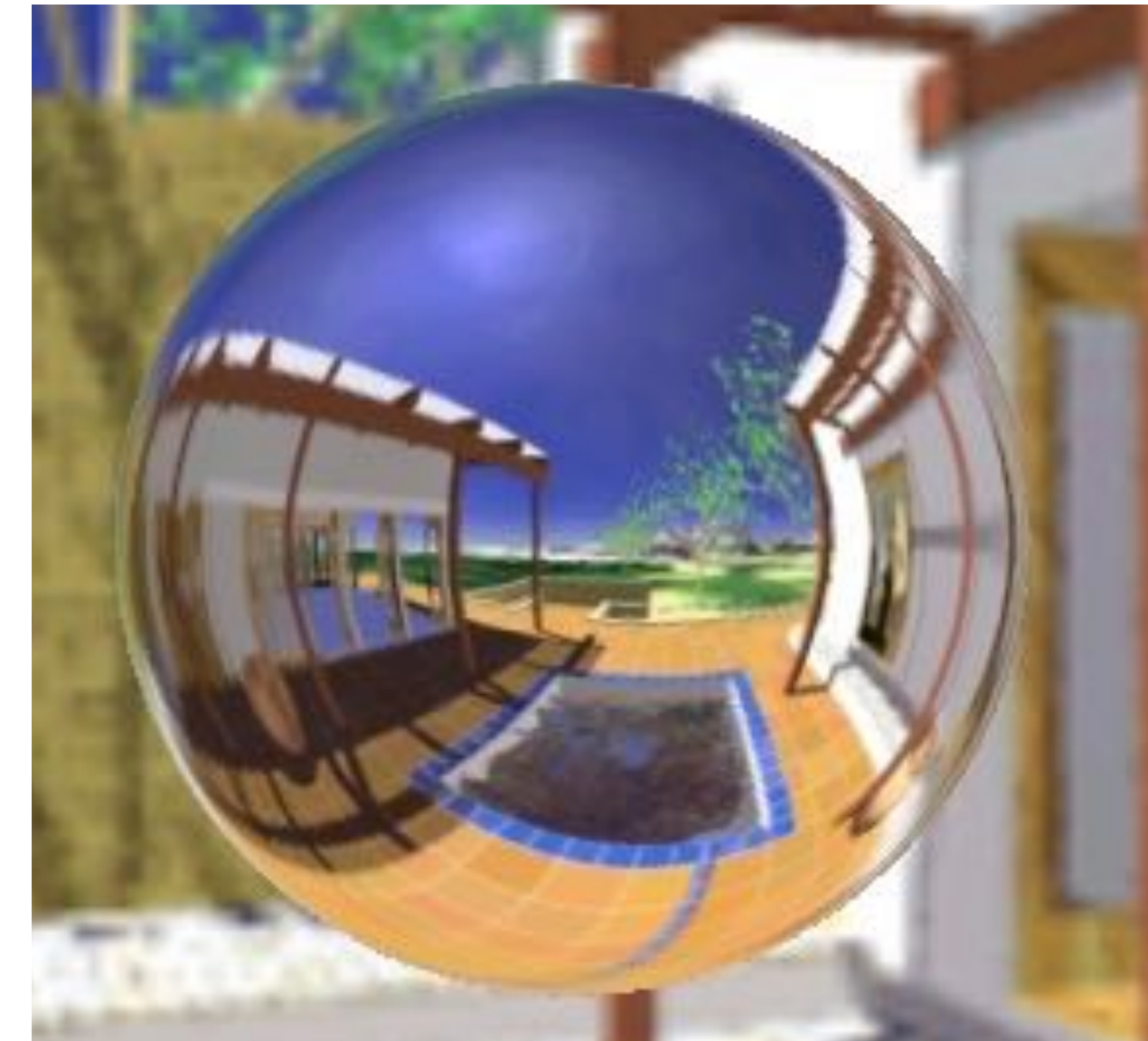
Point Sprites

- Draw points
 - ➔ E.g., as a $k \times k$ pixel square
- Map a (small) texture on the point
- Typically used with transparency
- Examples:
 - ➔ Cursor
 - ➔ Particle Systems, Snowflakes
 - ➔ Imposters/Billboards
 - ➔ Firebreathing Balrog

Environment Mapping

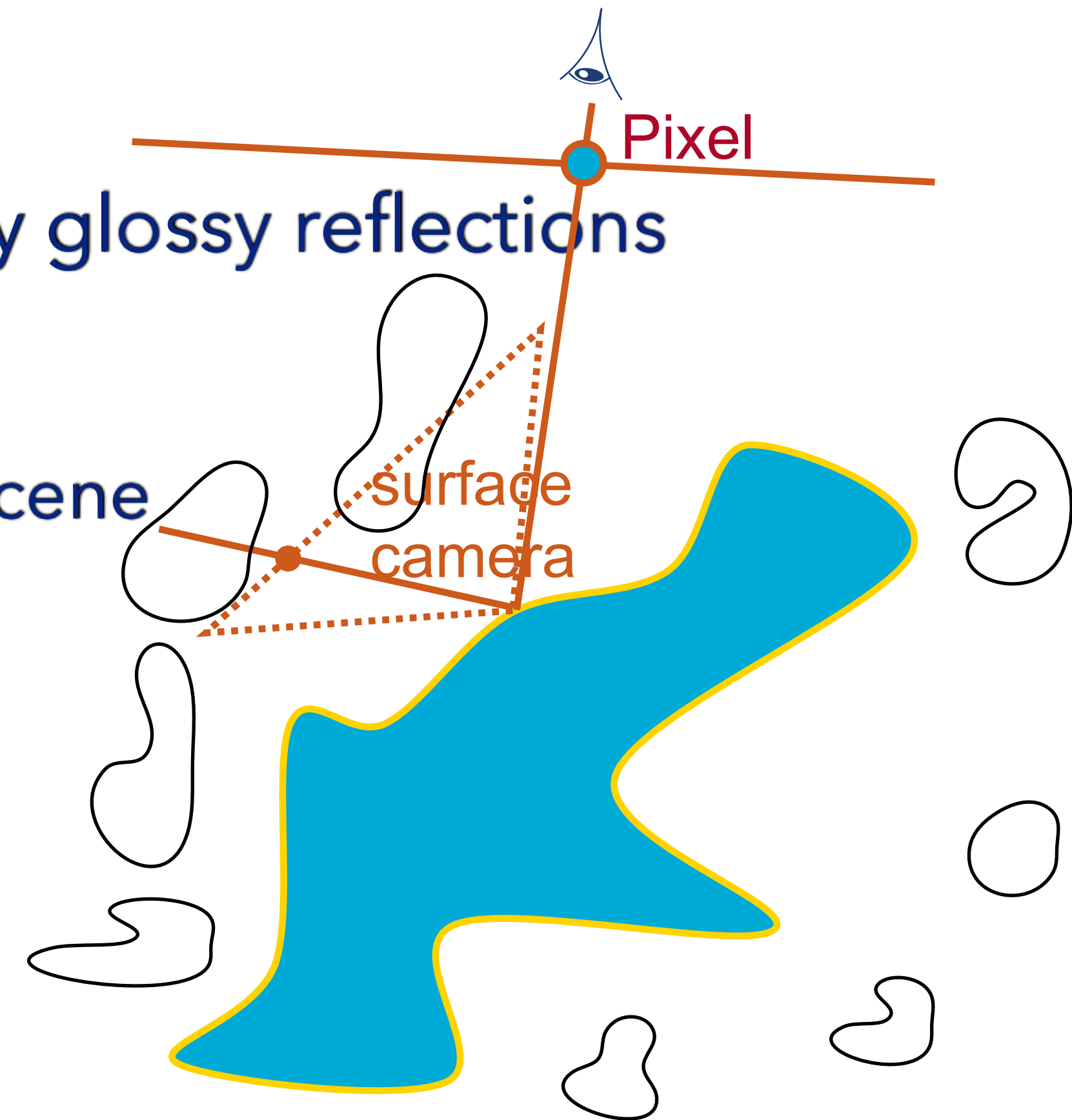


- Environment/Reflection map simulates highly glossy reflections

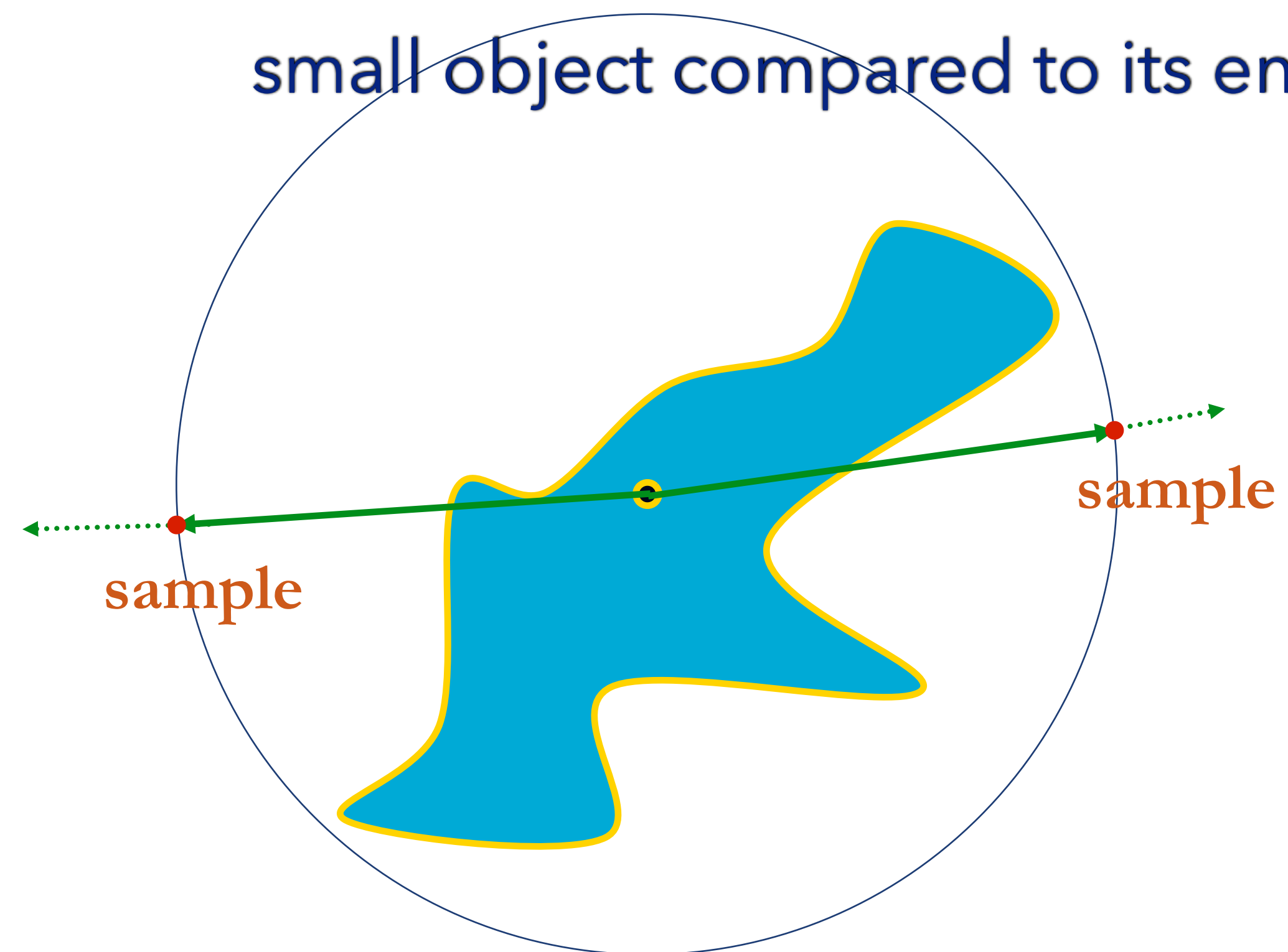


Environment Mapping

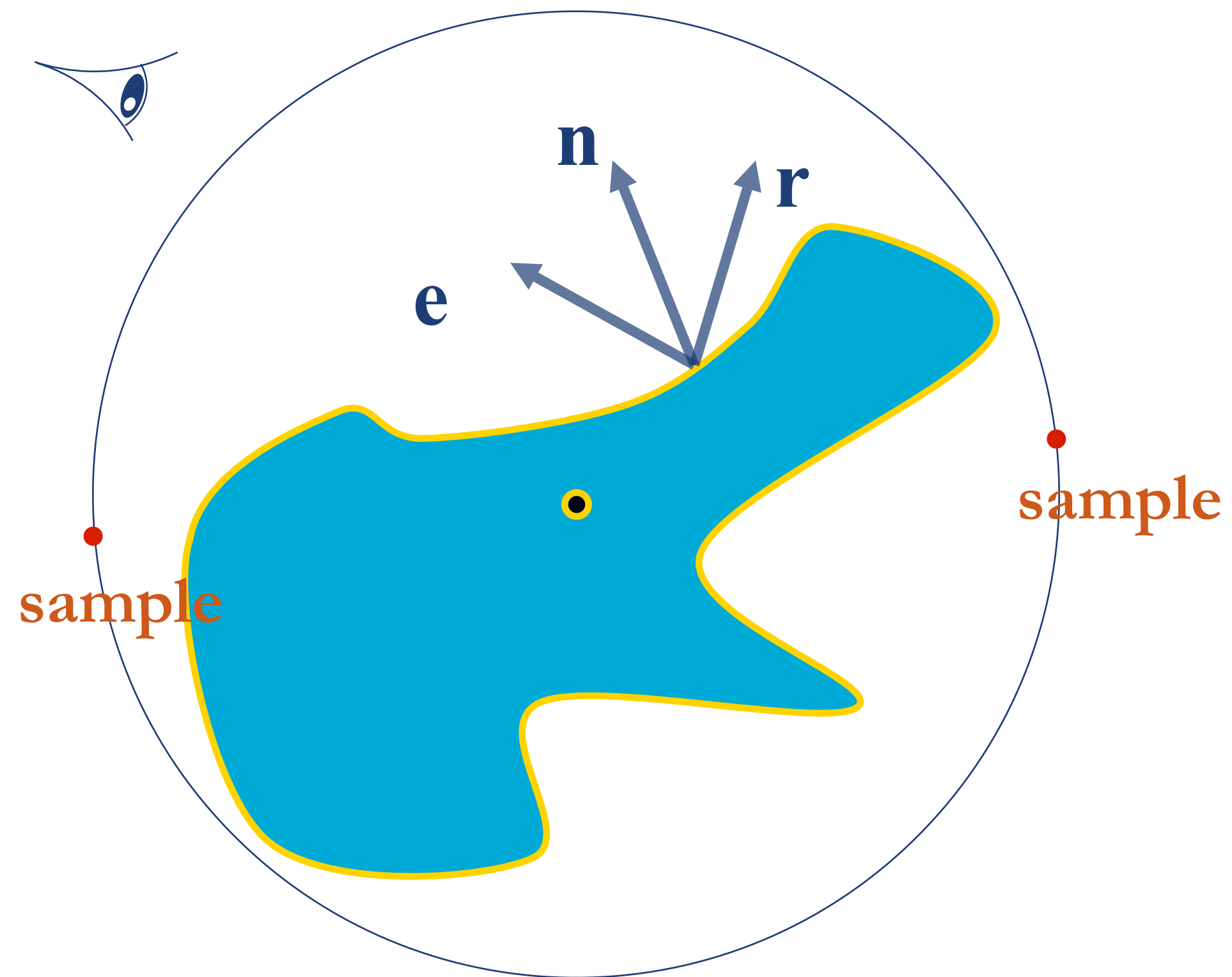
- Environment/Reflection map simulates highly glossy reflections
- Poor-man's global illumination:
 - ➔ From "each point" on the surface, render the scene
 - ➔ Find ideal reflected direction for each views
 - Look-up the color in that direction
- Approximation: use single rendering
 - ➔ Eye at the center of the object
 - Introduces distortions in the reflection
 - Good for small objects in a large room



Environment Mapping

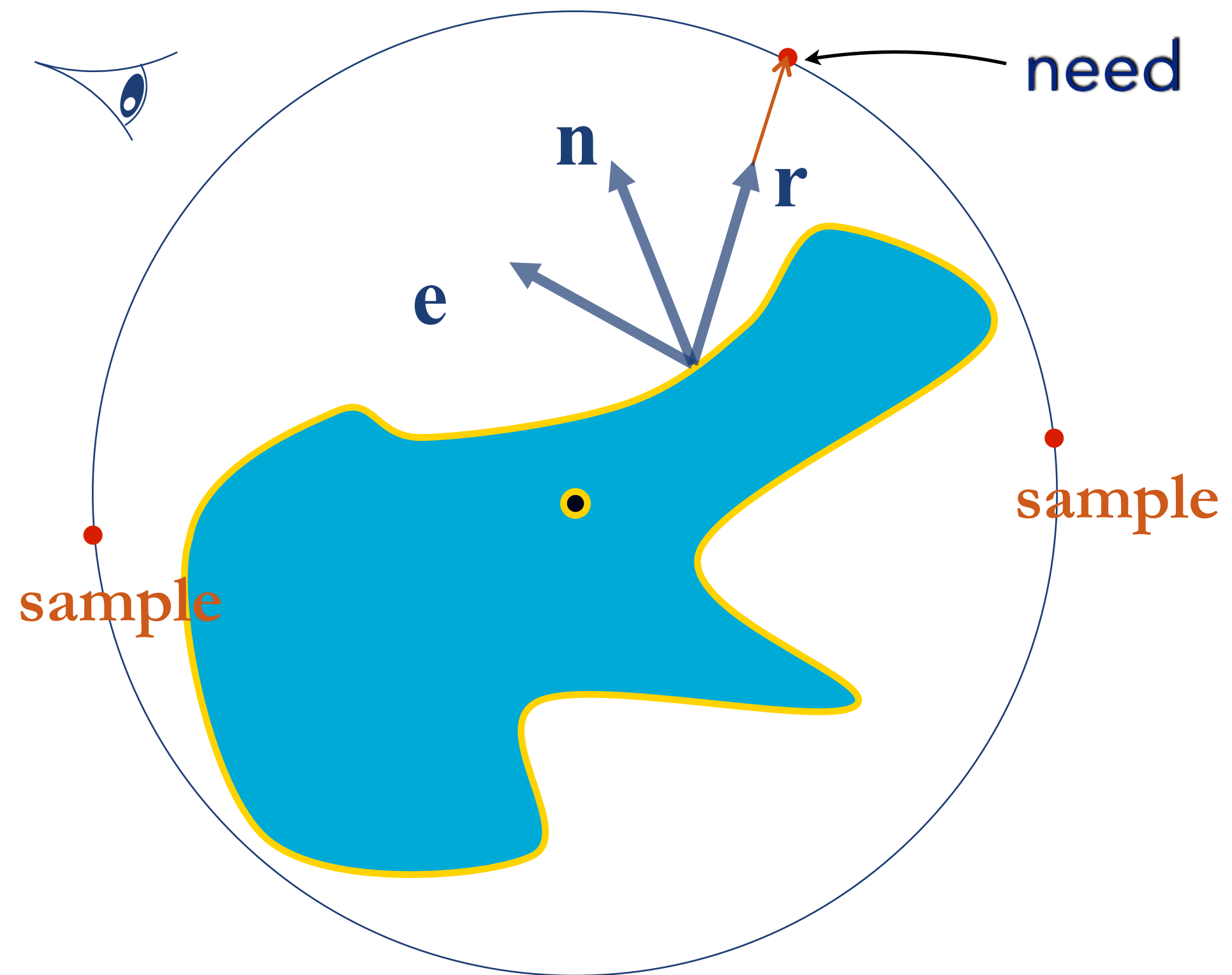


Environment Mapping



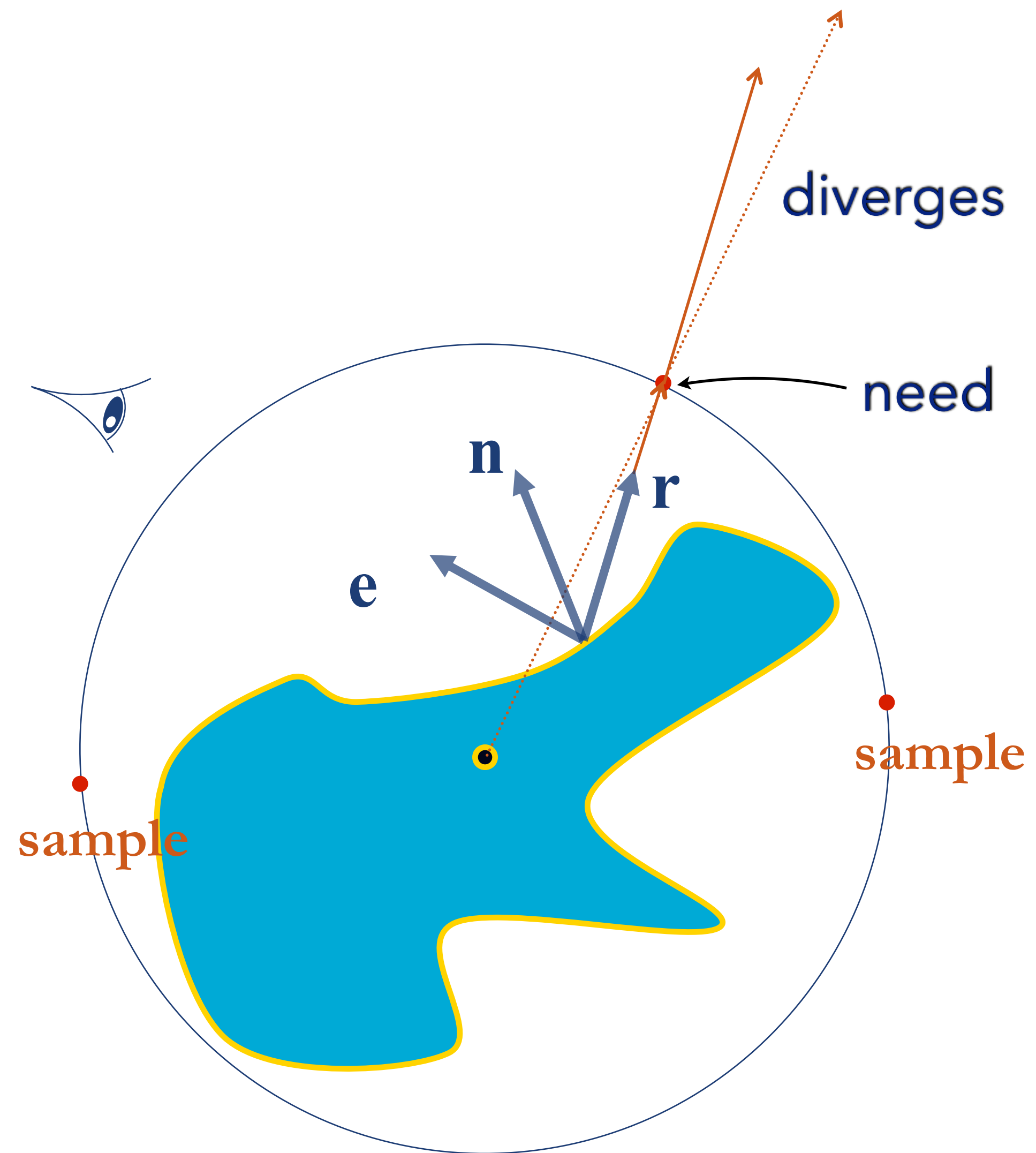
$$r = 2(e \cdot n)n - e$$

Environment Mapping



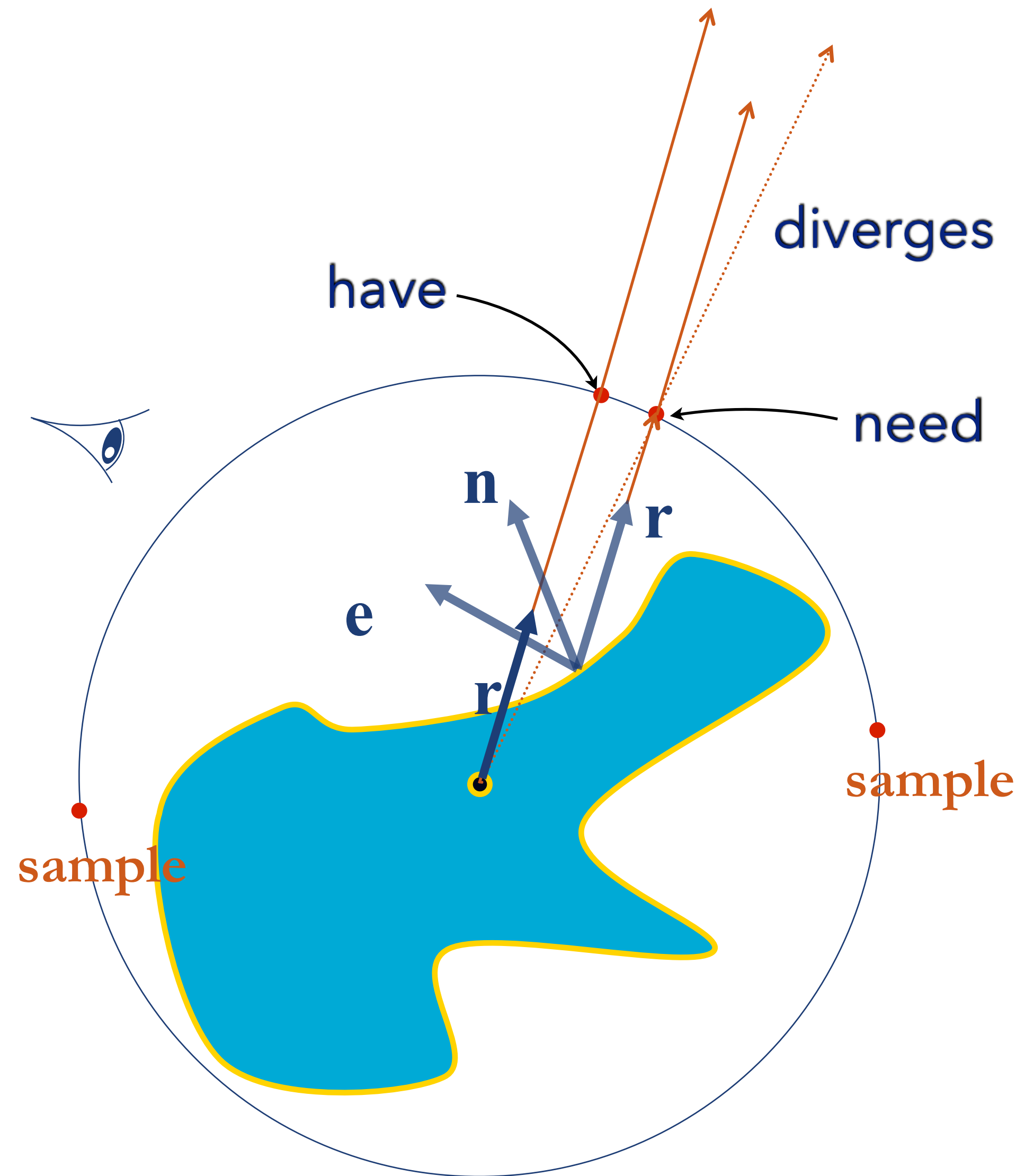
$$r = 2(e \cdot n)n - e$$

Environment Mapping



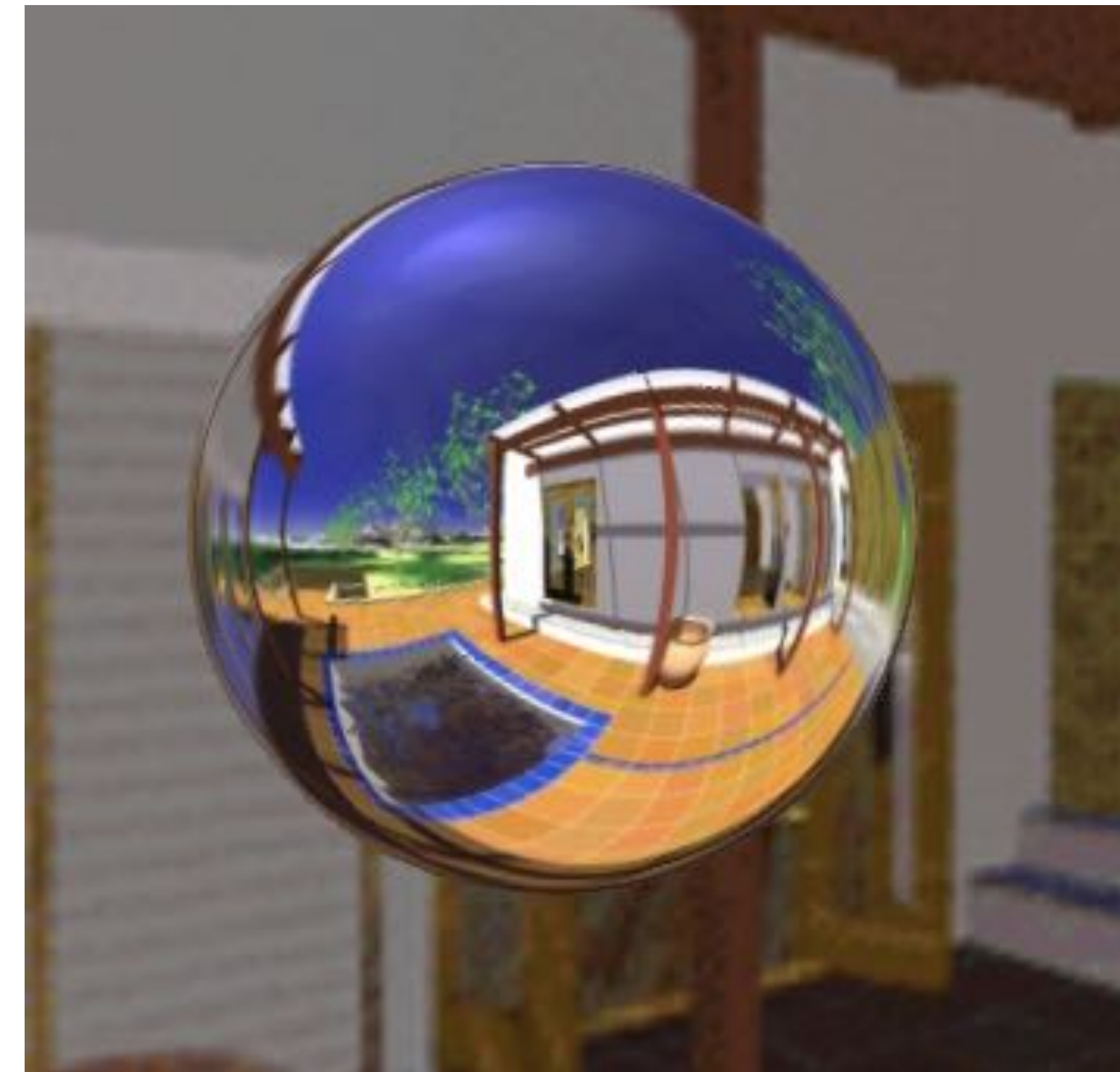
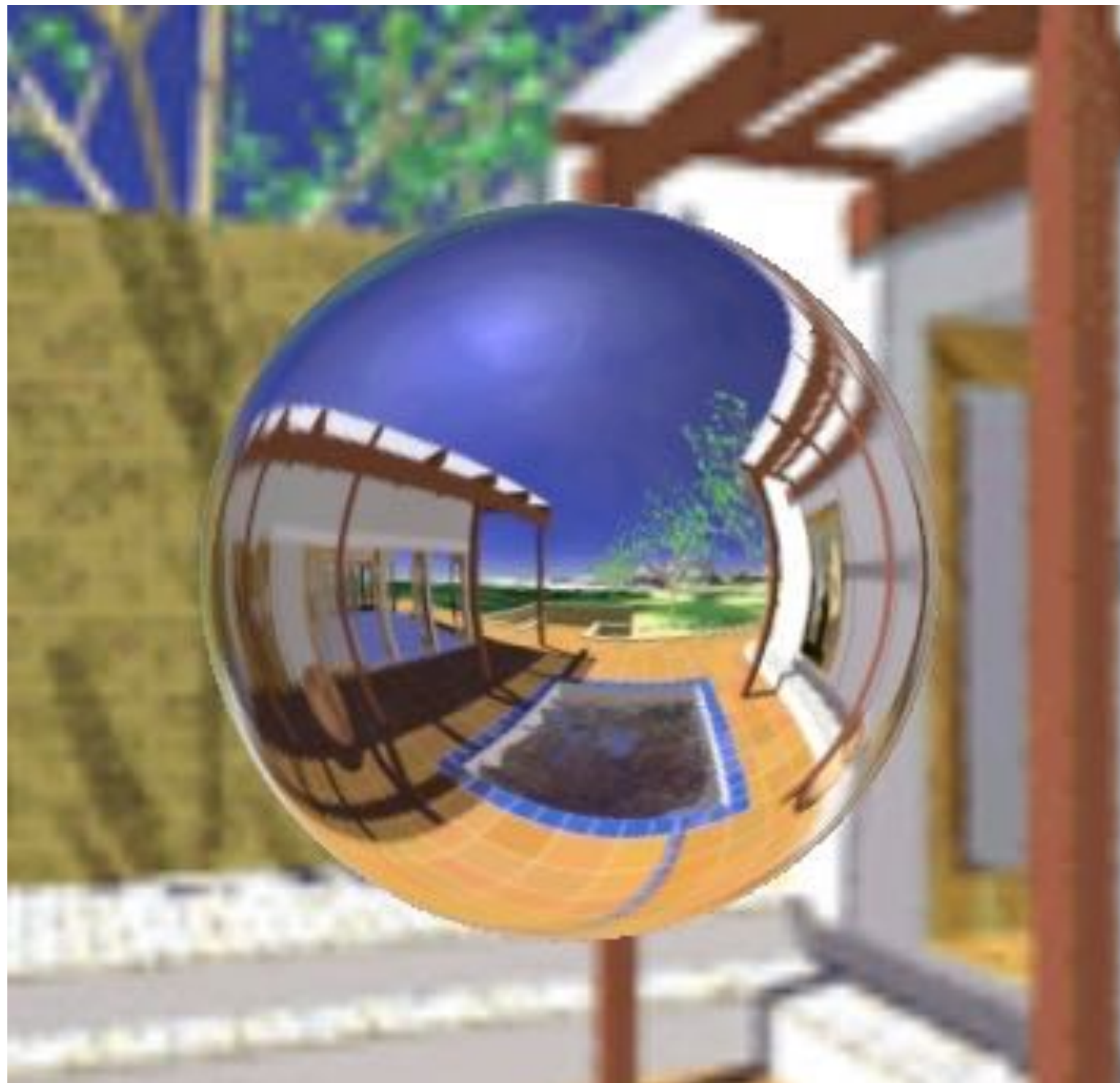
$$r = 2(e \cdot n)n - e$$

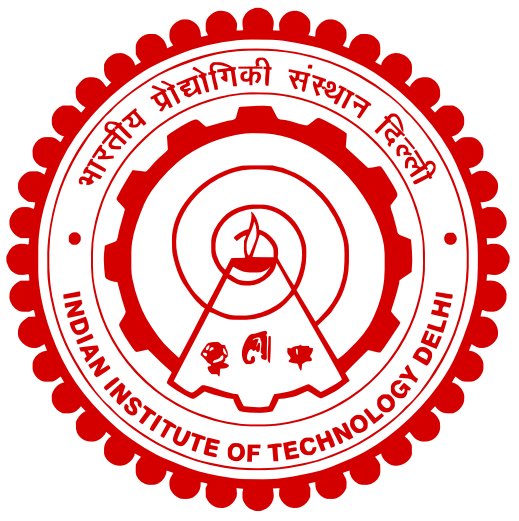
Environment Mapping



$$r = 2(e \cdot n)n - e$$

Environment Mapped Example

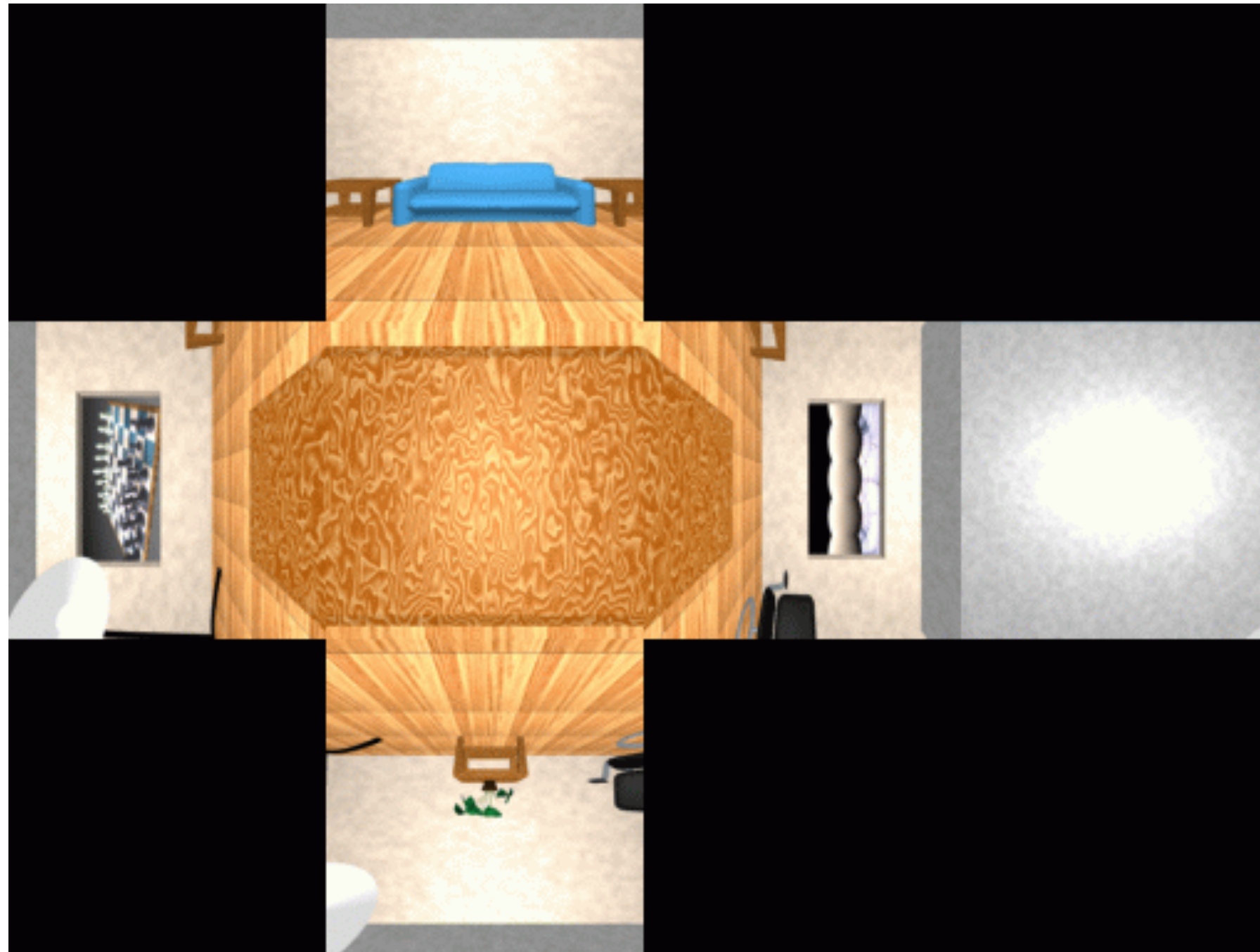




Cube Mapping

- Map onto surfaces of a cube (around the object)
 - ➔ Usually axis-aligned
- Find desired reflected/refracted direction "r" from a surface point
- Cube map is directly accessed by the direction
 - ➔ One of the six faces of the cube intersects, the corresponding texture is looked up
- For environment mapping:
 - ➔ Render scene from object-center onto cube faces
 - Use, e.g, the six axis-aligned directions
 - Rendering may even be "pre-computed" at a high quality
 - E.g., 6 pictures of a real environment

Cube Map Example





Cube vs Sphere Mapping

- Less distortion than sphere environment mapping
- Typically a smaller number of sample (texels) are sufficient
- More efficient
 - ➔ No inverse trigonometric functions
- Generate six views
 - ➔ Each is render to texture
 - ➔ Can use `gl_Layer` in geometry shader to replicate geometry 6-way
 - ➔ Supported by hardware

```
vec3 I = normalize(Pos - cameraPos);  
vec3 R = reflect(I, normalize(Norm));  
FragColor = vec4(texture(skybox, R).rgb, 1.0);
```



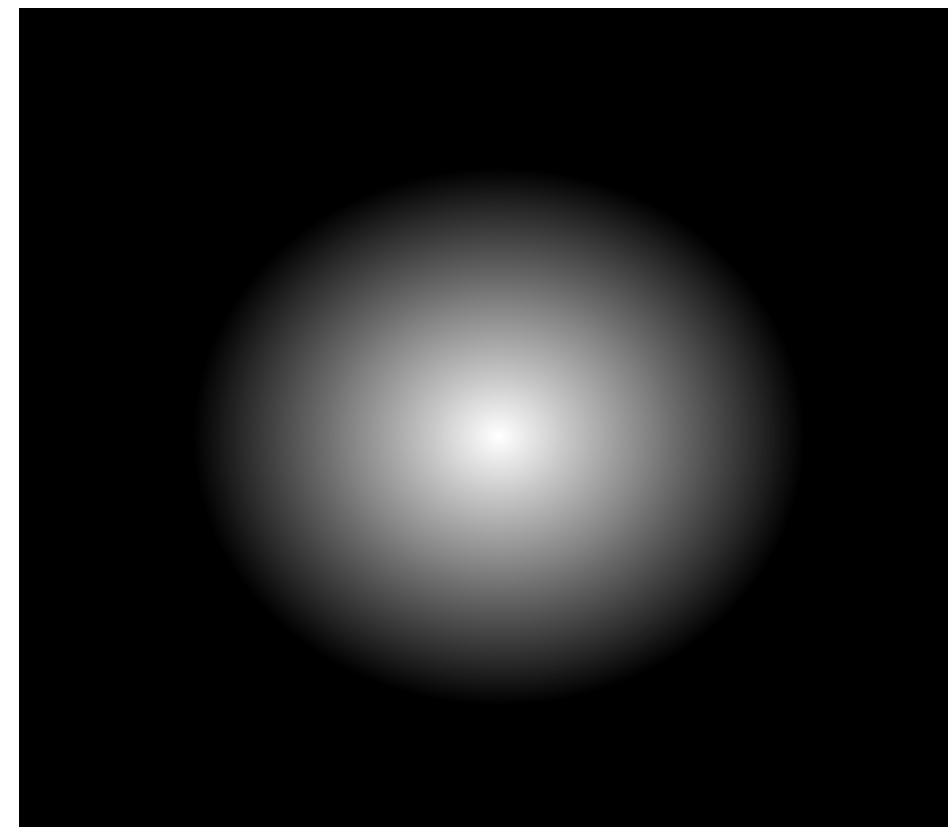
Cube vs Sphere Mapping

```
glBindRenderbuffer(GL_RENDERBUFFER, rbo);  
glRenderbufferStorage(GL_RENDERBUFFER, GL_DEPTH_COMPONENT24, W, H);
```

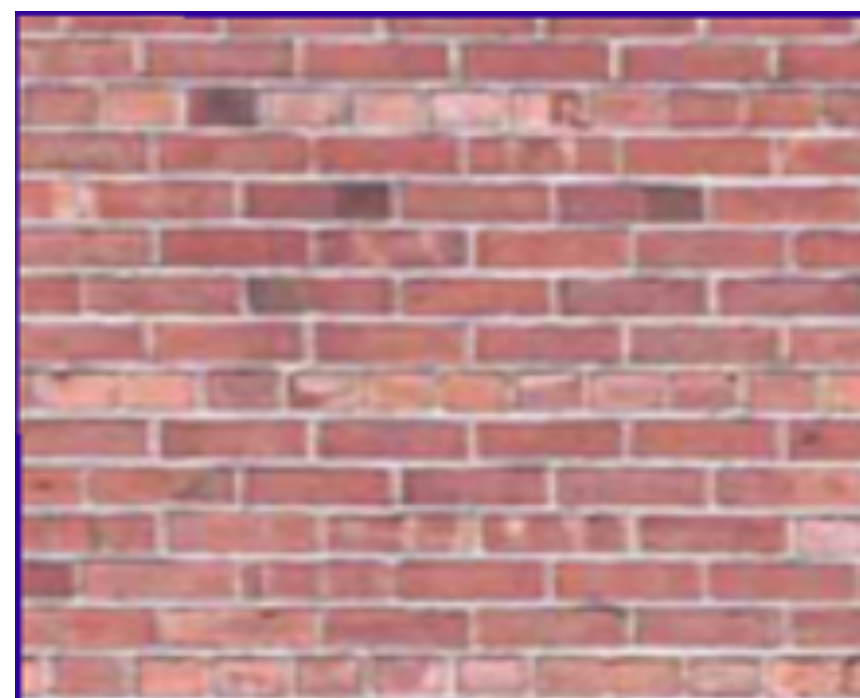
- Less distortion than sphere environment mapping
- Typically a smaller number of sample (texels) are sufficient
- More efficient
 - ➔ No inverse trigonometric functions
- Generate six views
 - ➔ Render (color) to texture
 - ➔ Can use `gl_layer` in geometry
 - ➔ Supported by hardware

```
glBindTexture(GL_TEXTURE_CUBE_MAP, cubemap);  
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X+i, 0  
             GL_RGB, W, H, 0, GL_RGB, GL_UNSIGNED_BYTE, 0);  
...  
glBindFramebuffer(GL_FRAMEBUFFER, fbo);  
glFramebufferTexture(GL_FRAMEBUFFER,  
                    GL_COLOR_ATTACHMENT0, cubemap, 0);  
glFramebufferRenderbuffer(GL_FRAMEBUFFER,  
                          GL_DEPTH_ATTACHMENT, GL_RENDERBUFFER, rbo);
```

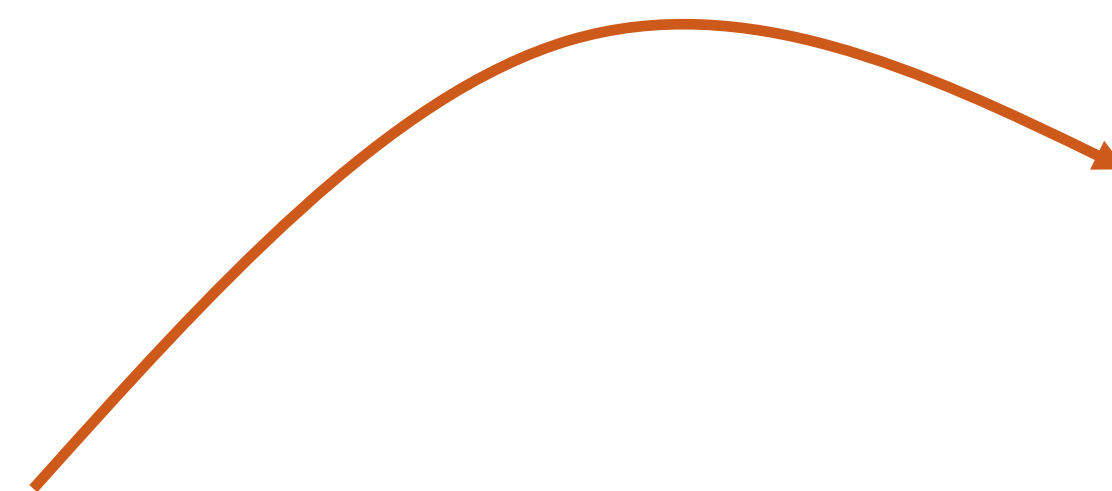
Light Texture



Light Texture



Diffuse Texture



Textured Surface

Light Maps

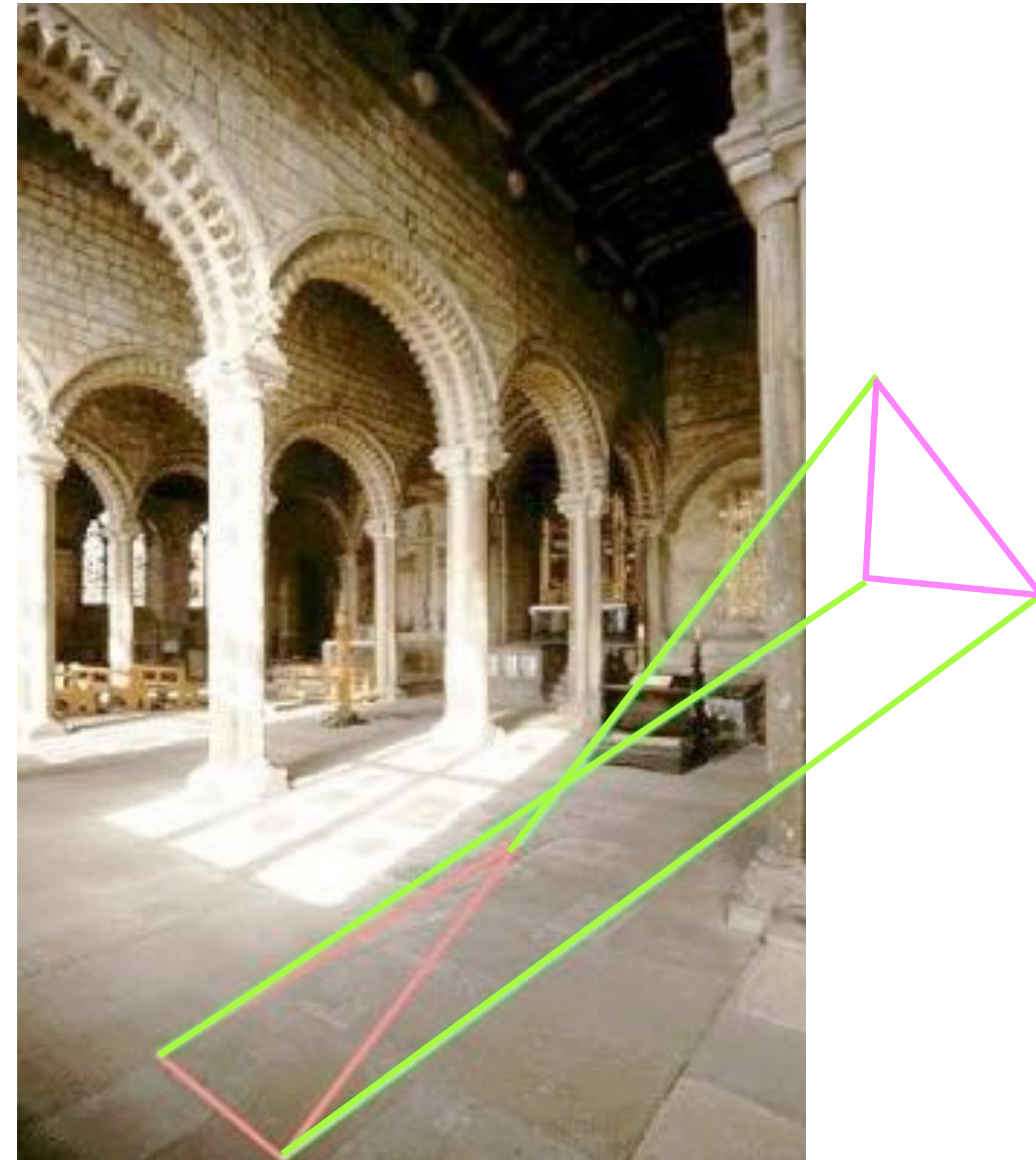


Quake

A Word on Texture Generation



- Texture creation is hard
 - ➡ Variations in reflectance and illumination
 - ➡ Tiling problems
 - ➡ Photographs have projective distortion



A Word on Texture Synthesis

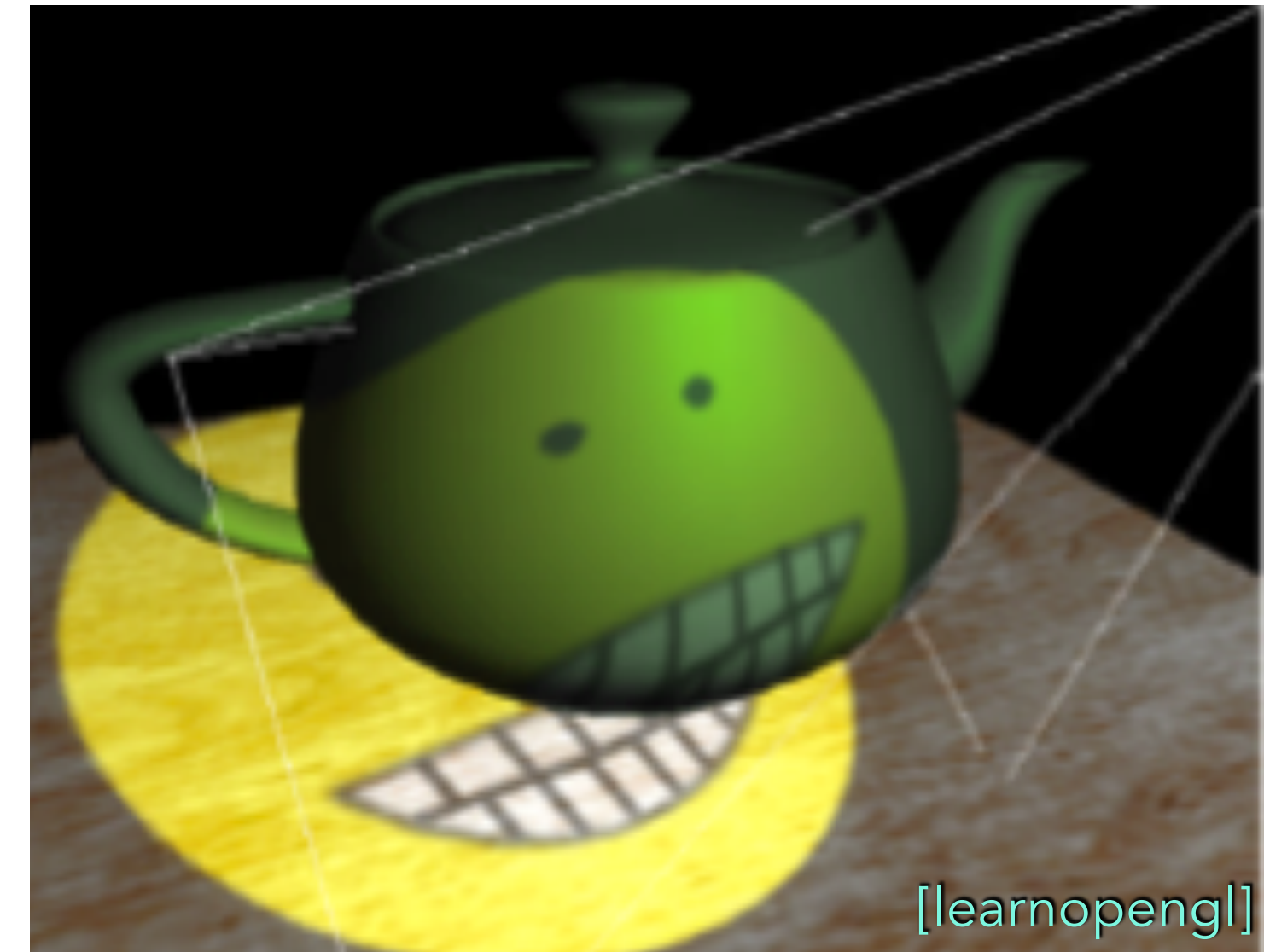


Tiling seldom works

A Word on Texture Coordinates



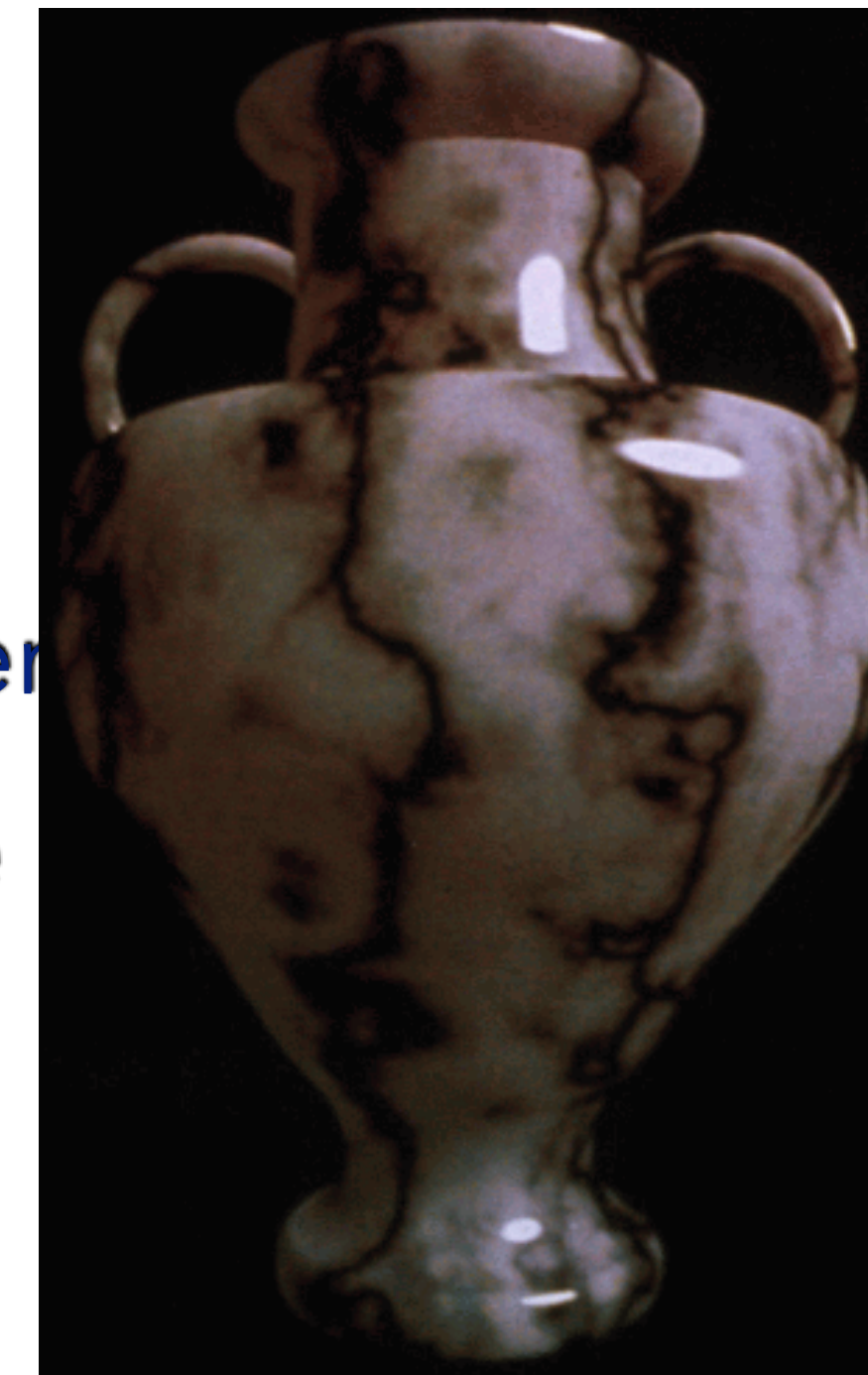
- Modeling-time assignment or on-the-fly
- A hard problem
 - ➔ Wrapping a paper over an arbitrary surface
 - ➔ Want to minimize distortion
- Projective textures: “Project” geometry into texture space
 - ➔ Orthographic, Perspective, Spherical/Cube, Cylindrical, etc.
 - ➔ Situate the texture plane as geometry in world space



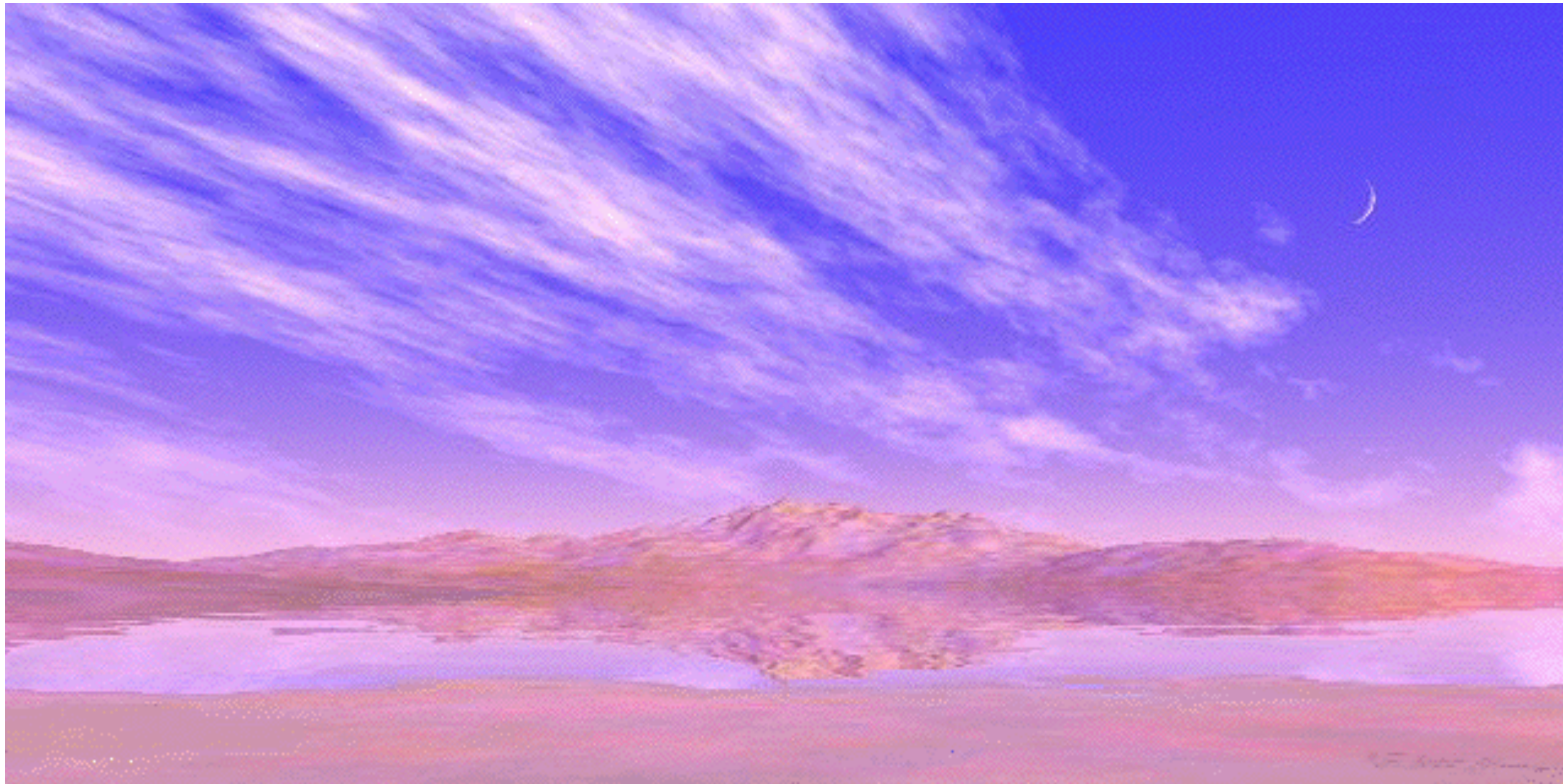


Solid Textures

- 3D Texture space
 - ➔ "3D" image
- Many procedural textures are 3D
 - ➔ Solid wood grains, Marble, Fire, Clouds, etc.
- May map to surface or volume
 - ➔ Volume textures may be approximated by multiple "translucency" layers
 - ➔ Surface solid textures appear carved out of a block of texture

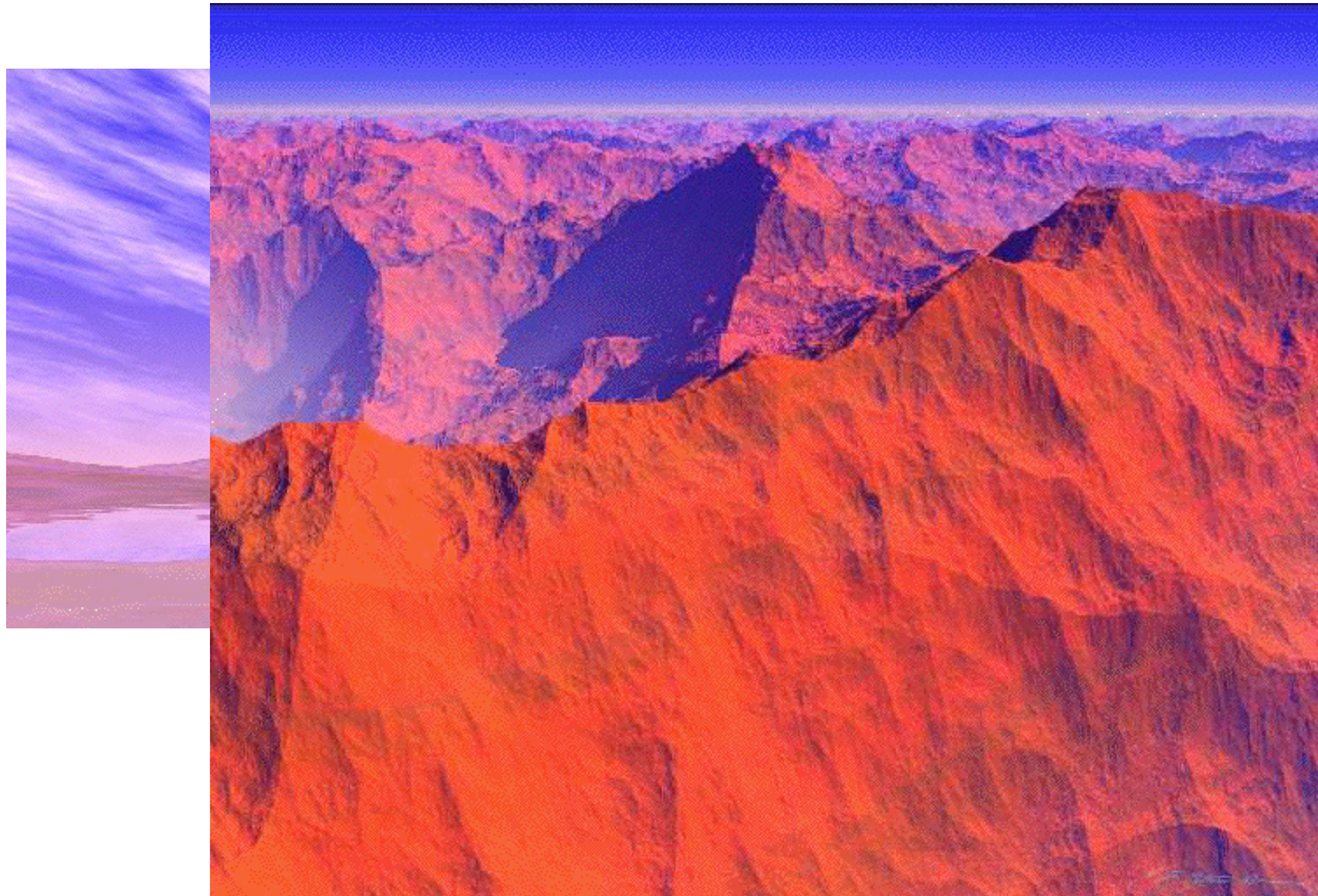


Procedural Textures



[courtesy Musgrave]

Procedural Textures

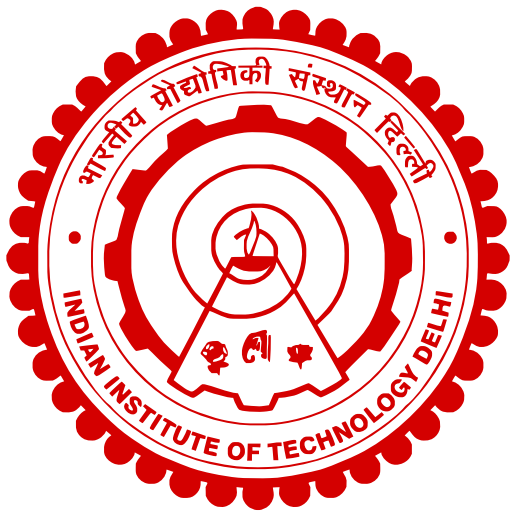


[courtesy Musgrave]

Procedural Textures

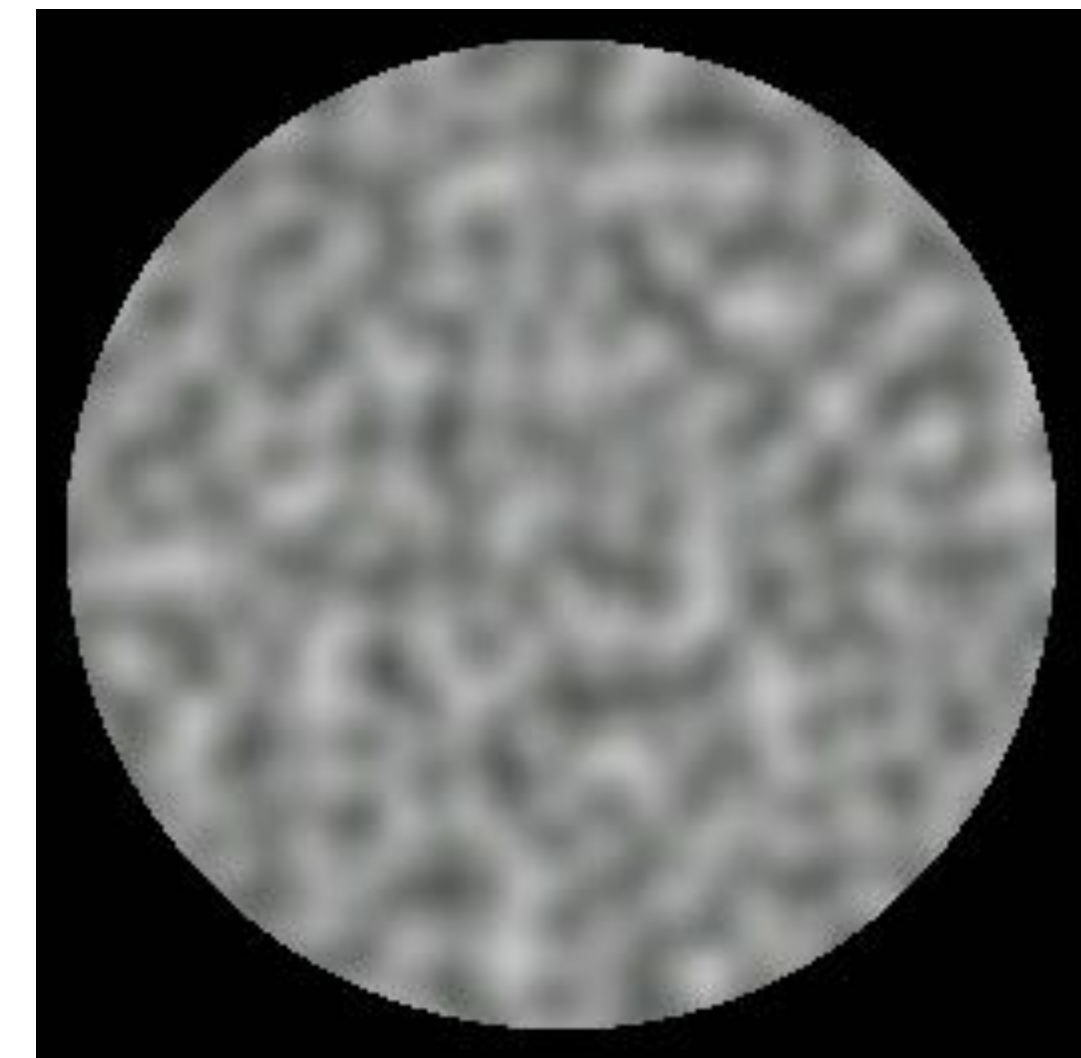


[courtesy Musgrave]



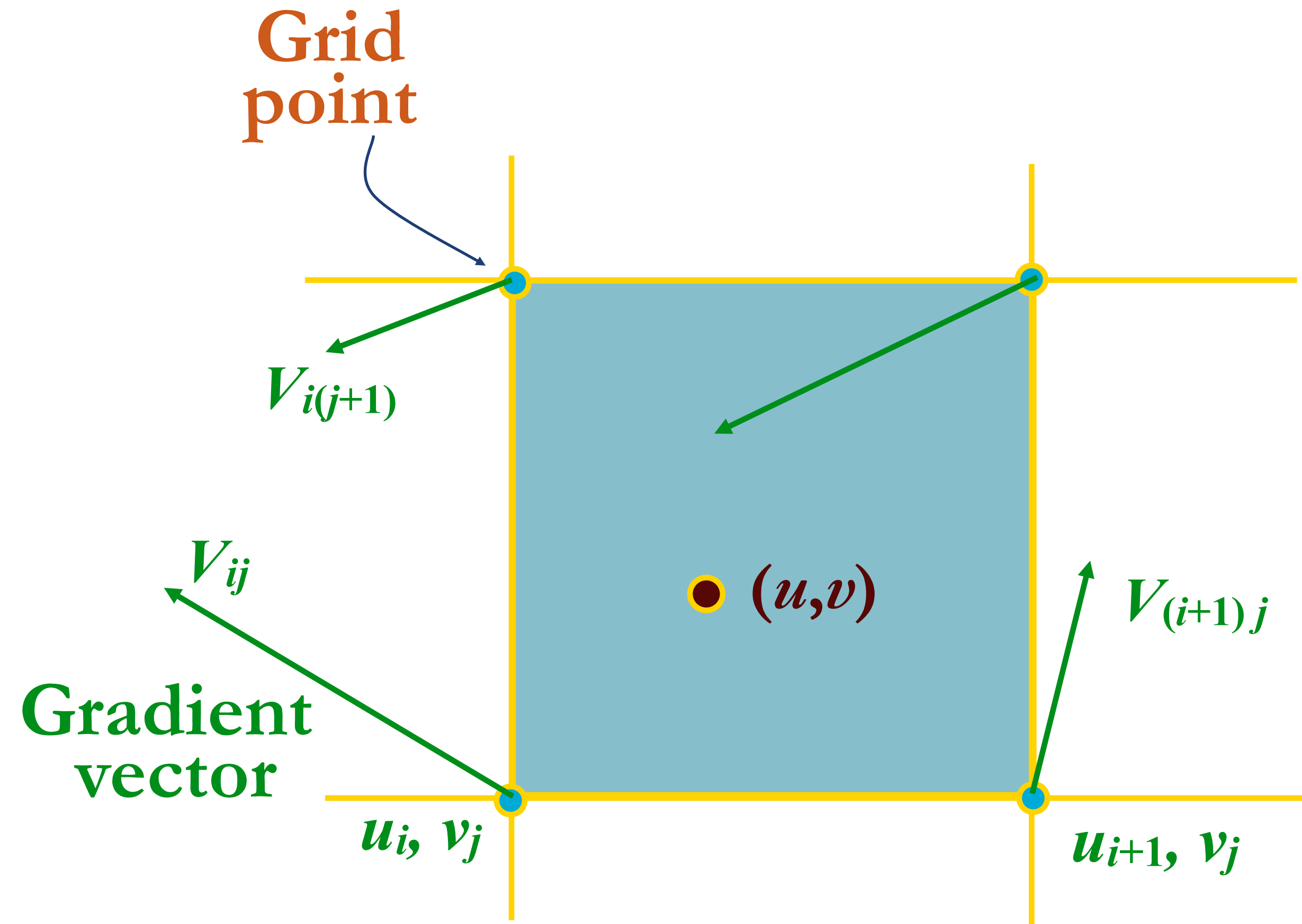
Perlin's Noise

- Visual features roughly the same size
- n -dimensional band-limited random function
 - ➔ Repeatable
- Approximation to gaussian filtered noise



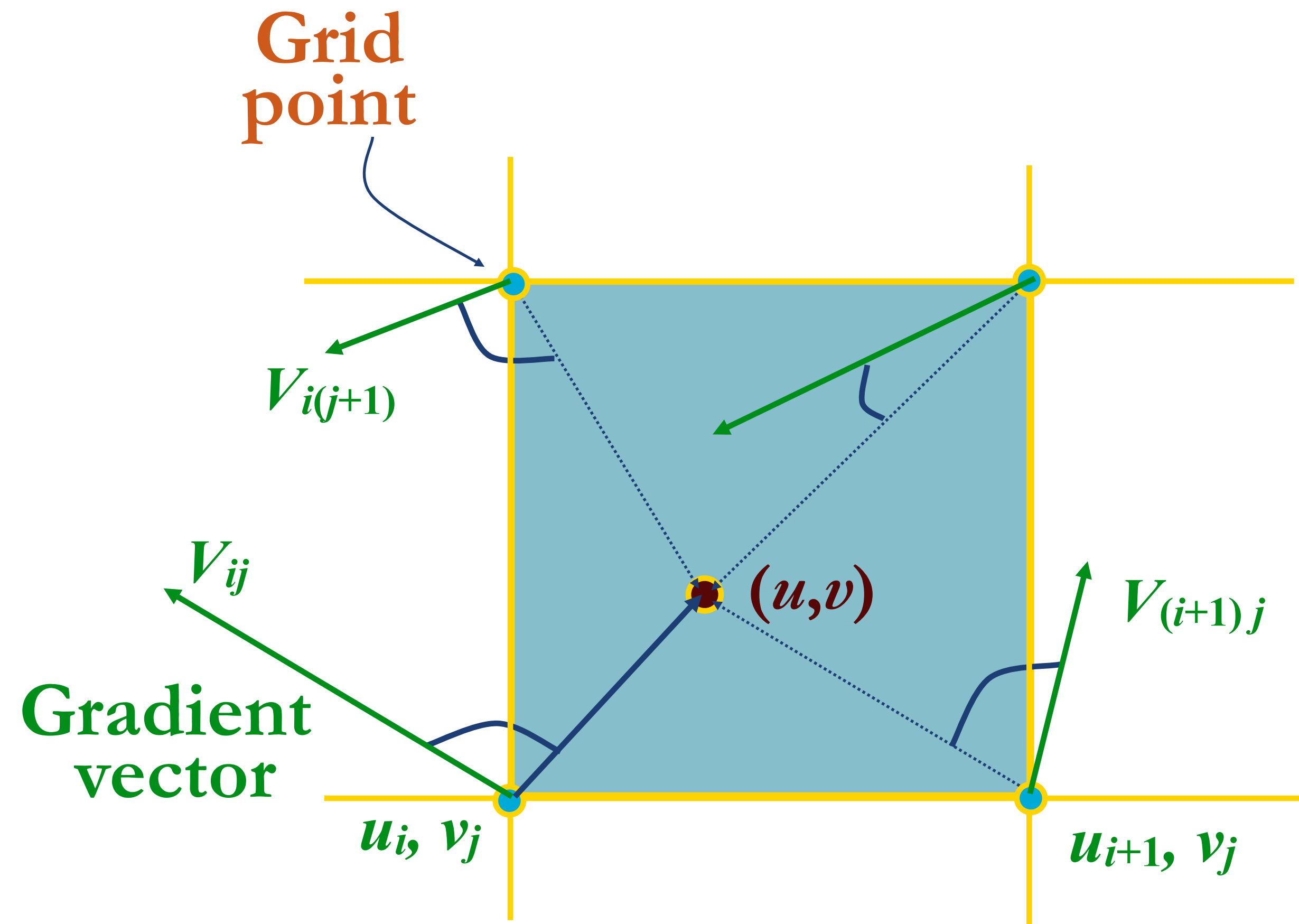


Smooth Noise



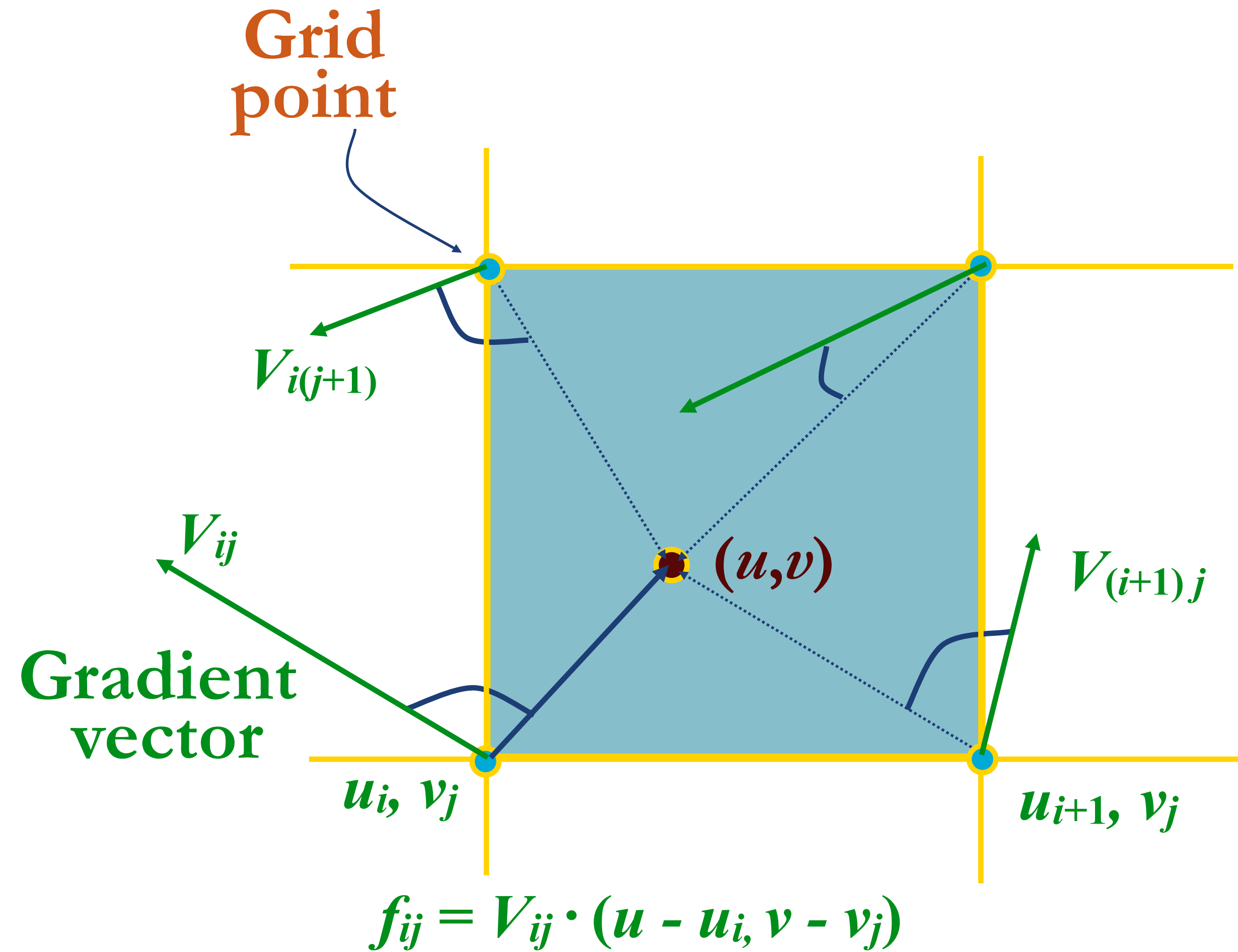


Smooth Noise





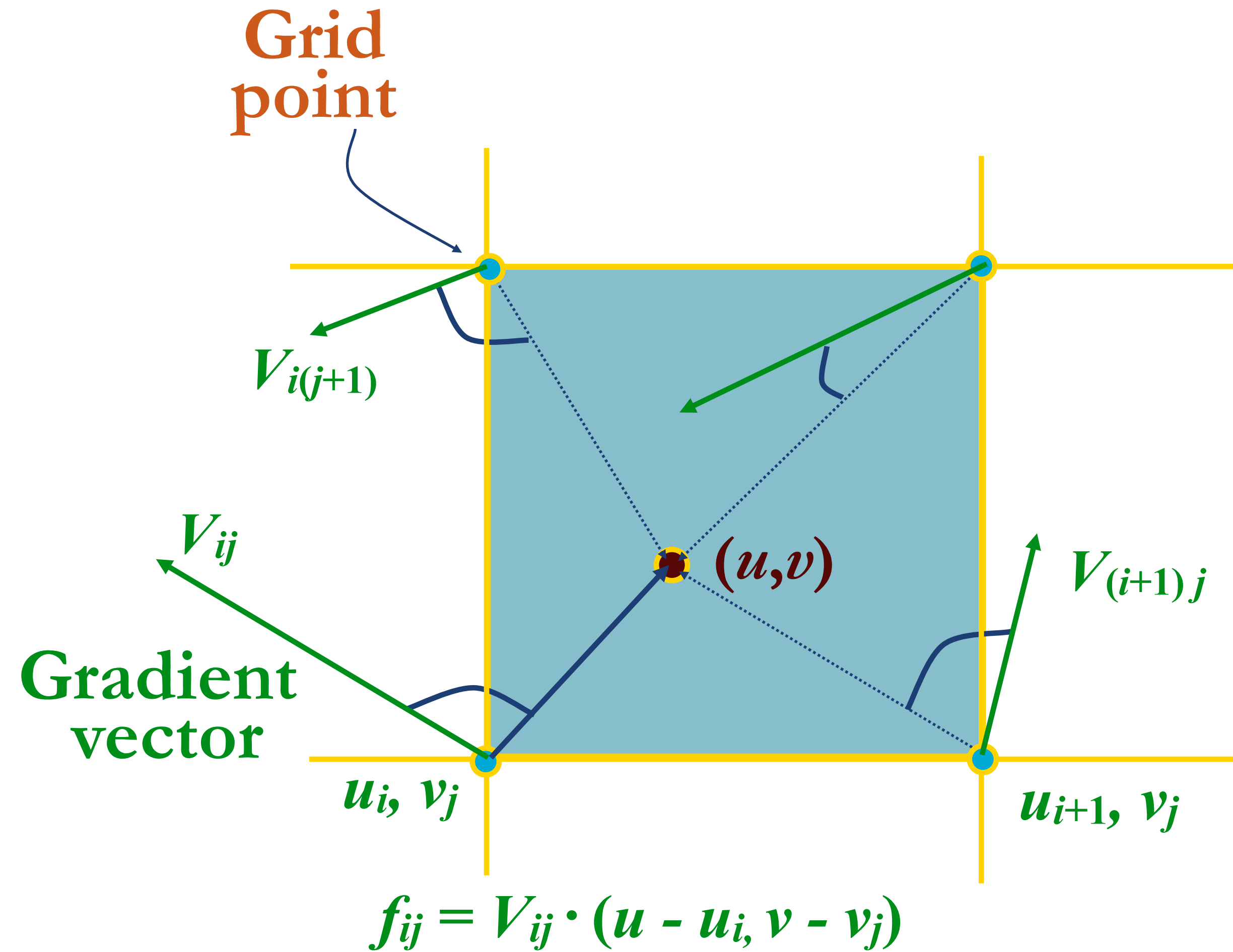
Smooth Noise



$$f(u, v) = \text{Interpolate} [f_{ij} + f_{(i+1)j} + f_{i(j+1)} + f_{(i+1)(j+1)}]$$

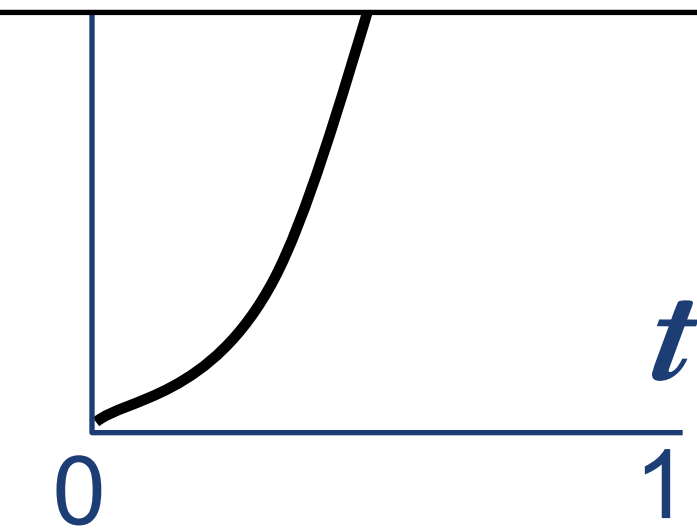


Smooth Noise



Ease Curve

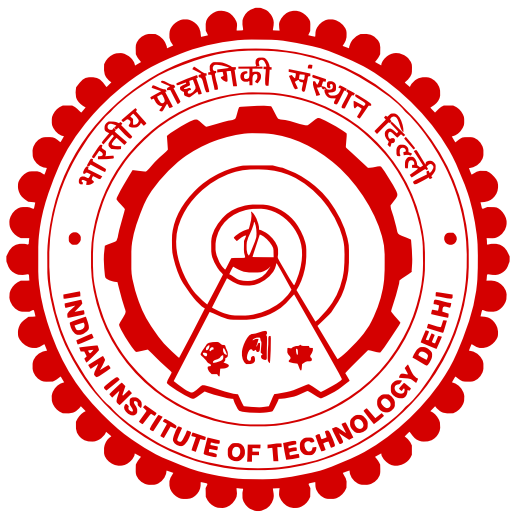
$$t^{(n+1)} = \sum_{k=0}^n \binom{n+k}{k} \binom{2n+1}{n-k} (-t)^k$$



e.g., $3t^2 - 2t^3$

$$t = (u - u_i) / (u_{i+1} - u_i)$$

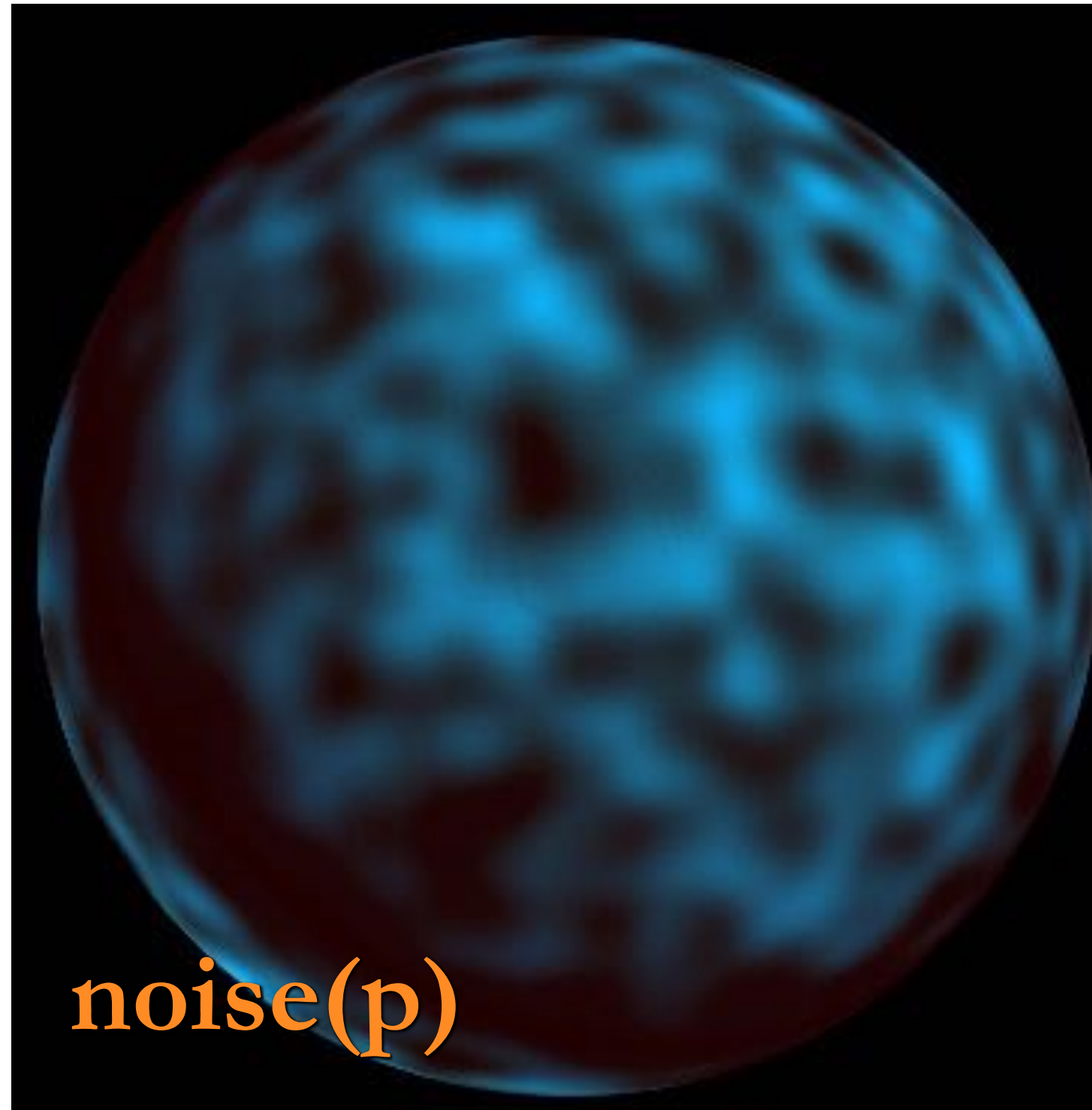
$$f(u, v) = \text{Interpolate} [f_{ij} + f_{(i+1)j} + f_{i(j+1)} + f_{(i+1)(j+1)}]$$



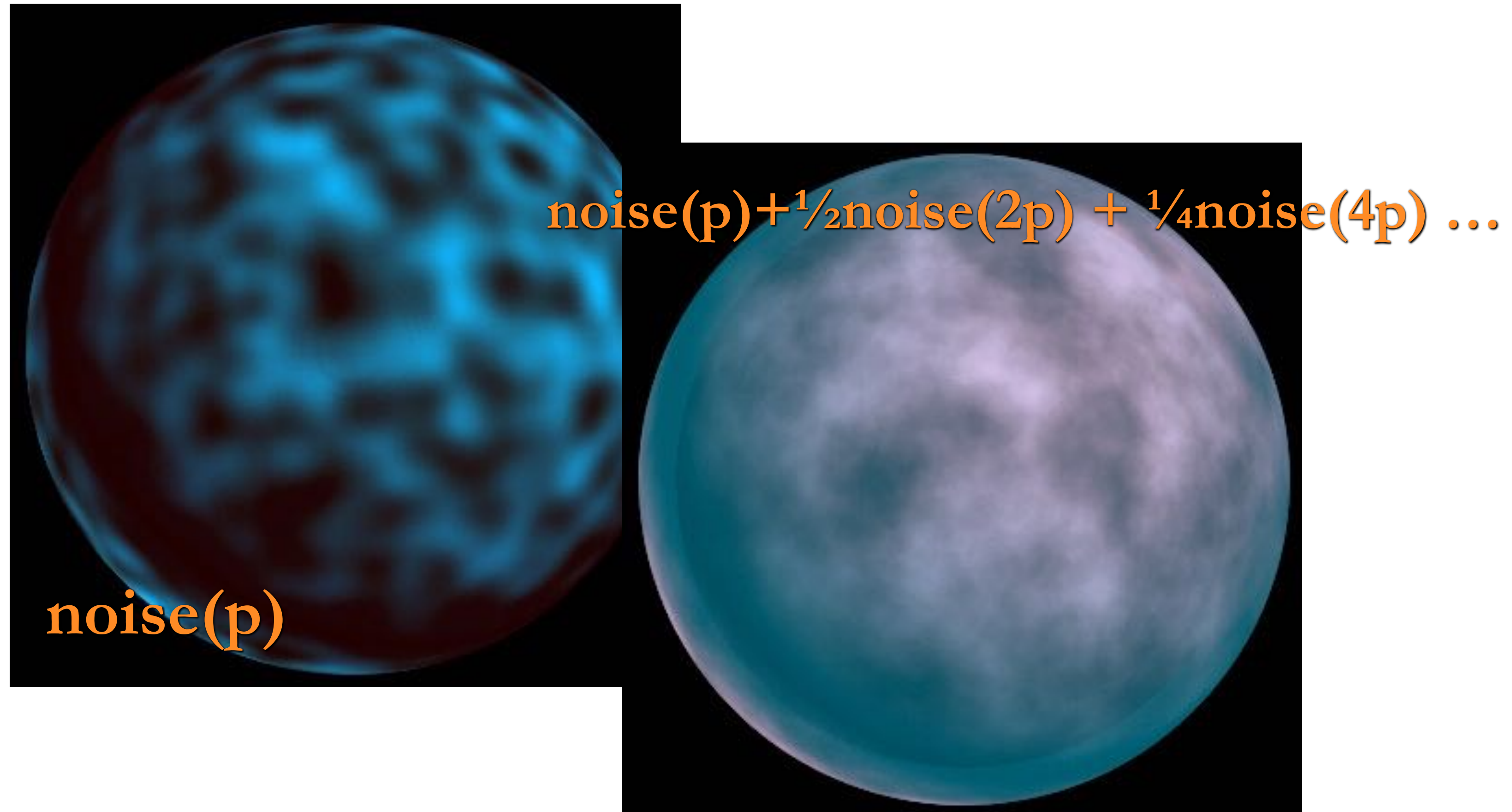
Algorithm

- Start with a grid of "pseudo-random" gradient vectors
 - ➡ Precompute table of permutations $P[n]$
 - ➡ Precompute table of gradients $G[n]$
 - $G = G[(i + P [(j + P[k]) \% n]) \% n]$
- For any point:
 - ➡ Compute vector from each neighboring gridpoint
 - ➡ Find dot products of the vectors and the gradients
 - ➡ Compute interpolation, using **ease** curves
 - $3t^2 - 2t^3$

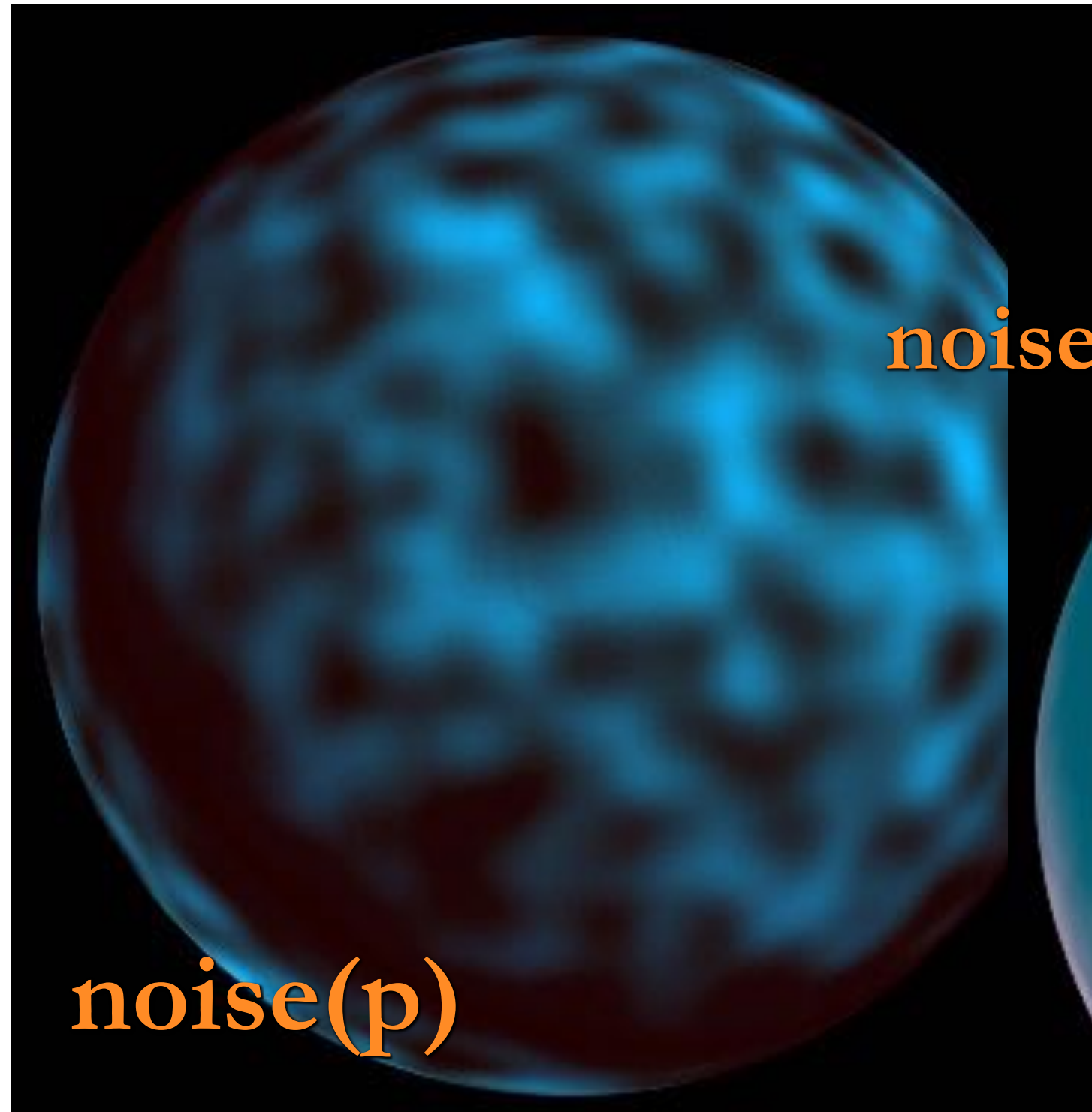
Examples



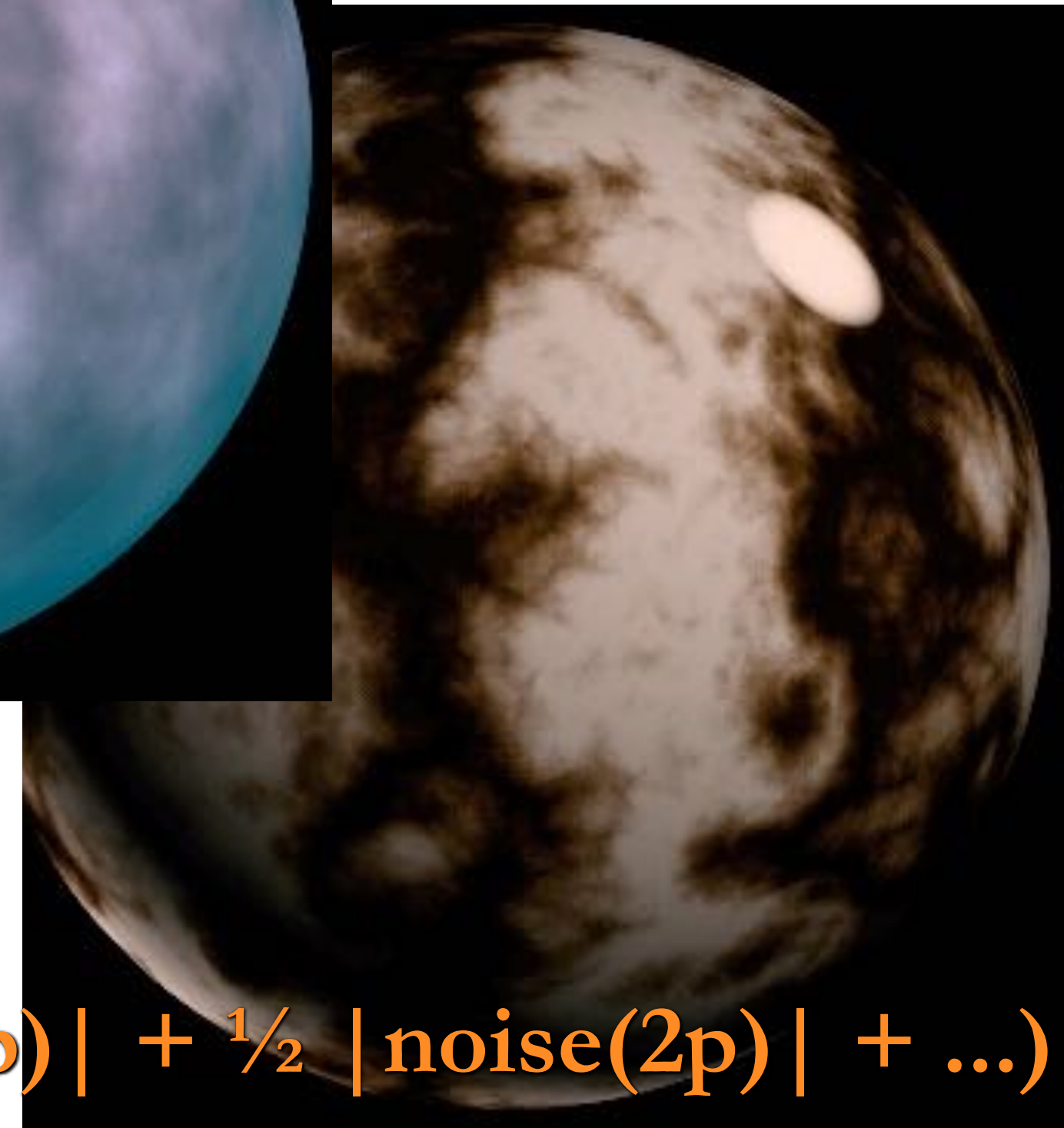
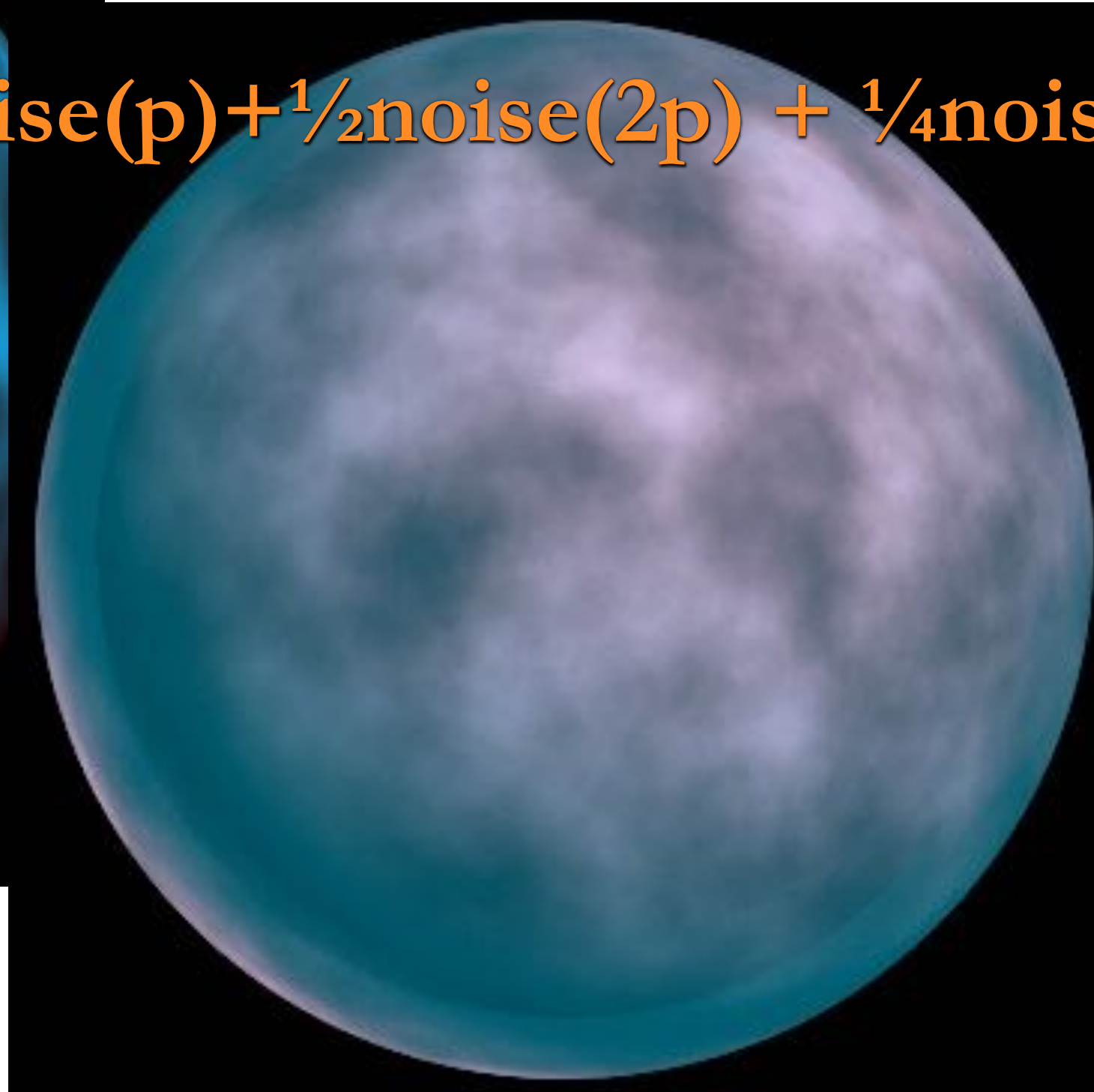
Examples



Examples



$$\text{noise}(p) + \frac{1}{2}\text{noise}(2p) + \frac{1}{4}\text{noise}(4p) \dots$$



$$\sin(x + |\text{noise}(p)| + \frac{1}{2}|\text{noise}(2p)| + \dots)$$