Texture Mapping

To add surface details…

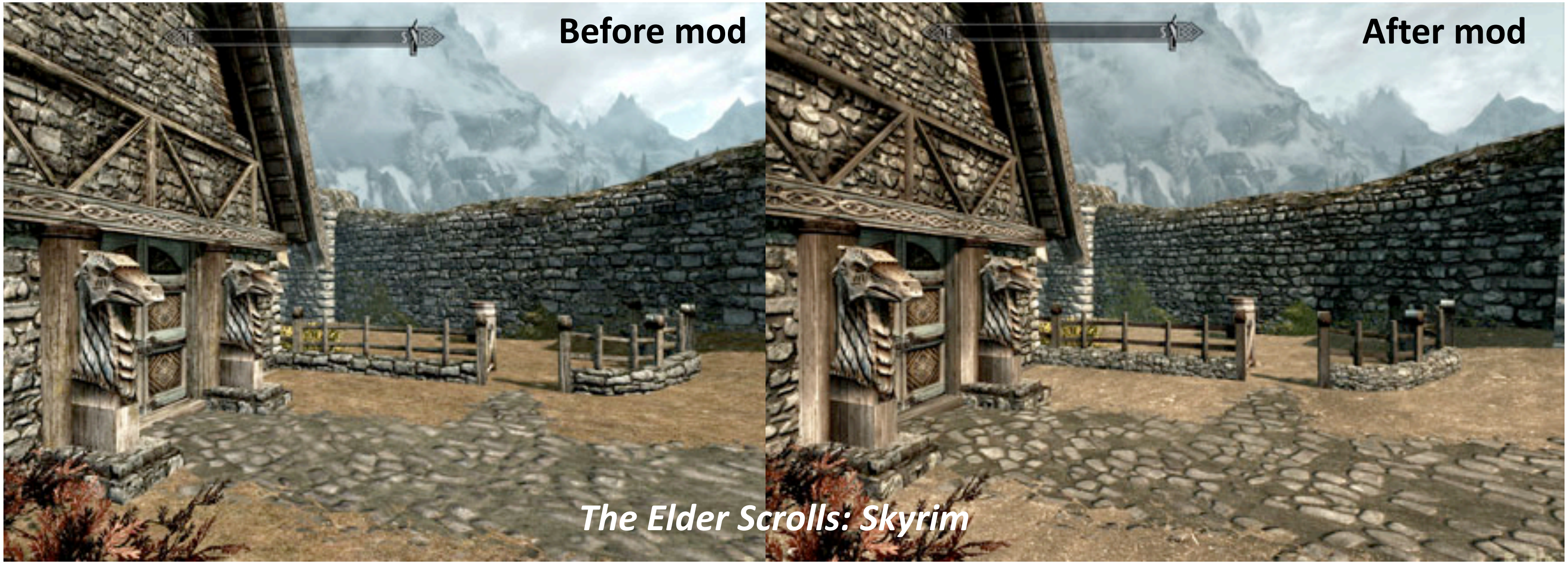
- More polygons (slow and hard to handle small details)
- Less polygons but with textures (much faster)

World of Warcraft, Blizzard Inc.

World Screenshots

Advantages

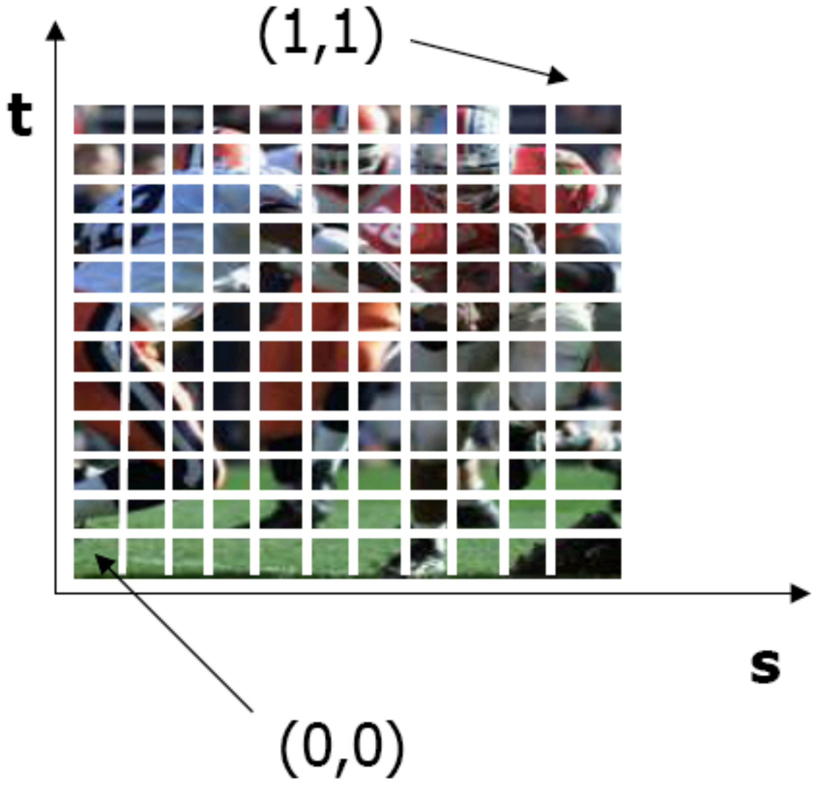
- Texture mapping doesn’t affect geometry processing, such as transformation, clipping, projection, …
- It does affect rasterization, which is highly accelerated by hardware.
- Textures can be easily replaced for more details: texture mod in games.



The Elder Scrolls: Skyrim

Texture Representation

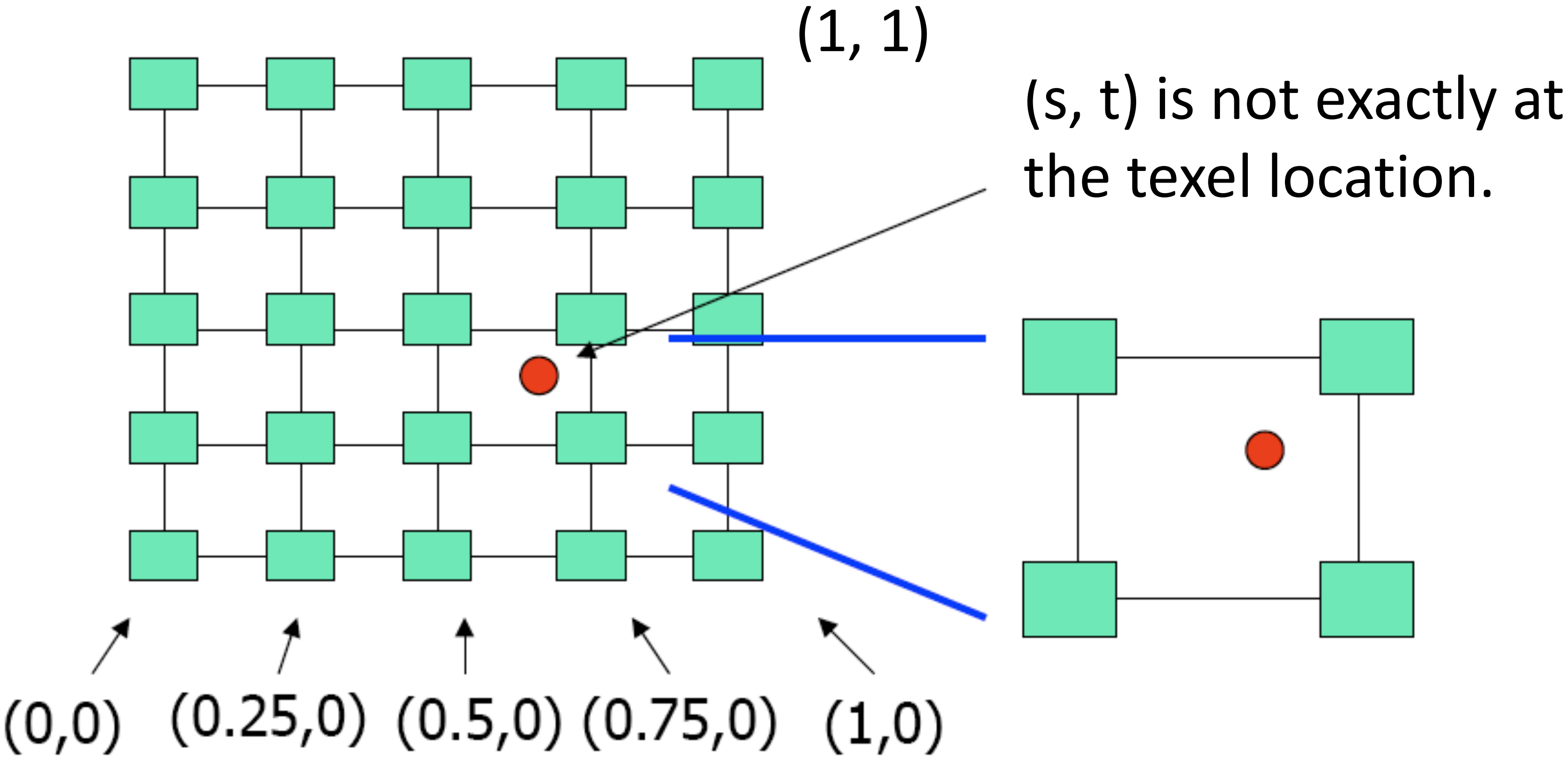
- **Bitmap textures** (supported by OpenGL)
- Procedural textures (used in advanced programs)



- **Bitmap texture**
  - A 2D image, represented by a 2D array (width-by-height).
  - Each pixel (or called texel) has a unique texture coordinate (s, t).
  - s and t are defined from 0 to 1.
  - Given (s, t), we can find a unique image value.

Texture Value Lookup

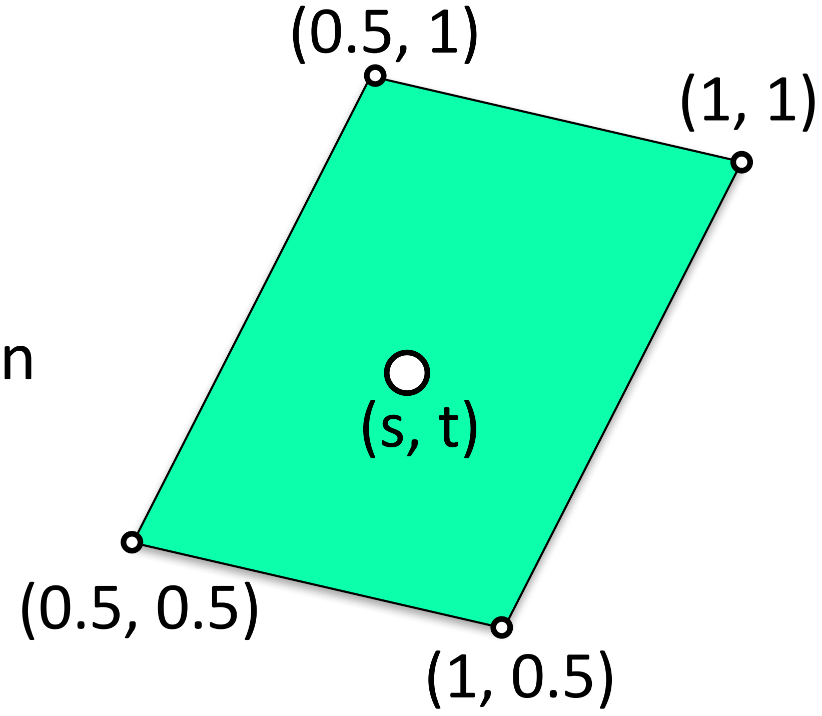
- To find the unique image value at (s, t):
  - Nearest neighbor
  - Bi-linear interpolation
  - Other filters



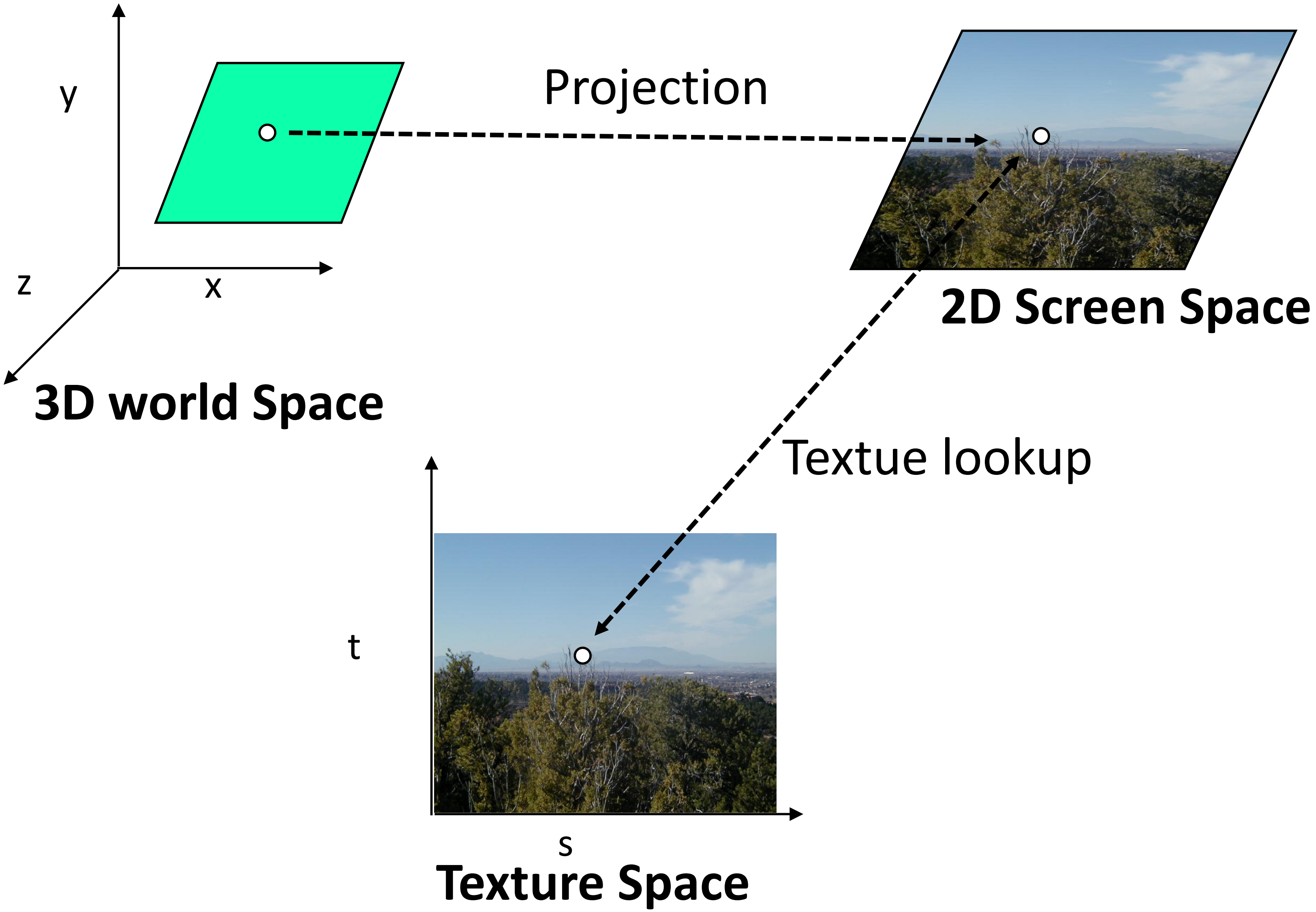
Mapping from Texture to Polygon

- Texture mapping is performed in rasterization.

- Given texture coordinates at vertices,
  - Calculate the texture coordinate (s, t) at each pixel, using linear interpolation
  - Find the texture value using texture lookup
  - Combine it with the illumination effect…



Texture Mapping



OpenGL Texture Mapping

- Steps in OpenGL
  - Specify the texture: read/generate the image, assign it as a texture
  - Specify texture mapping parameters: wrapping, filtering, …
  - Enable OpenGL texture mapping (GL_TEXTURE_2D)
  - Assign texture coordinates to vertices
  - Draw your objects
  - Disable OpenGL texture mapping

Specify Textures

- Load the texture from main memory into texture memory

```
glTexImage2D(target, level, iformat, width, height,
           border, format, type, pointer);
```

- For example,

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, 64, 64, 0,
         GL_RGB, GL_UNSIGNED_BYTE, pointer);
```

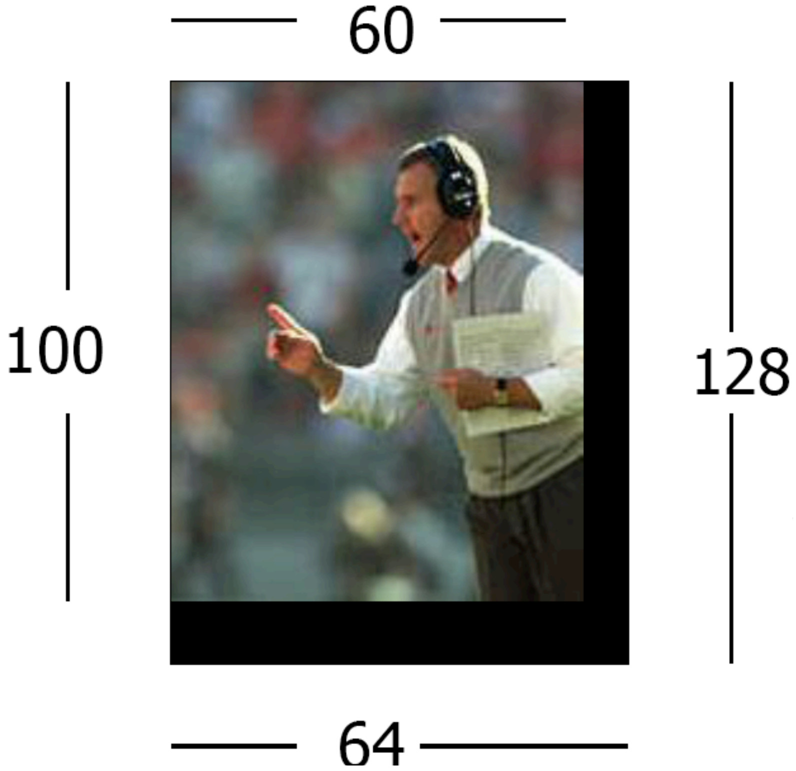
```
GLuByte pointer[64][64][3];
```

- The texture resolution must be power of 2.

Fix Texture Size

- If the resolution is not power of 2:
  - Pad zeros by yourself

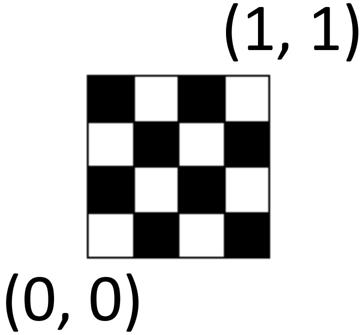
    (remember to adjust texture coordinates for your vertices, or you will see the dark boundary).
  - Or, resize the image using:



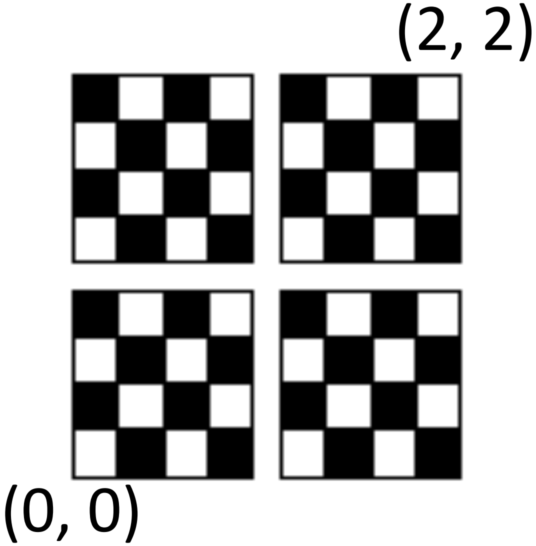
```
GLint gluScaleImage(GLenum format,
                    GLsizei wIn, GLsizei hIn,
                    GLenum typeIn, const void *dataIn,
                    GLsizei wOut, GLsizei hOut,
                    GLenum typeOut, const void *dataOut)
```

Texture Mapping Parameters (1)

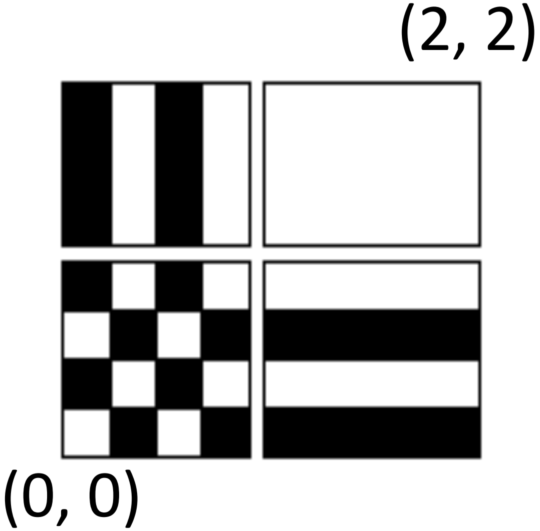
- (s, t) in the texture space are from 0 to 1. But what if vertices have texture coordinates beyond this range?



Texture



GL_REPEAT



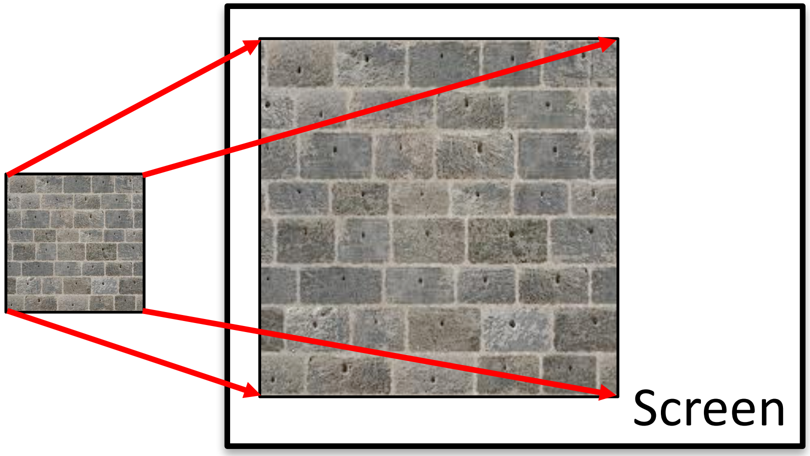
GL_CLAMP

- For example,

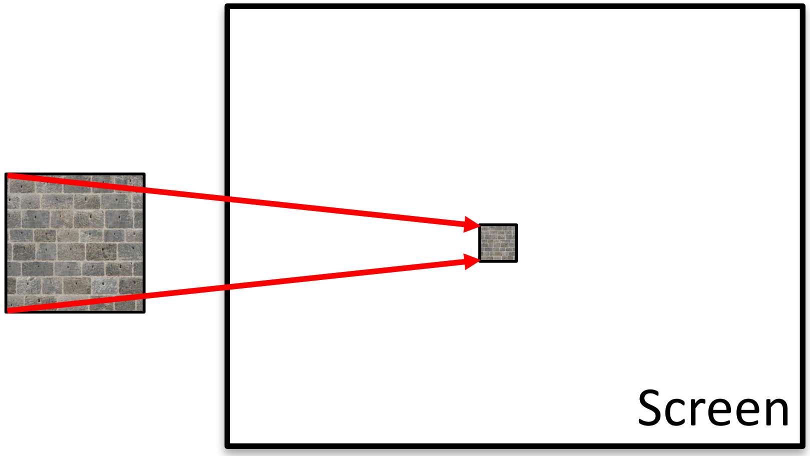
```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
GL_CLAMP)
```

Texture Mapping Parameters (2)

- Since a texture can be mapped arbitrarily to an image region, it can either be magnified or minified.

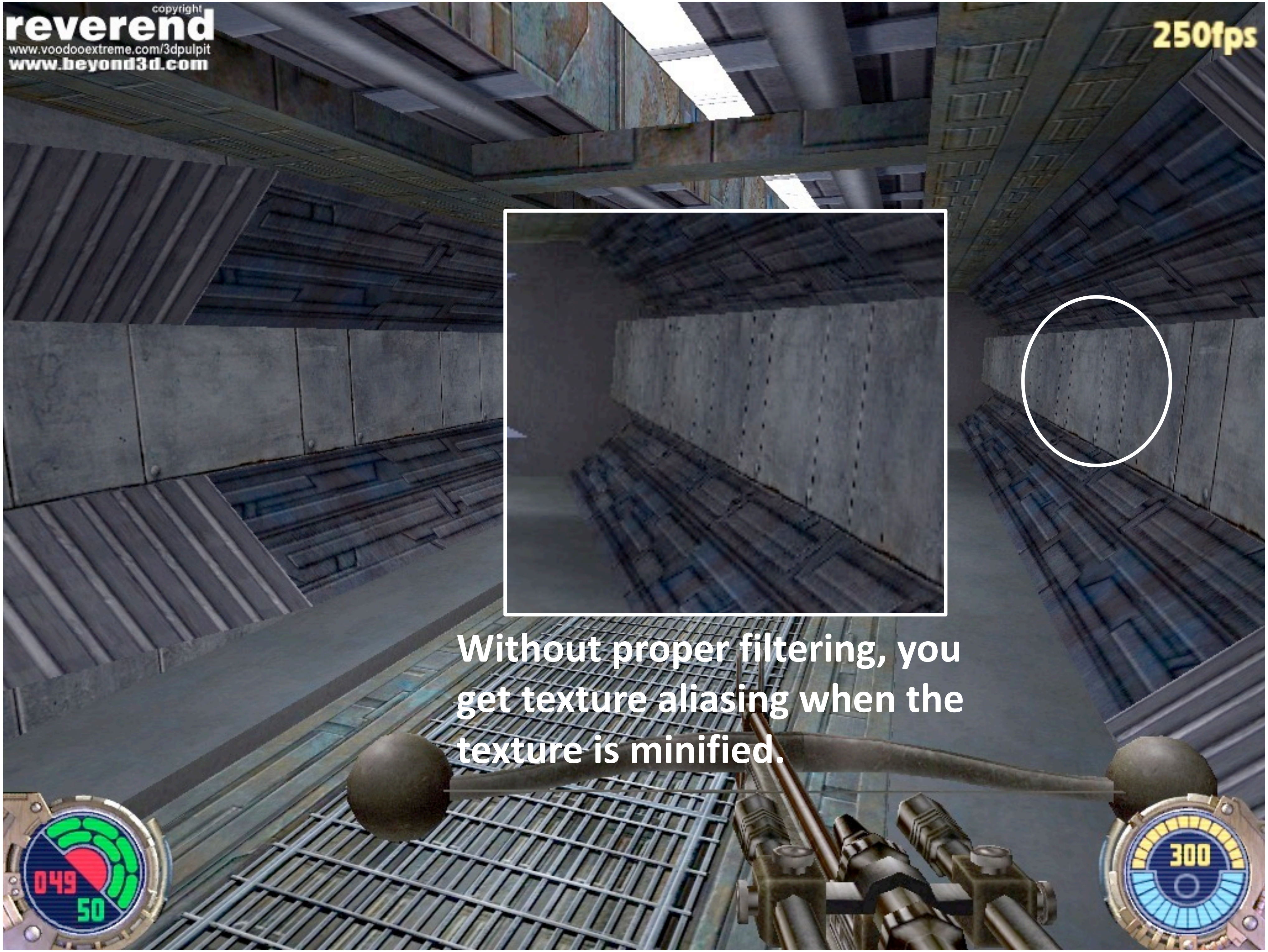


Magnification



Minification

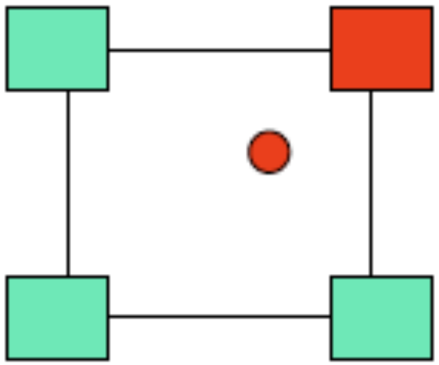
- Mag filter: To interpolate a value from neighboring texels
- Min filter: Combine multiple texels into a single value



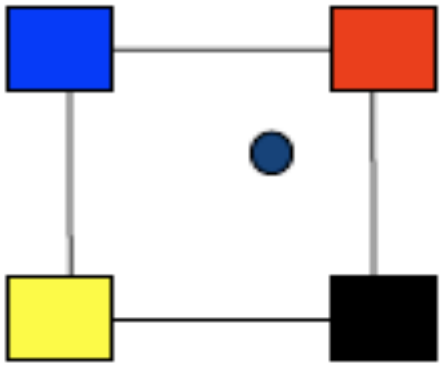
Without proper filtering, you get texture aliasing when the texture is minified.

Texture Mapping Parameters (3)

- OpenGL texture filtering:



Nearest Neighbor
(fast, but with aliasing)



Bi-linear Interpolation
(slow, but less aliasing)

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST)
```

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)
```

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)
```

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
```

Texture Color Blending

- Determine how to combine the texture color with the object color
  - GL_MODULATE: multiply texture with object color
  - GL_BLEND: linear combination of texture and object color
  - GL_REPLACE: use texture color to replace object color
- For example,

```
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE)
```

- Remember to use GL_MODULATE (default) if you want to have the light effect.

Enable/Disable Textures

```
glEnable(GL_TEXTURE_2D)
```

```
glDisable(GL_TEXTURE_2D)
```

Remember to disable texture mapping when drawing non-texture polygons.

Specify Texture Coordinates

- Define texture coordinates before each vertex

```
glBegin(GL_QUADS);
glTexCoord2D(0, 0);
glVertex3f(-1, 0, -1);
      . . .
glEnd();
```

- Texture coordinates can be transformed as well, using a GL_TEXTURE matrix.
    - Switch to:

      ```
      glMatrixMode(GL_TEXTURE);
      ```

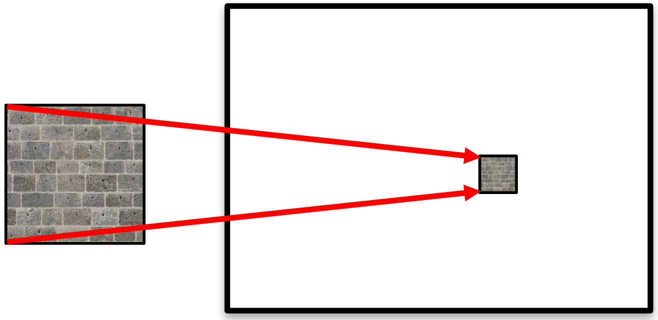
    - Apply 2D transformation
    - Then draw your object
    - Not necessary to use

Summary

```
…
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
…

glEnable(GL_TEXTURE_2D);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, 64, 64, 0, GL_RGB,
GL_UNSIGNED_BYTE, img_pointer);

glBegin(GL_TRIANGLES);
glTexCoord2D(0, 0);
glNormal3f(0, 1, 0);
glVertex3f(-1, 0, -1);
…
glEnd();
glDisable(GL_TEXTURE_2D);


…
```

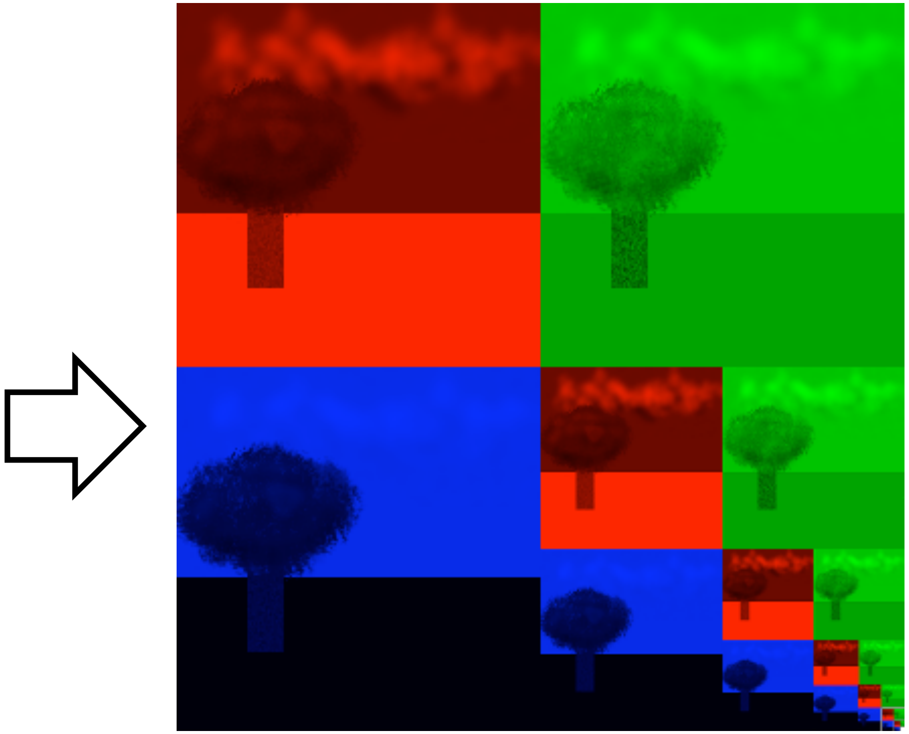
Mip Map

- Mip map is a technique that helps improve the computational efficiency and avoid aliasing:



Minification

Original Image



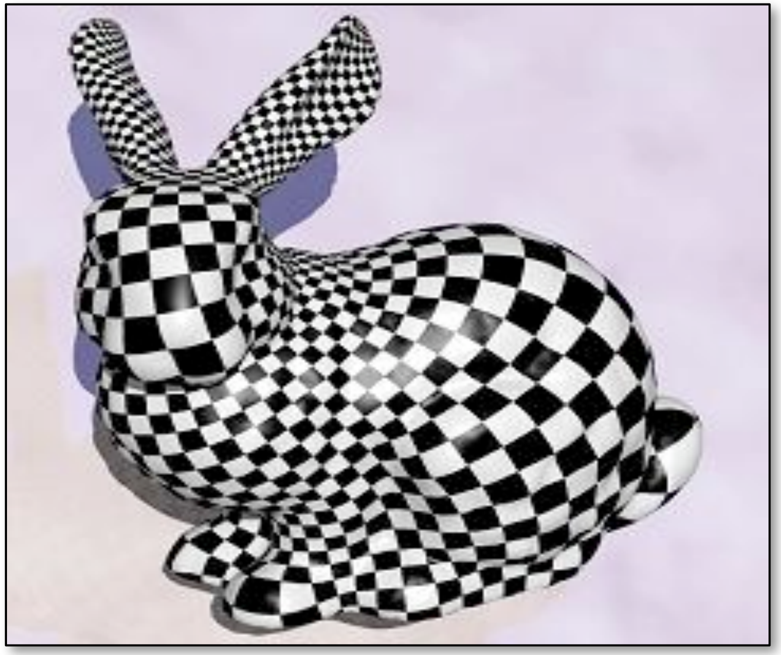
Mip Map

Multiple MipMap Textures in OpenGL

```
…
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_NEAREST_MIPMAP_NEAREST);
…

Gluint texture1, texture2;
glGenTextures(1, &texture1);
glBindTexture(GL_TEXTURE_2D, texture1);
gluBuild2DMipmaps(GL_TEXTURE_2D, 3, width, height, GL_RGB,
GL_UNSIGNED_BYTE, data1);
                                 Number of input channels

glGenTextures(1, &texture2);
glBindTexture(GL_TEXTURE_2D, texture2);
gluBuild2DMipmaps(GL_TEXTURE_2D, 4, width, height, GL_RGBA,
GL_UNSIGNED_BYTE, data2);
                                 Number of input channels

//To use them
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texture1);
Draw_Object_1();
glBindTexture(GL_TEXTURE_2D, texture2);
Draw_Object_2();
```

Surface Parameterization

- Find texture coordinates for a planar surface is trivial.
- However, finding texture coordinates for an arbitrarily curved surface is a research problem called: **surface parameterization**.
- It means parametrizing the surface using texture coordinates (s, t).

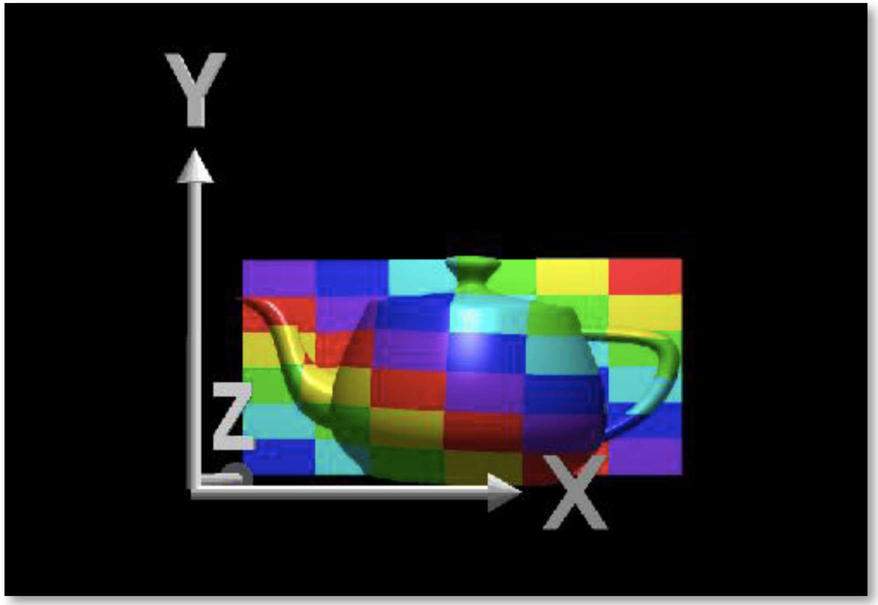


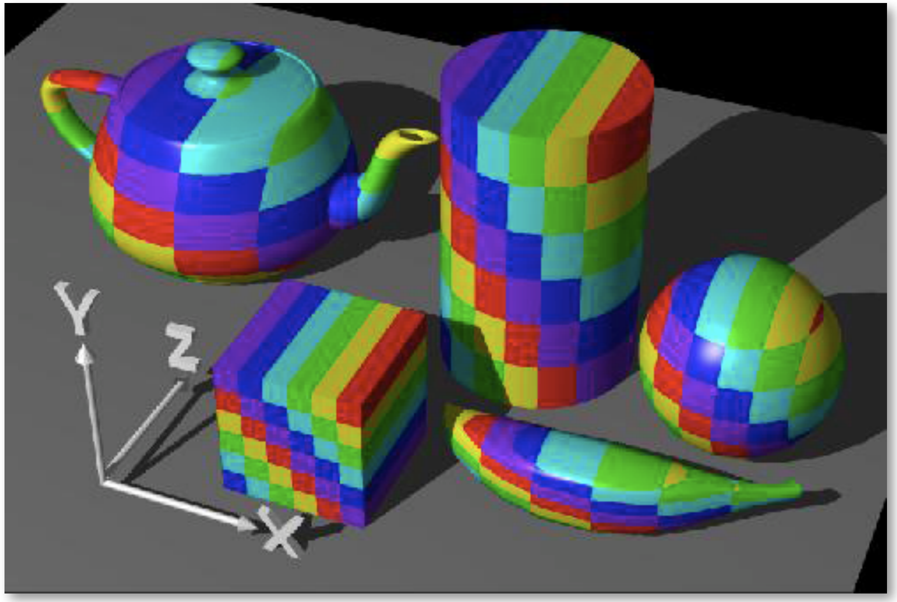
An Example

2D Texture

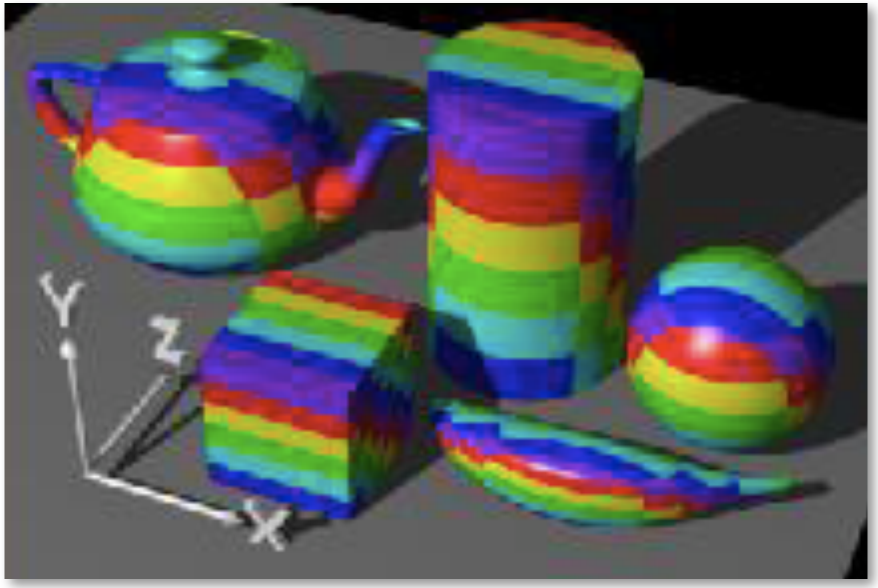
3D Earth

Planar Projection

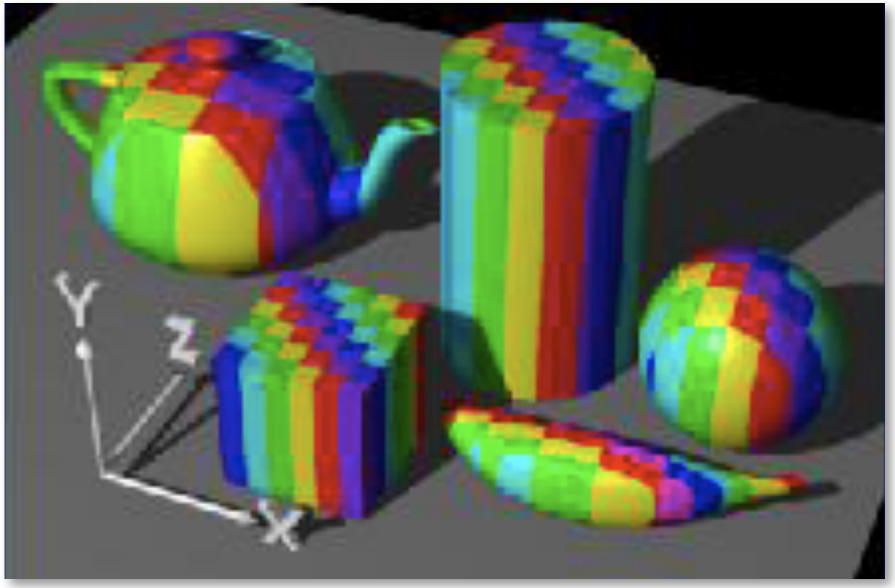




Vertex (x, y, z) -> Texture (x, y)

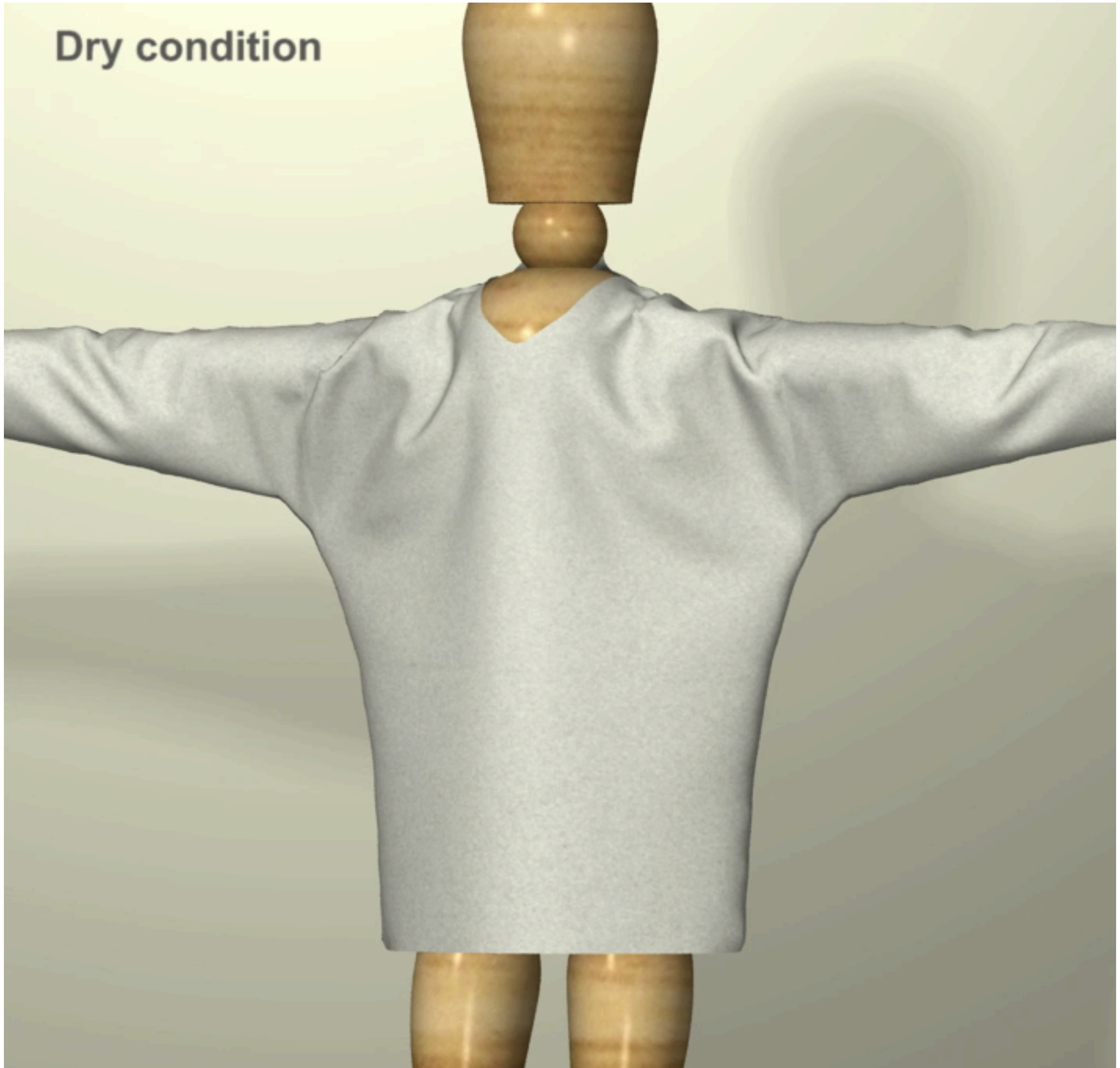


Vertex (x, y, z) -> Texture (y, z)



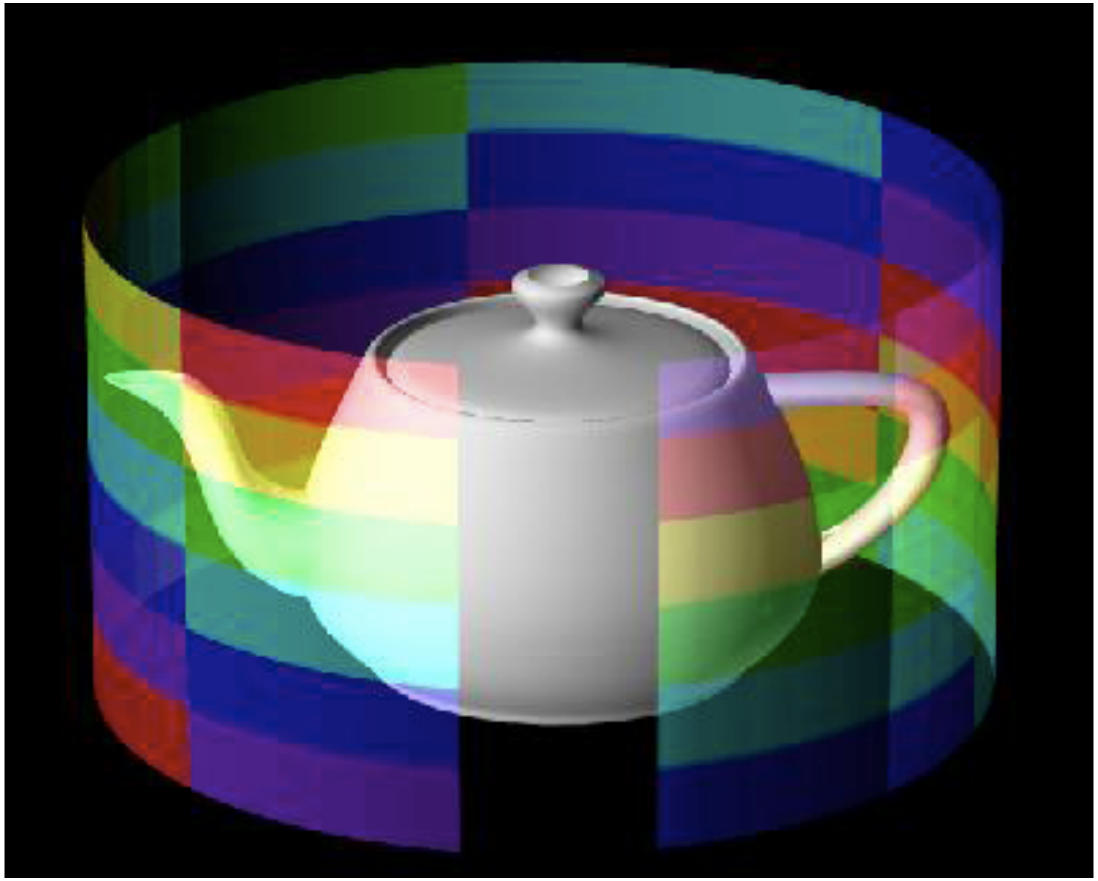
Vertex (x, y, z) -> Texture (x, z)

Planar Projection



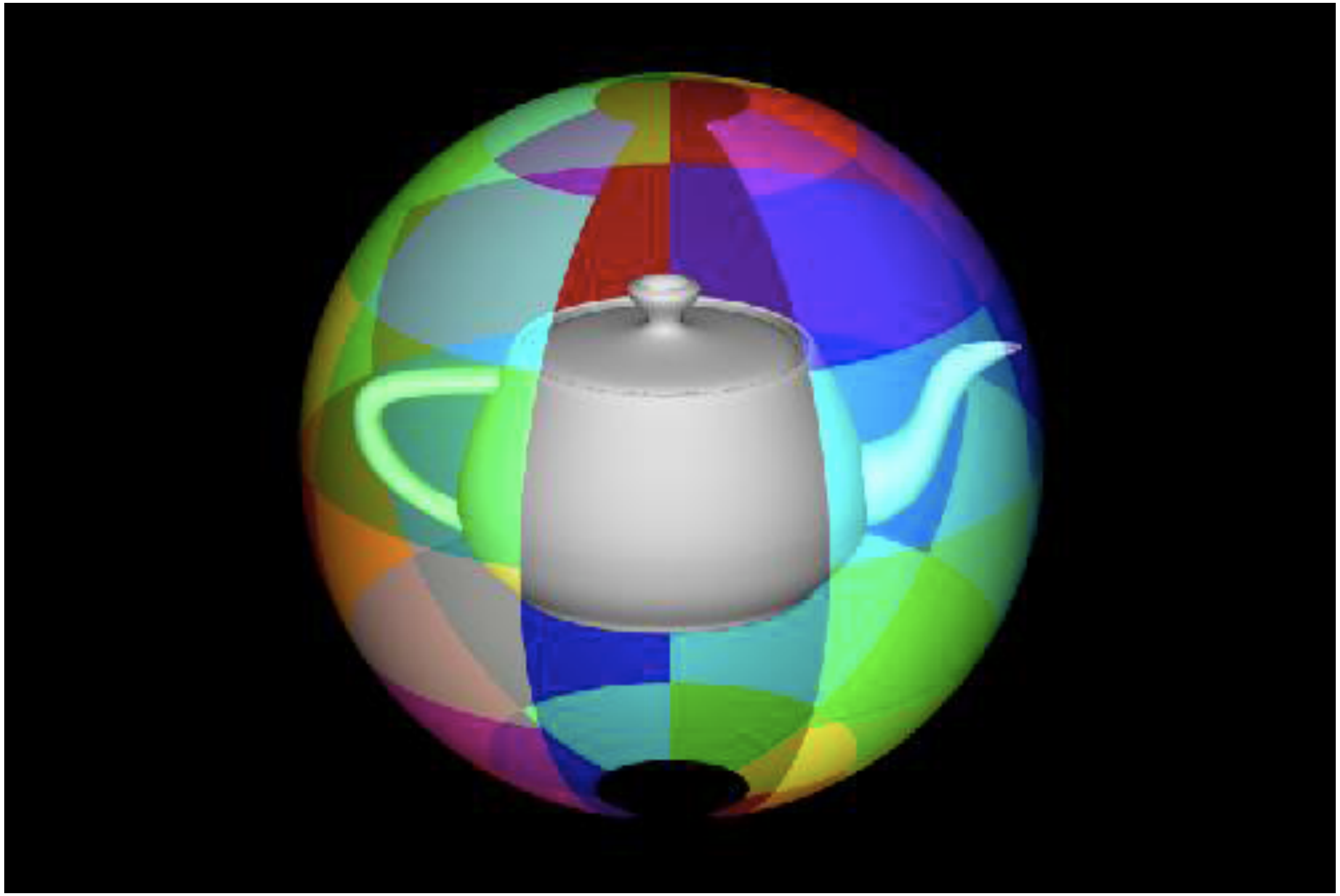
Cylindrical Projection

- Project any 3D point onto a cylinder. The height and the angle become texture coordinates: (s, t).



Spherical Projection

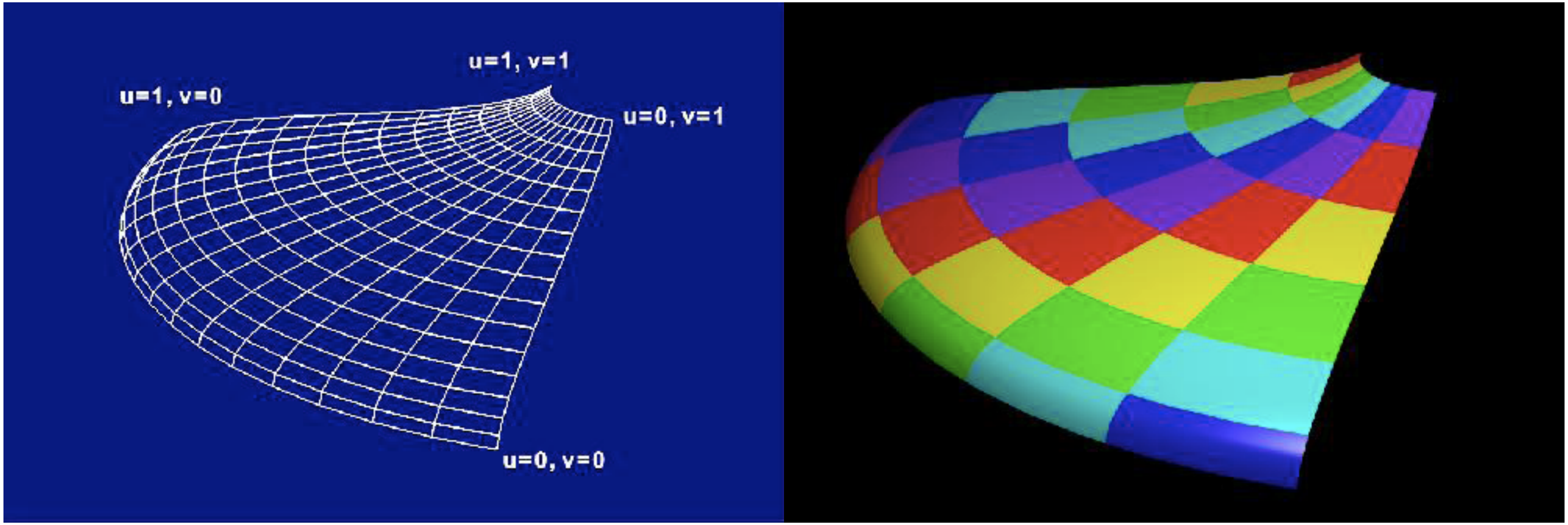
- Project any 3D point onto a unit sphere. The spherical coordinates are texture coordinates: (s, t).



Parametric Surfaces

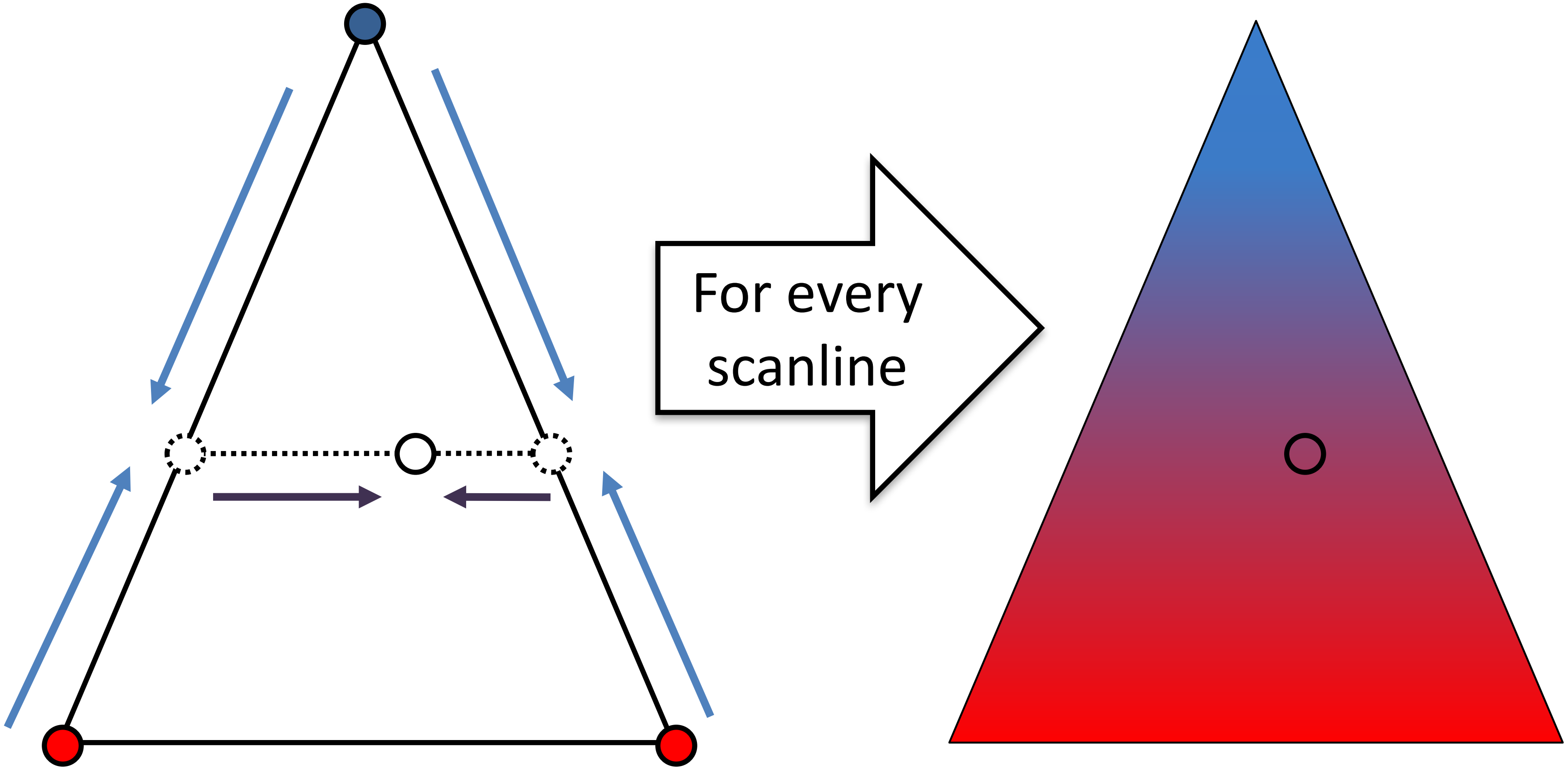
- Surfaces can also be created in a parametric way. Any 3D point on the surface are defined as functions of texture coordinates:

$$x = \mathrm{f}(s,t), \qquad y = \mathrm{g}(s,t), \qquad z = \mathrm{h}(s,t)$$



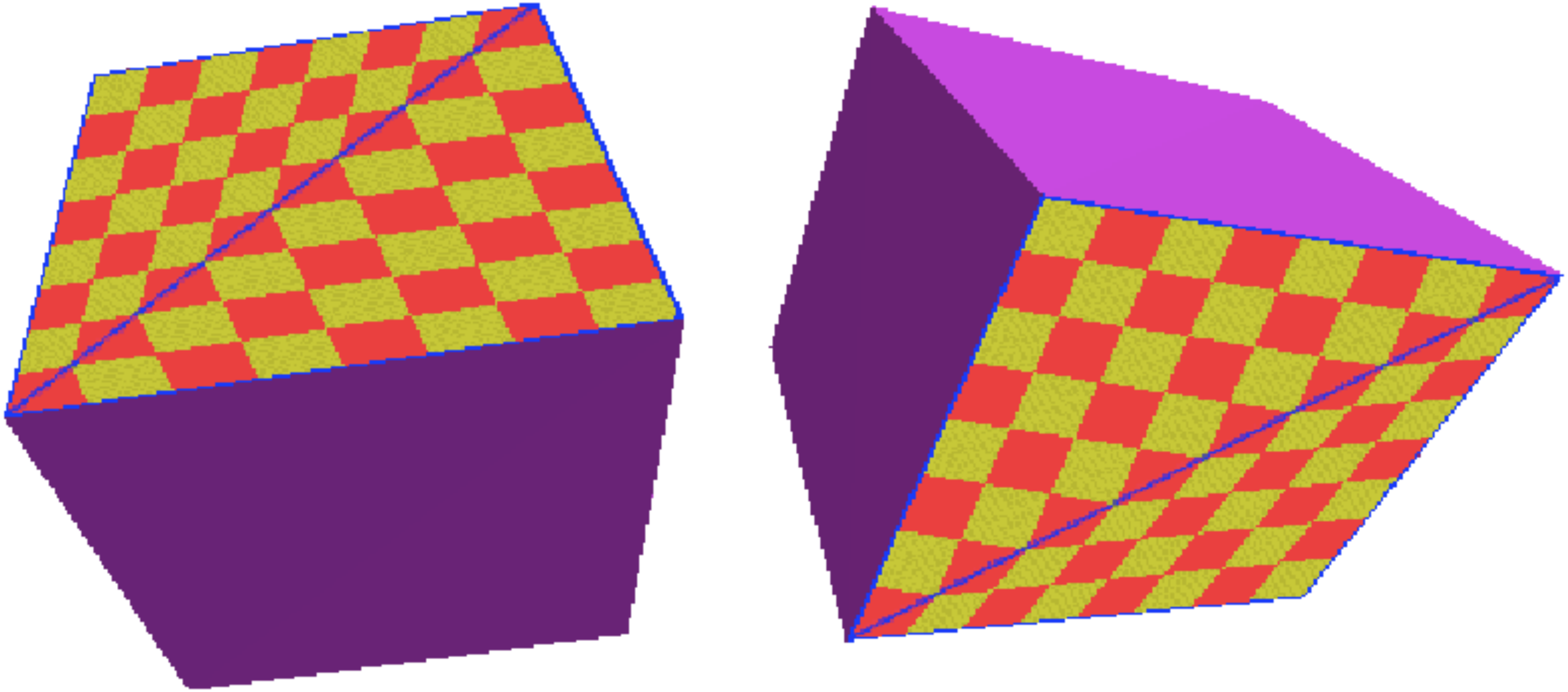
Rasterization

- Rasterization uses scanlines to interpolate a lot of things:
  - Color (Gouraud shading)
  - Depth (depth buffer)
  - Texture coordinates



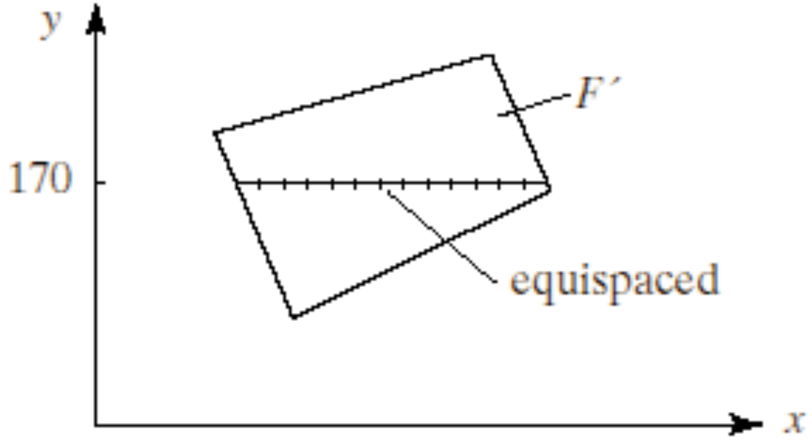
Linear Texture Coordinate Interpolation

- It has artifact when using perspective projection and large polygons. Textures are warped! Very noticeable in animation.

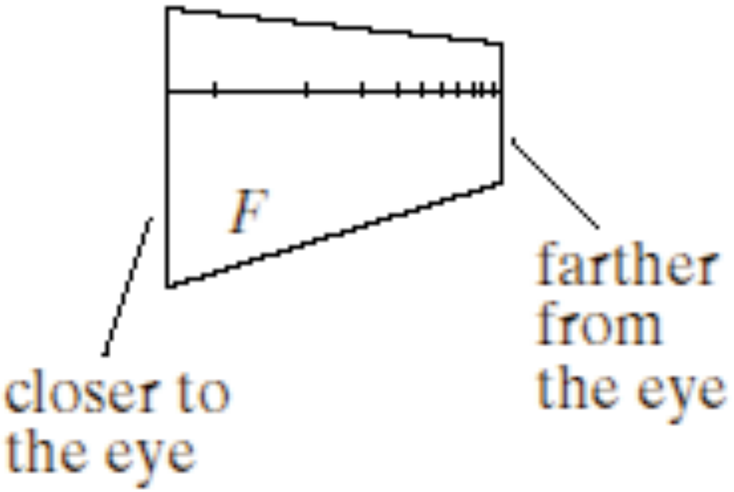


Linear Texture Coordinate Interpolation

- This is because perspective projection has foreshortening effect. Linear interpolation in 2D does not consider this.



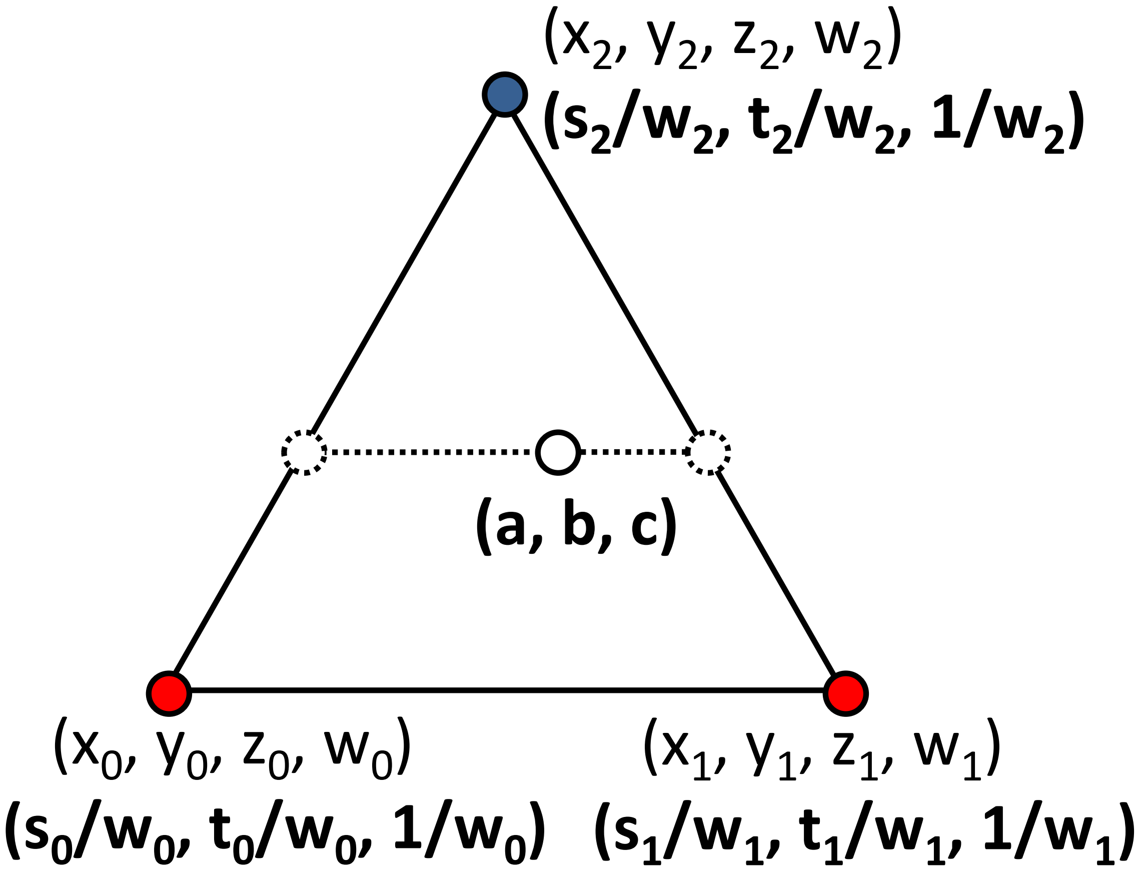
In a 2D view



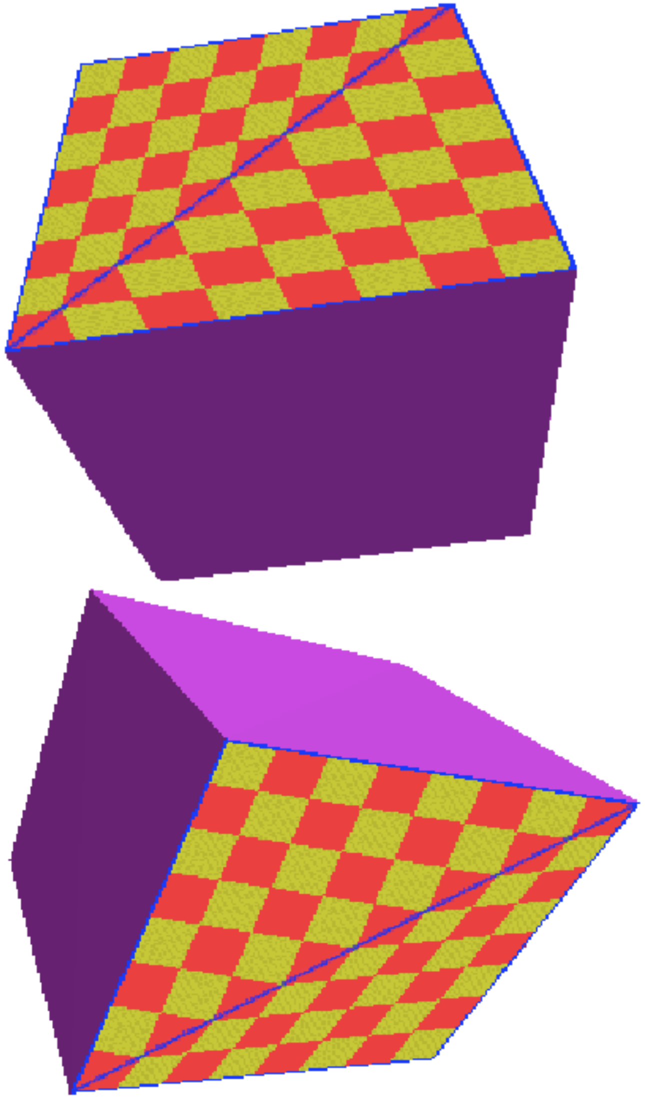
In a 3D perspective view

Solution

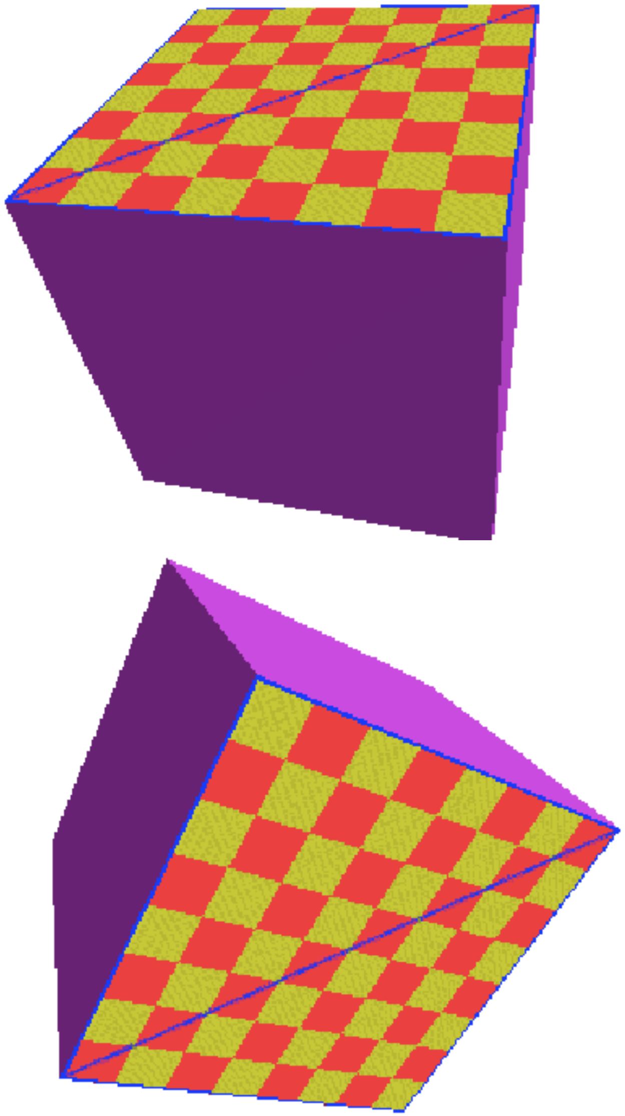
- Let w be the homogenous coordinate.
  - Interpolate (s/w, t/w, 1/w) to get three values: (a, b, c).
  - The final result is: (a/c, b/c).


(x_2, y_2, z_2, w_2)
$\mathbf{(s_2/w_2, t_2/w_2, 1/w_2)}$
$\mathbf{(a, b, c)}$
(x_0, y_0, z_0, w_0)
$\mathbf{(s_0/w_0, t_0/w_0, 1/w_0)}$
(x_1, y_1, z_1, w_1)
$\mathbf{(s_1/w_1, t_1/w_1, 1/w_1)}$

Perspective Correction



Before Correction



After Correction

Perspective Correction

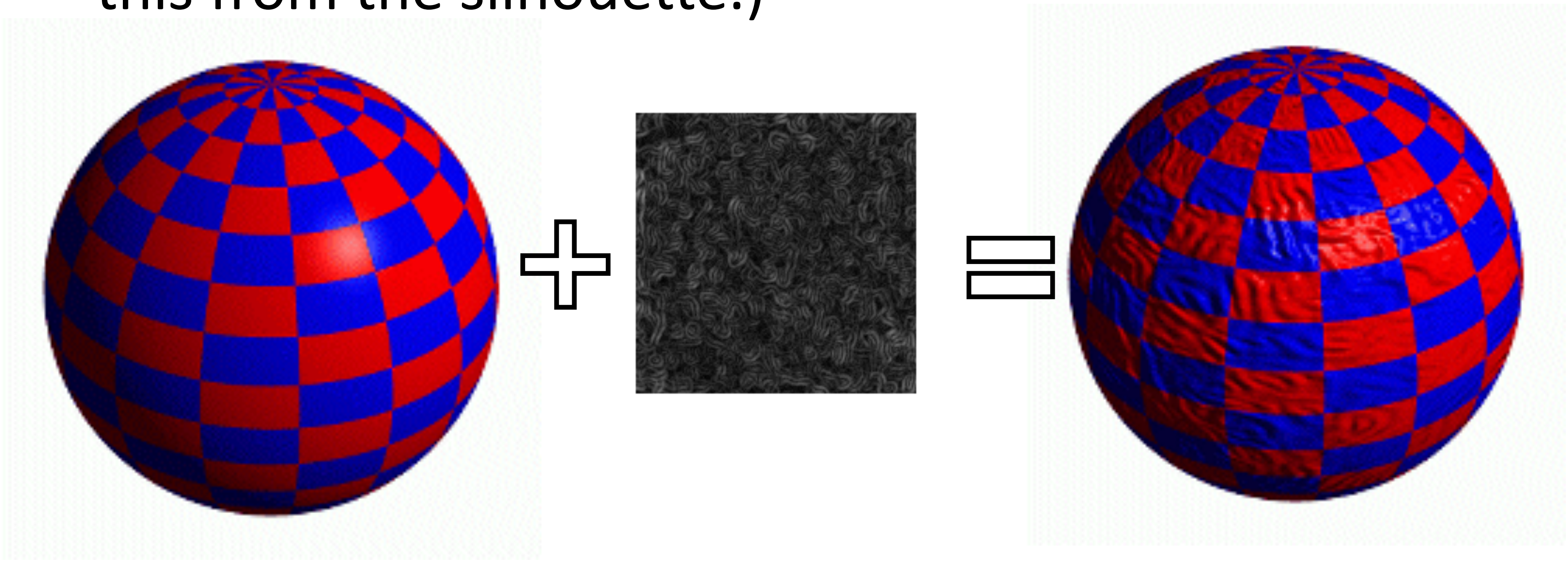
- To enable perspective correction:

```
glHint(GL_PERSPECTIVE_CORRECTION_HINT, hint)
```

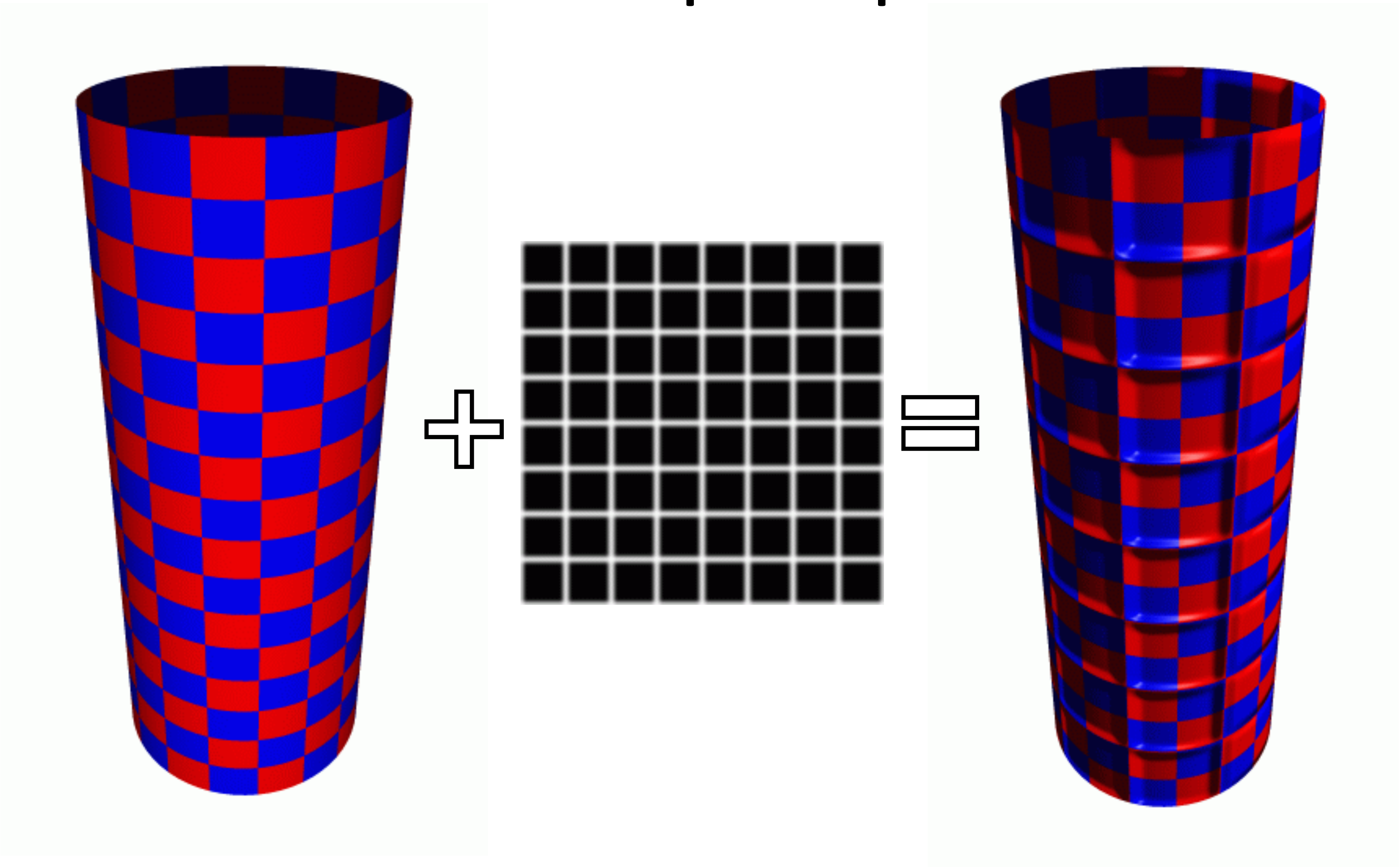
- Hint can be:
  - GL_NICEST (with correction, sloow)
  - GL_FASTEST (linear)
  - GL_DONT_CARE (linear)

Advanced Textures

- OpenGL only uses textures to change surface colors.
- But textures can have other usages.
- For example, the bump map that changes the surface normal. (The geometry stays the same. You can tell this from the silhouette.)

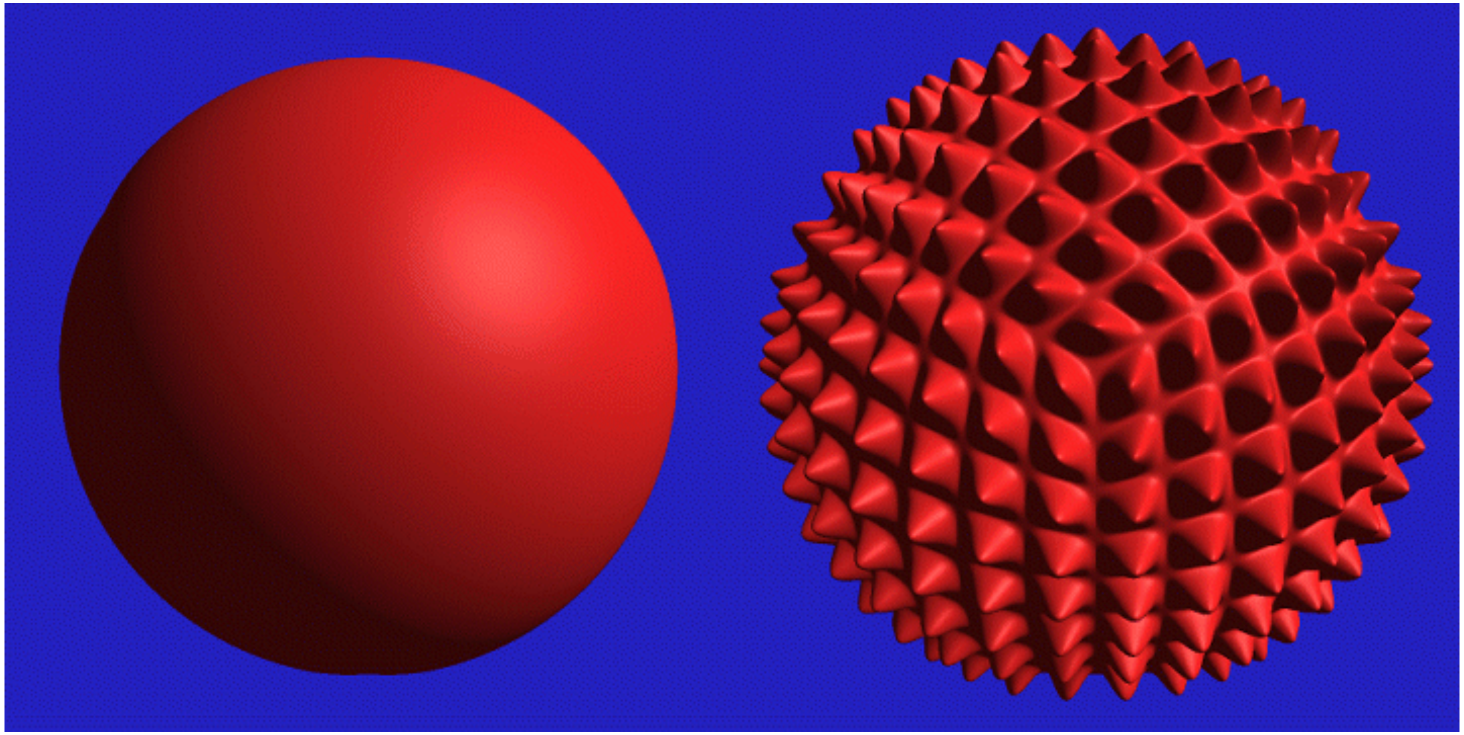


Bump Map



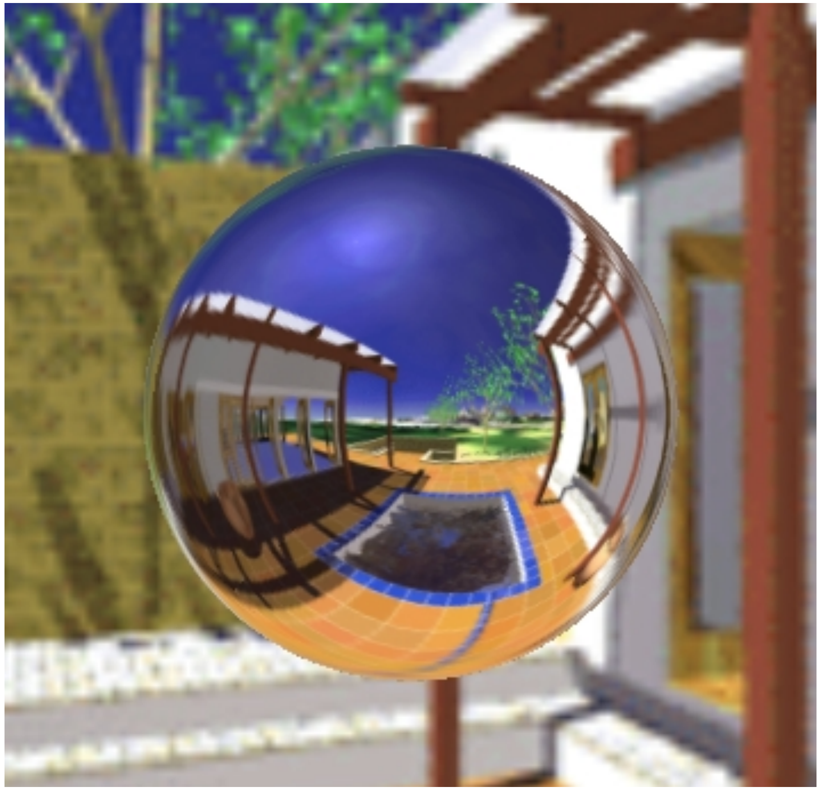
Displacement Map

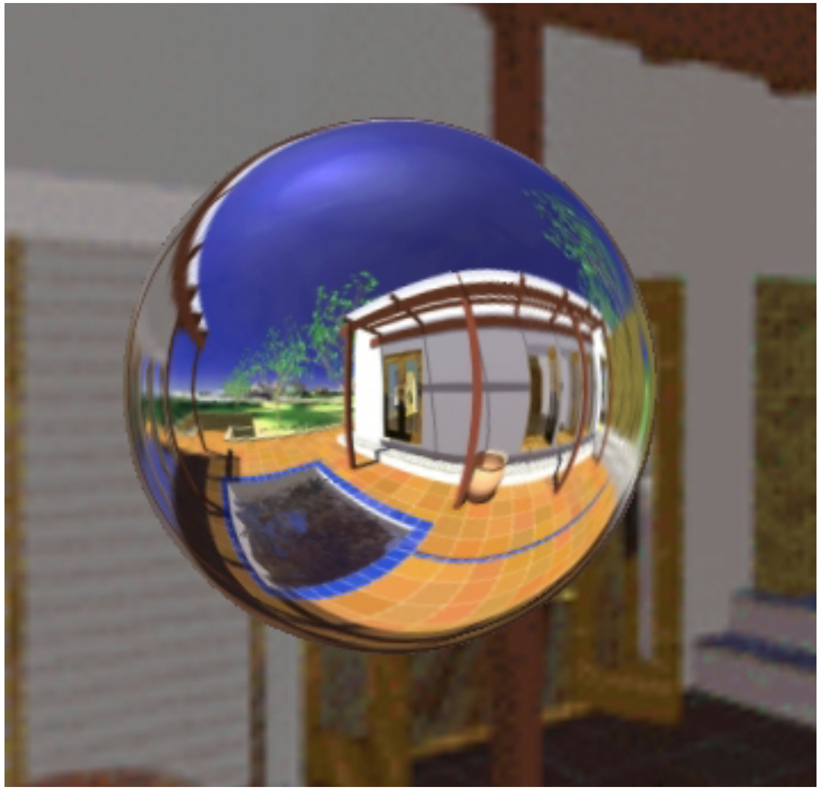
- You can even change the geometry, by treating the texture as an offset map to the vertex position.



Environment Map

- You can surround the 3D space with a texture image too. You can consider the 3D space is contained in an infinite sphere, textured by an image.





Texture Animation

- Animate the texture over time
  - Apply transformations to texture coordinates
  - Change textures
    (but try to avoid multiple texture loadings)

A fountain in the game “world of Warcraft”