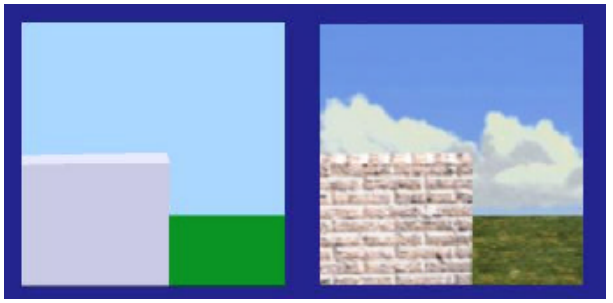
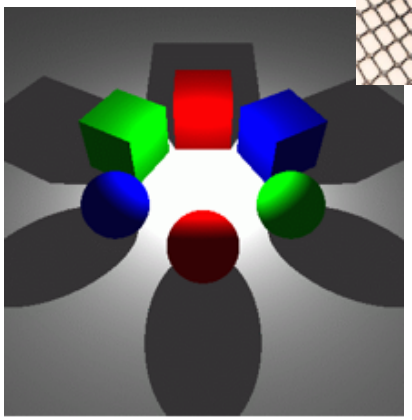
Textures and shadows

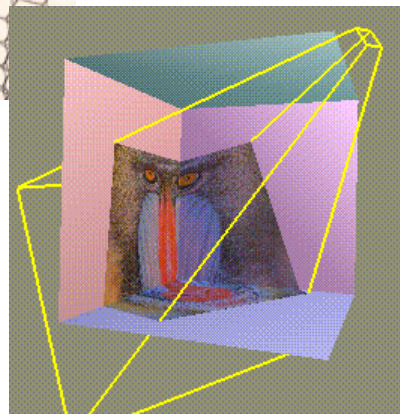
Georgia Tech College of Computing
Interactive & Intelligent Computing

- Generation
- Mipmap
- Texture coordinates, Texgen
- Rendering, Blending
- Environmental, bump mapping
- Billboards
- 3D (solid) textures
- Shadows



Eric Wernert





Motivation

- **Texture mapping:** Apply (paint) image on a polygon
- Add realism to the scene without having to design and represent detailed geometry
  - Reduces modeling cost
  - Adds realism
    - Wood, fabric
  - Improves performance
    - Fewer polygons to transform

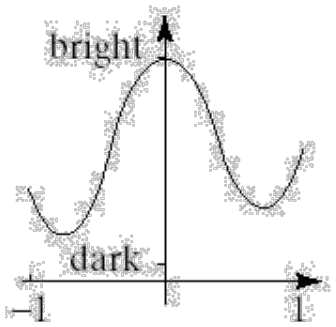
Eric Wernert

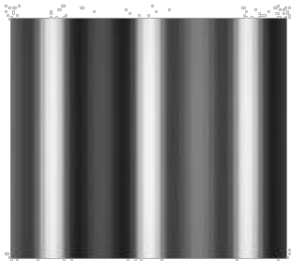
Process for using texture mapping

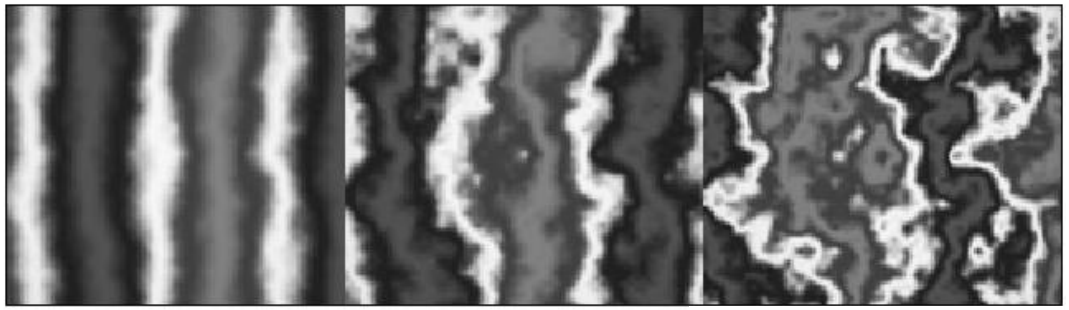
- Create an **image** for texture
  - **Load** image of texture (its pixels are called *texels*)
    - May be generated procedurally
  - Create **mipmap** (multi-resolution pyramid of textures)
- Assign texture **coordinates** to vertices
  - In the application or using transforms
- Enable texturing
  - Define context and select texture
  - Specify how to combine (blend) **texture** with **shading color**
- Render the object
  - Send texture coordinates with each vertex

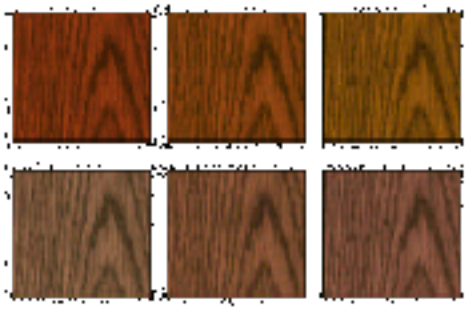
Texture generation

- Images (photos, paintings)
- Math functions
- Image effects, perturbations
- Tiling (seems?)
- Procedural
  - On mesh
  - In 3D

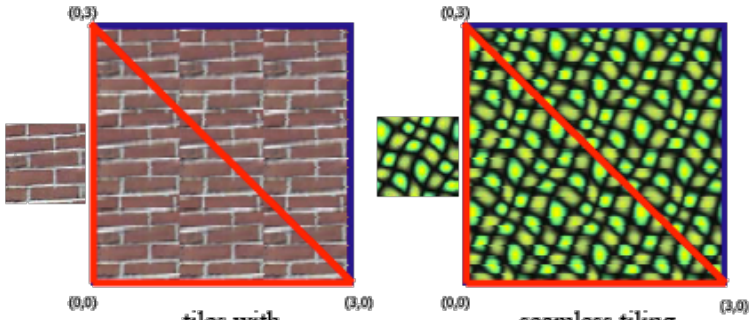








Eric Wernert



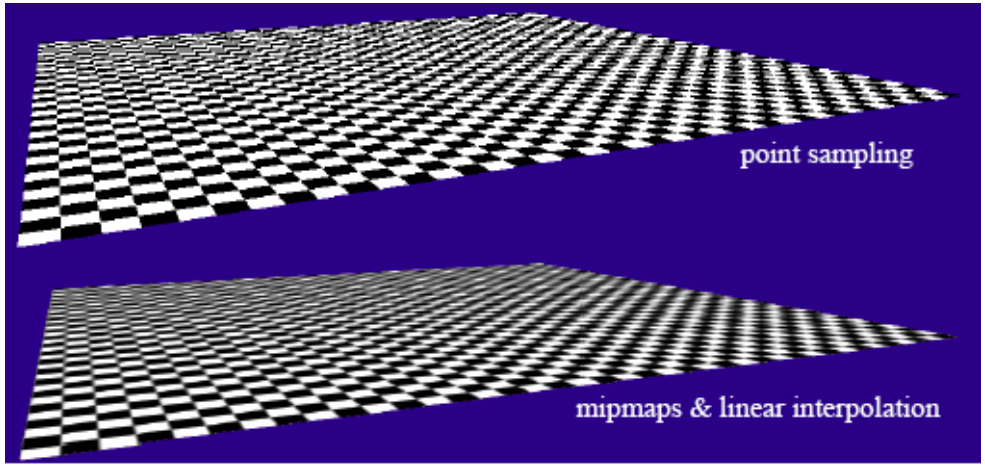
Durand&Cutler, MIT

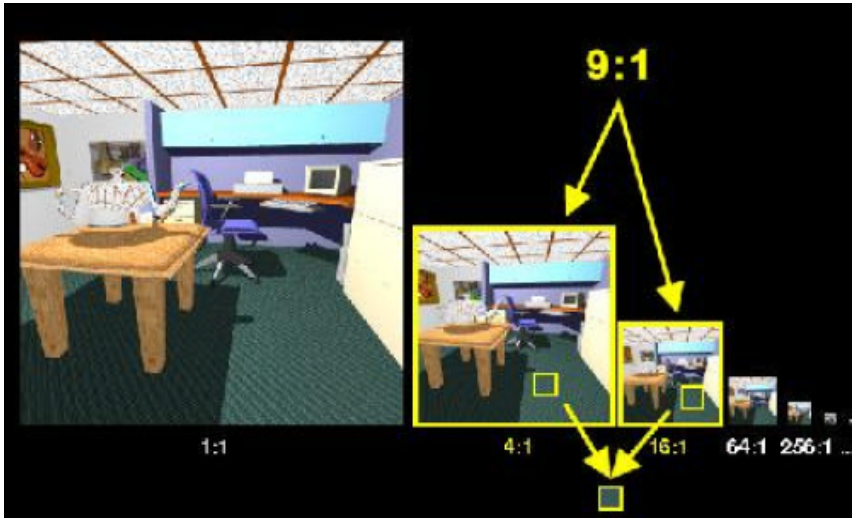
Mipmap

Used to reduce aliasing

when objects are small, a slight motion would assign a different texel to the same pixel

- Computed using 2×2 averaging
- Computed by OpenGL: gluBuild3DMipmaps()





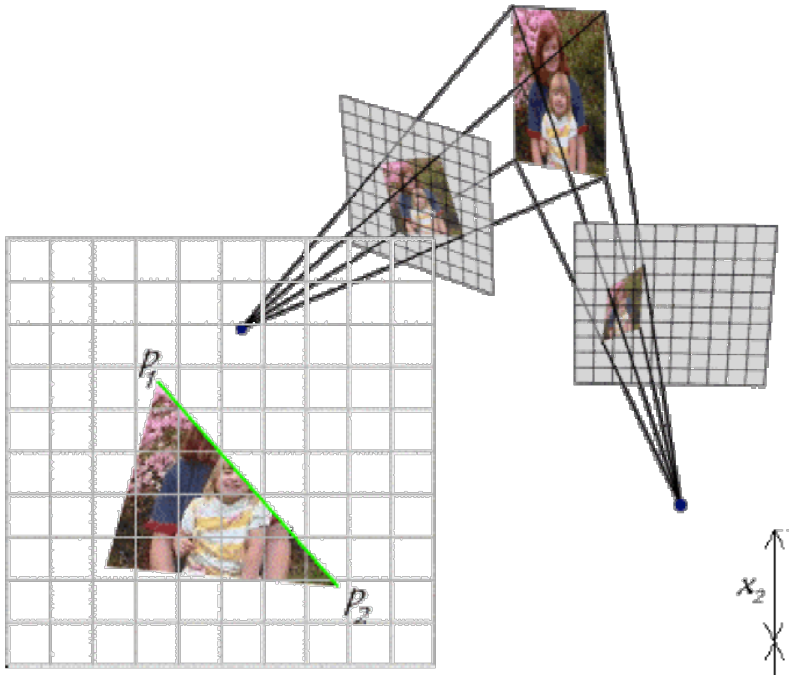
Assign/bind texture

glTexImage2D(target, level, internalFormat, width, height, border, format, type, *texels);

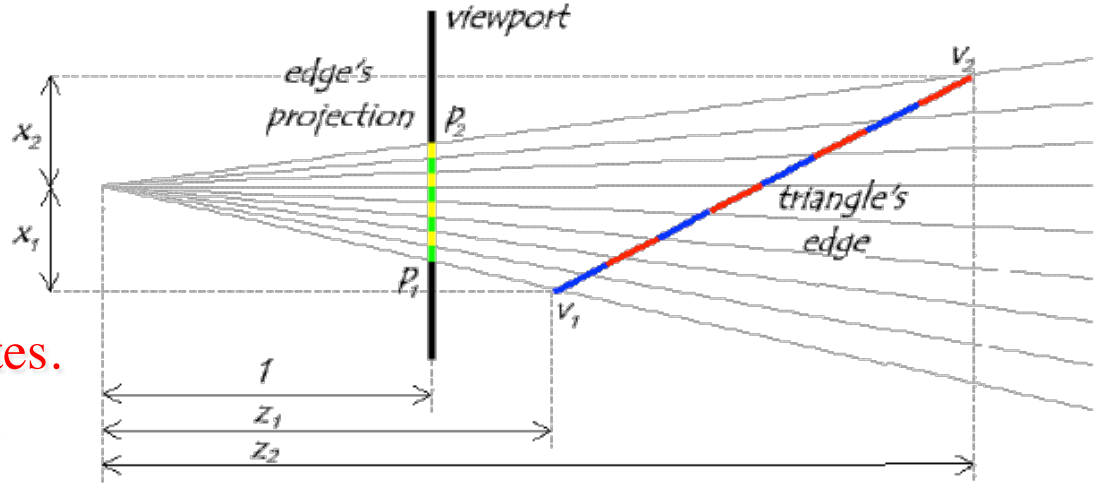
- `target` = GL_TEXTURE_2D
- `level` = level of detail for mipmapping; default is 0
- `internalFormat` = GL_RGB, GL_RGBA, …
- `width, height` = dimensions of image (powers of 2)
- `border` = used when repeating is enabled, default = 0
- `format` = GL_RGB, GL_RGBA, etc.
- `type` = GLubyte, GLuint,
- `texels` = pointer to actual data

Perspective **distortion** of texture

- When using linear interpolation of (s,t) in screen space



texture source what we get what we want



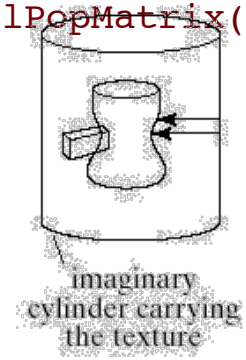
The rasterizer uses linear interpolation for texture coordinates. Perspective projection of uniform sampling is not uniform.

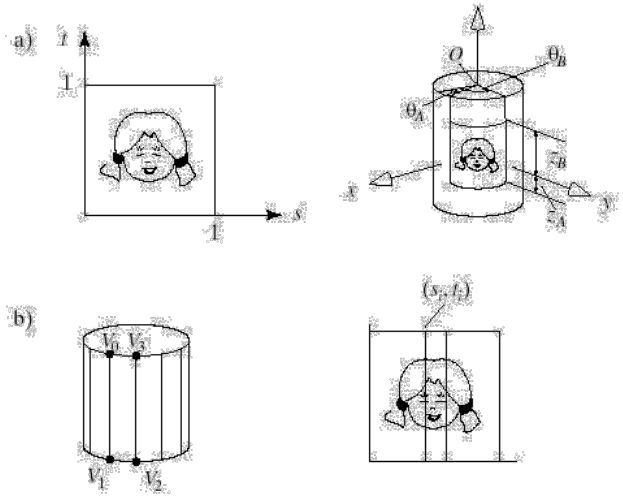
Durand&Cutler, MIT

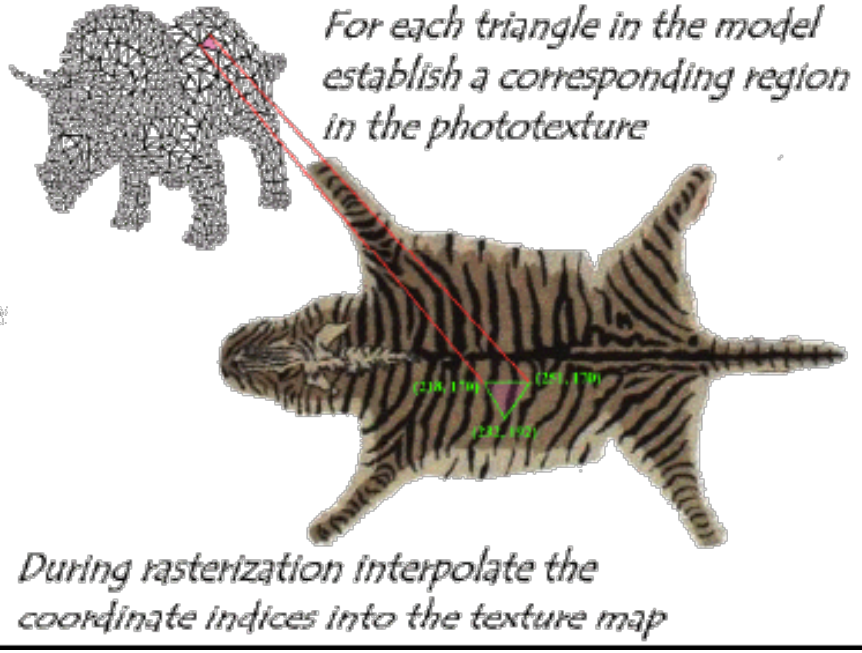
Texture coordinates

- Assign texture coordinates (s,t) in [0,1] to each **vertex**
  - **Hard-coded** by developer in your program
  - Assigned **procedurally** by application
  - **Computed** by OpenGL from vertex location & normal

```
glMatrixMode(GL_TEXTURE);
glPushMatrix();
glTranslatef(..); glRotatef(..)..
glBegin(GL_POLYGON);
  glTexCoord2f(0.2, 0.5);
  glVertex3f(1.0, 2.5, 1.5);
  …
   glEnd();
glPopMatrix();
```





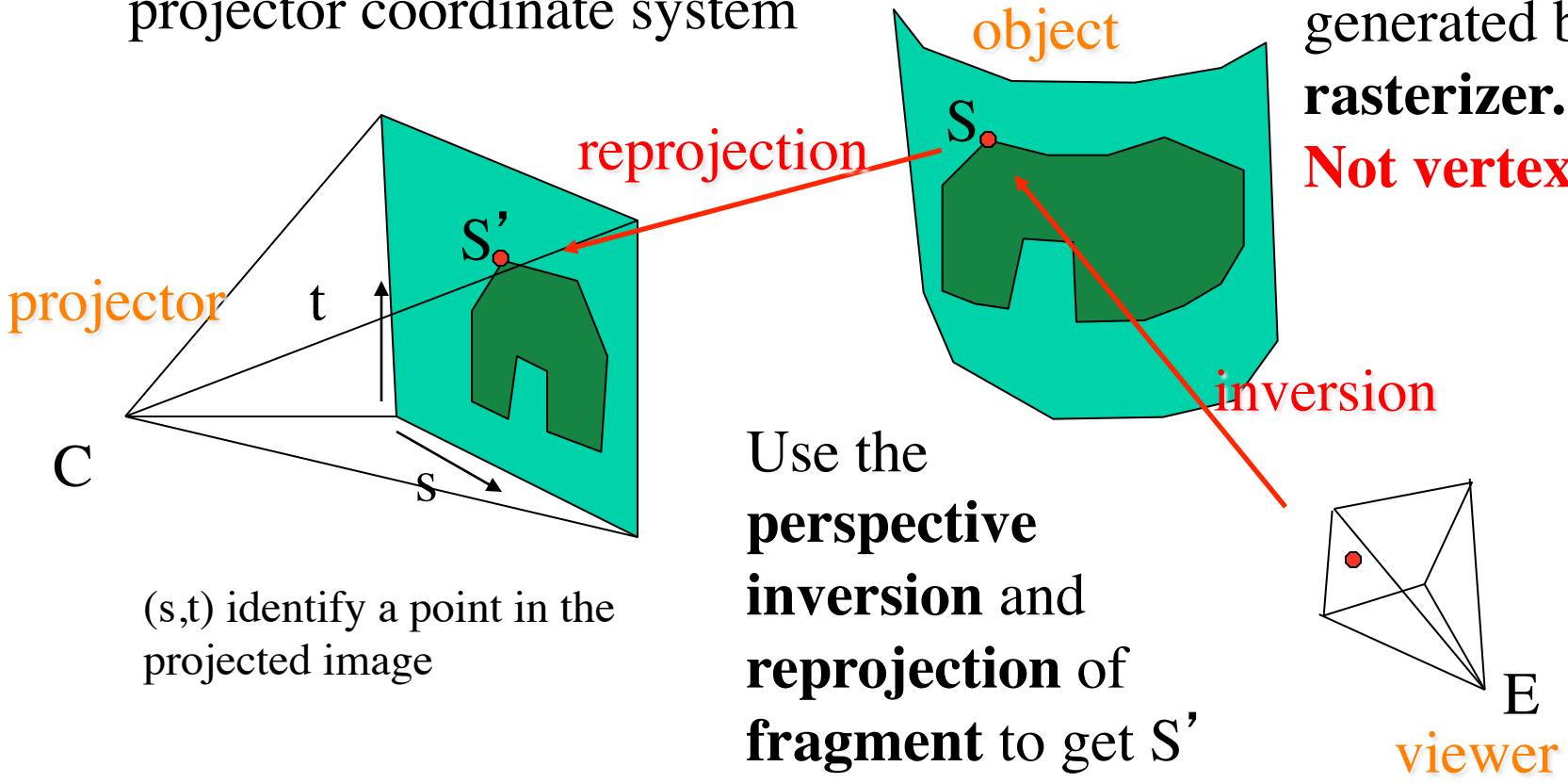


Eric Wernert

Computed texture (using reprojection)

- Assume projector C projecting an image into an object

Need (s,t) for S in the projector coordinate system

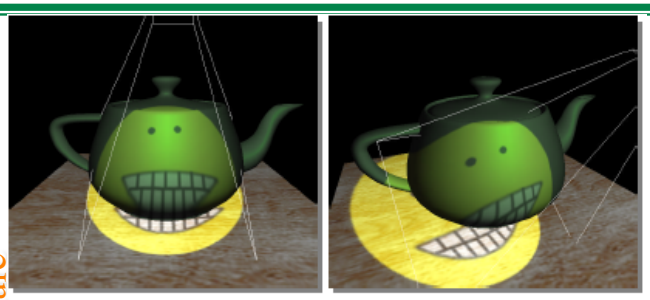


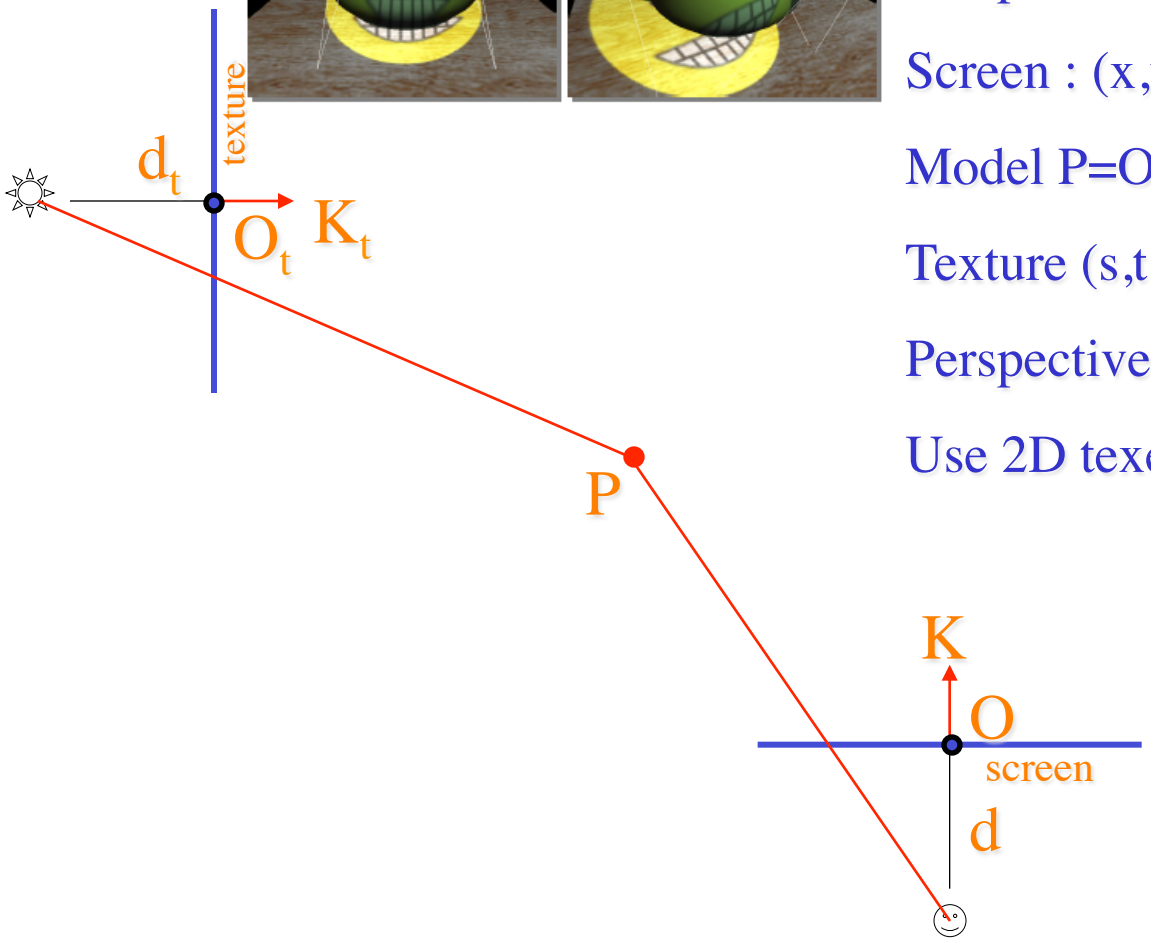
S **fragment** (pixel+depth) generated by **rasterizer.** **Not vertex!**

(s,t) identify a point in the projected image

Use the **perspective inversion** and **reprojection** of **fragment** to get S'

Correct interpolation of reprojected texture





For each **fragment:**

Perspective coordinates of P: x', y', z'

Screen : $(x,y,z)=(x', y', z')/(d-z')$

Model $P=O+xI+yJ+zK$

Texture $(s,t,r) = (O_tP \bullet I_t, O_tP \bullet J_t, O_tP \bullet K_t)$

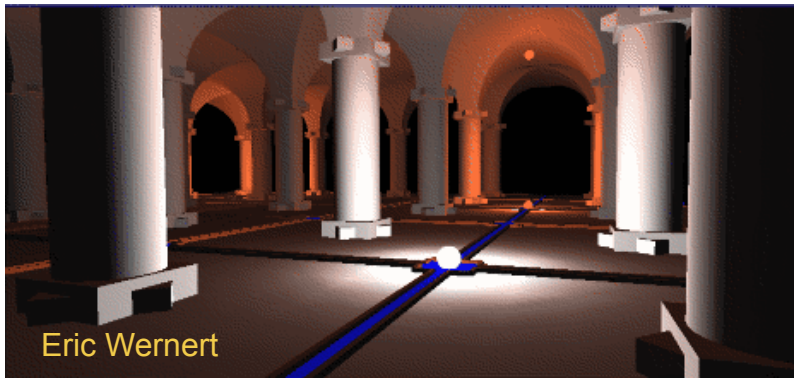
Perspective texture $(s', t', r') = (s,t,r)/(d_t+r)$

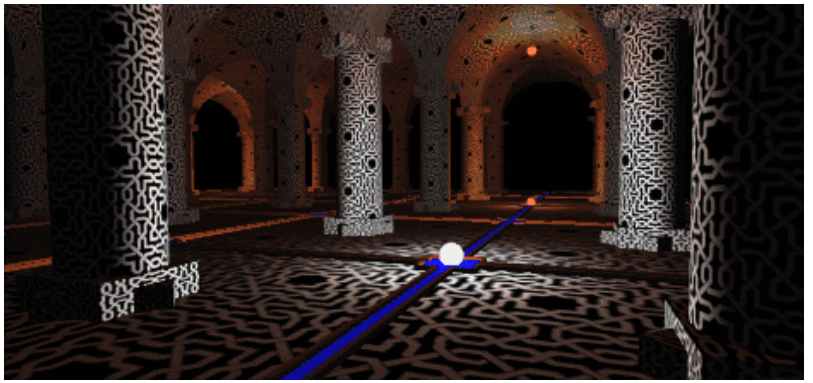
Use 2D texel (s', t') or 3D texel (s', t', r')

http://www.nps.navy.mil/cs/sullivan/MV4470/resources/projective_texture_mapping.pdf

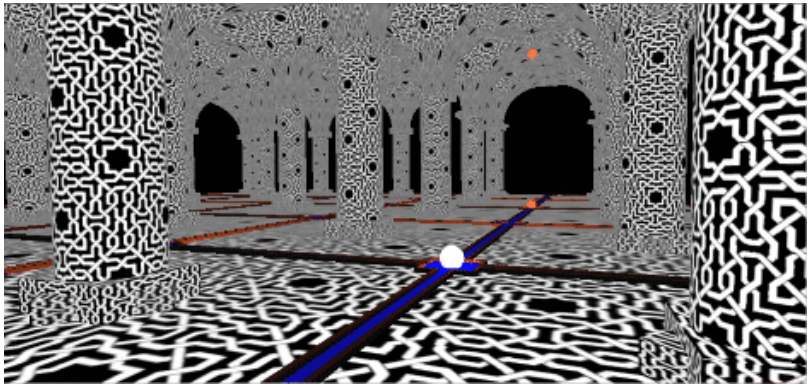
Blending texture and shaded color

- You can blend texture color with the shaded color (reflection)

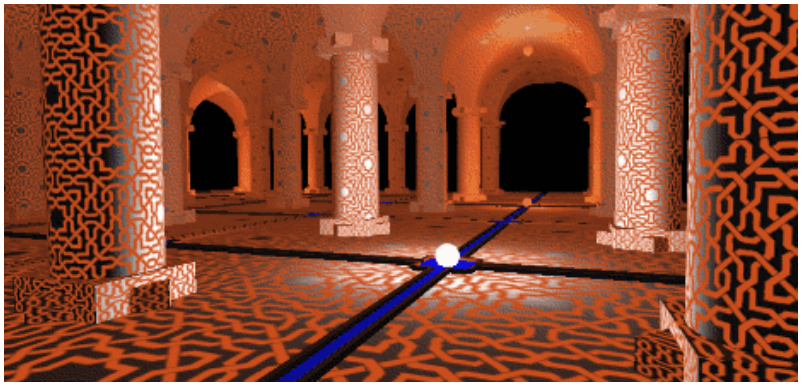




Modulate (default) multiplies shaded color by texture color. Alpha of texture modulates transparency.



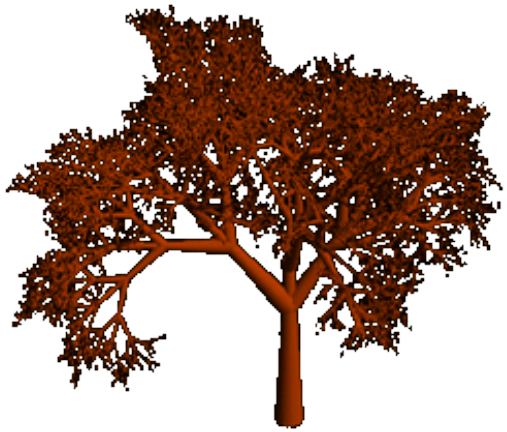
Decal replaces shaded color by texture color. Alpha of texture allows shaded color to show through.

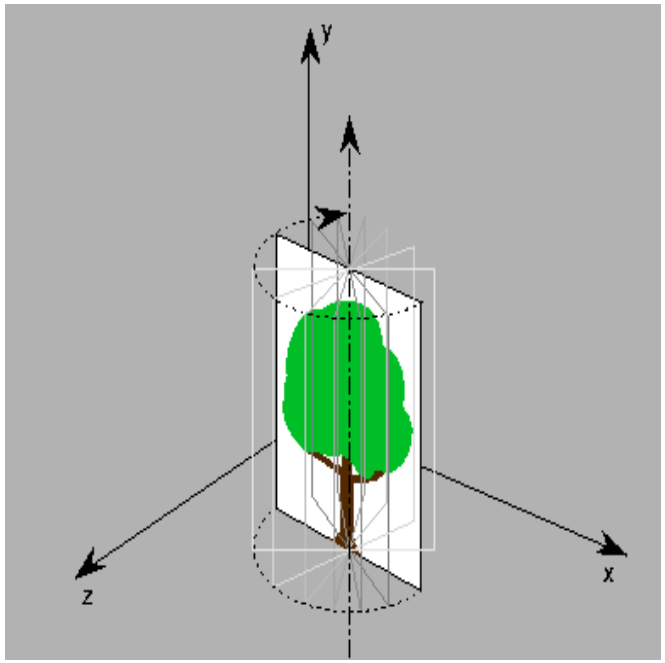


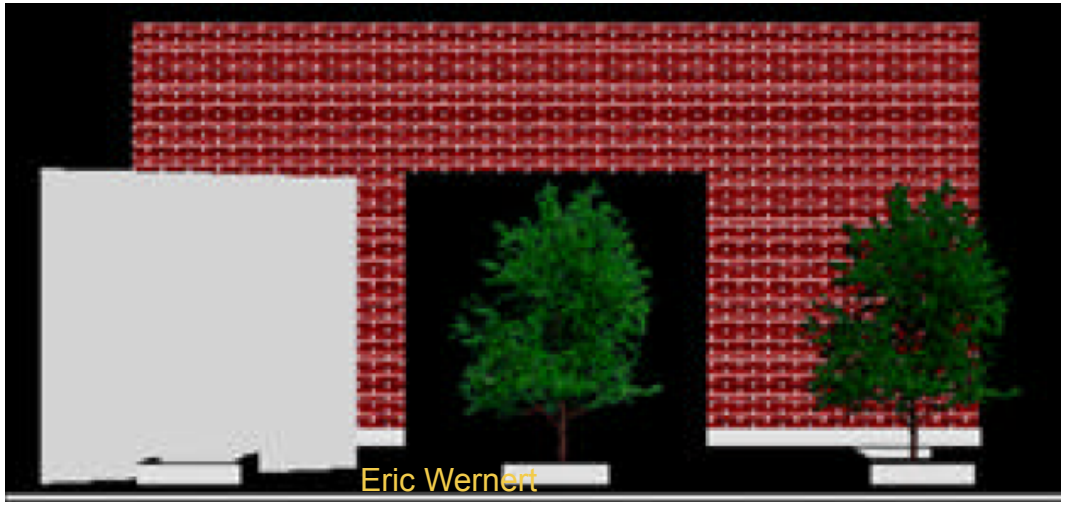
Blend texture intensity controls blend between shaded color and specified constant color.

Billboards

- For trees
- Always orient towards viewer
  - Or select image based on orientation
- Use transparency

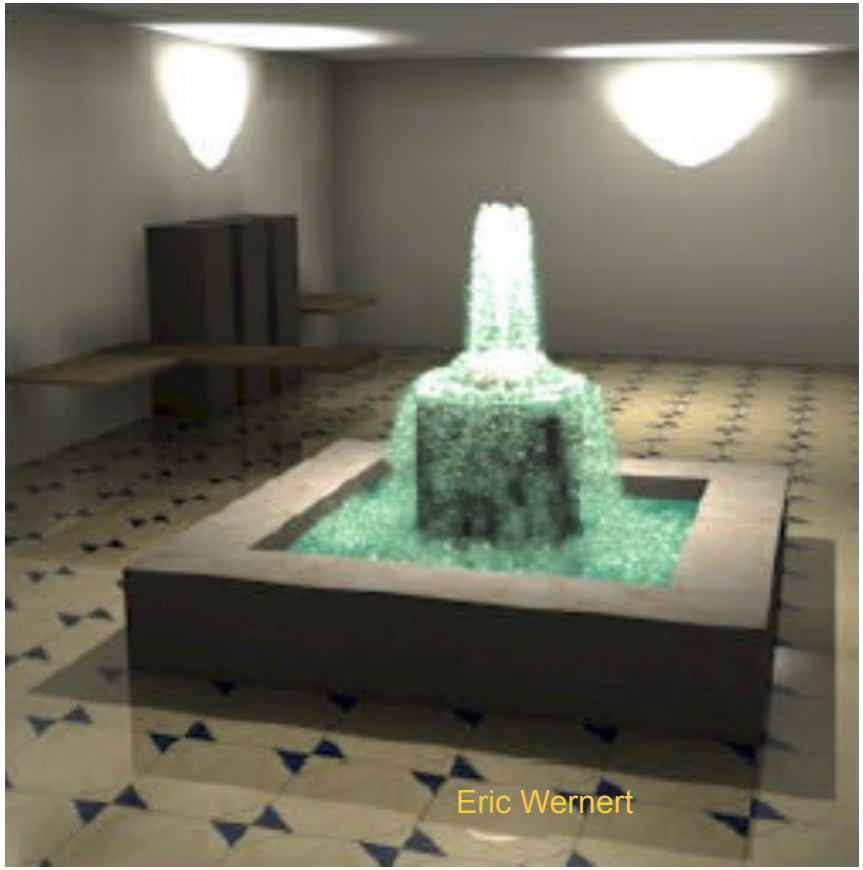






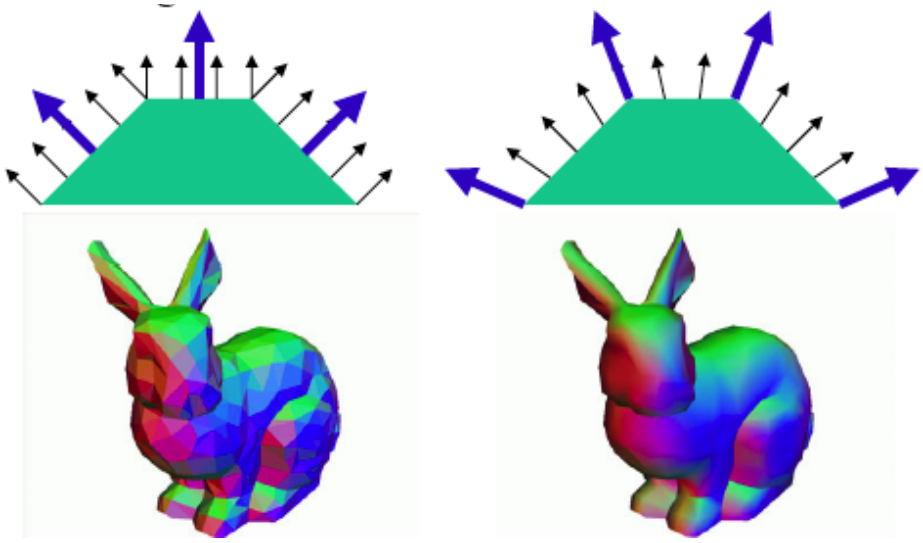
Particles

- Use particles as small billboards to render liquid, fire, smoke



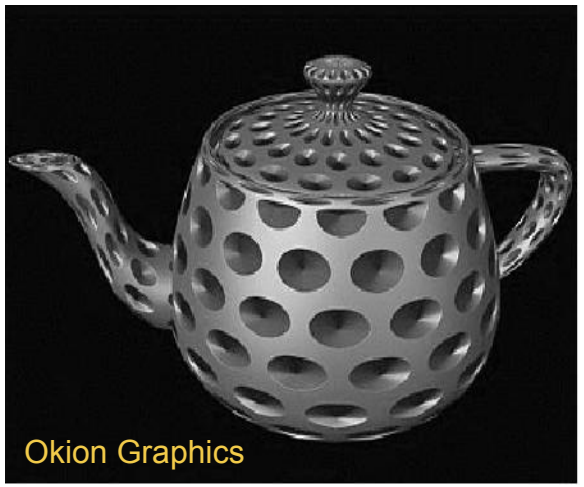
Phong normal interpolation

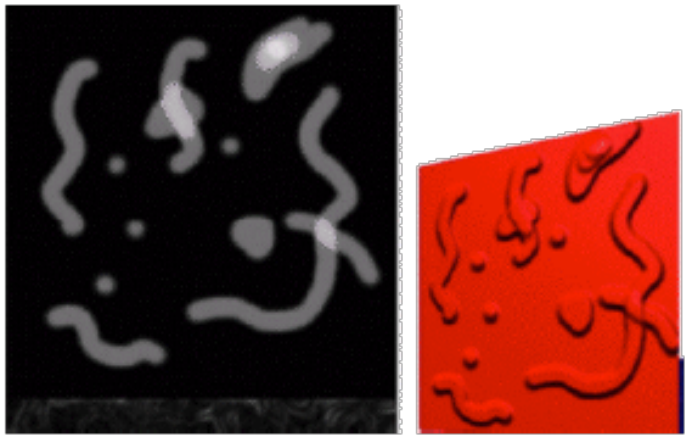
- Flat shading produces sharp ridges along edges
- Gouraud smooth shading **interpolates vertex colors** across the triangle, but misses highlights
- **Phong shading interpolates normals** across the triangle and uses each fragment's normal for lighting



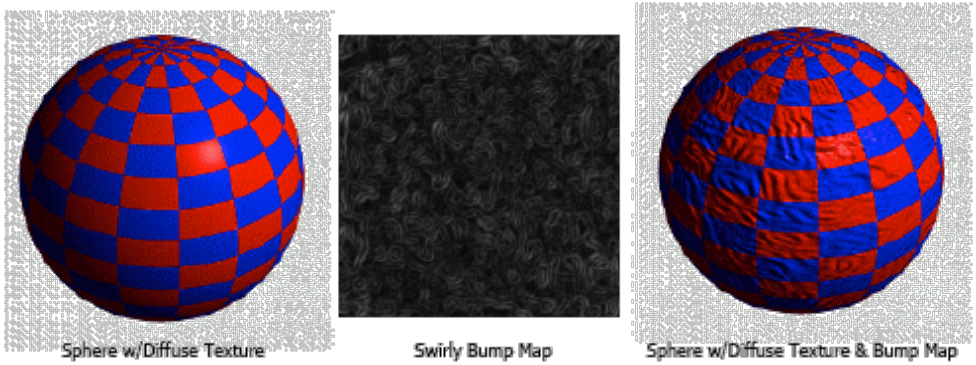
Bump mapping

- Intensity of texture used to **perturb fragment's normal**
  - Height function, derivatives added to normal
- Lighting is performed with the perturbed normal
  - Surface is not modified
  - Inconsistencies along **silhouette**





Durand&Cutler, MIT



Texture coordinates generation (texgen)

- OpenGL can generate texture coordinates automatically for you
  - linear combination of coordinates (eye- or object-space):

    glTexGenf(S, TEXTURE_GEN_MODE, OBJECT_LINEAR)

    $$g = p_1 x_o + p_2 y_o + p_3 z_o + p_4 w_o.$$

    glTexGenf(S, TEXTURE_GEN_MODE, EYE_LINEAR)

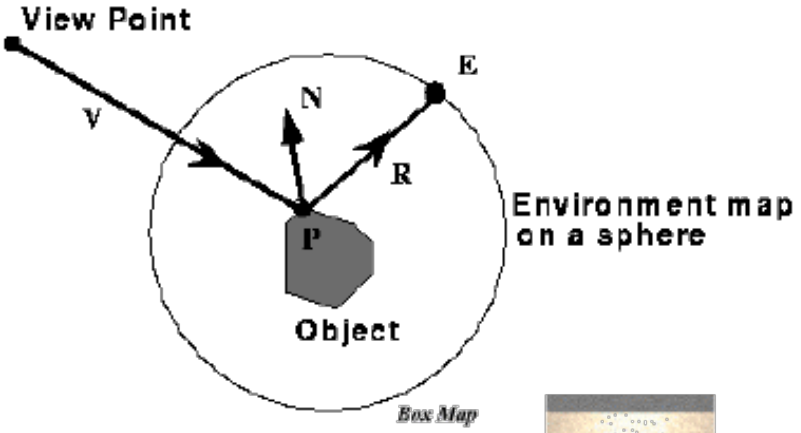
    $$g = p'_1 x_e + p'_2 y_e + p'_3 z_e + p'_4 w_e \quad (p'_1 \quad p'_2 \quad p'_3 \quad p'_4) = (p_1 \quad p_2 \quad p_3 \quad p_4)\, M^{-1}$$

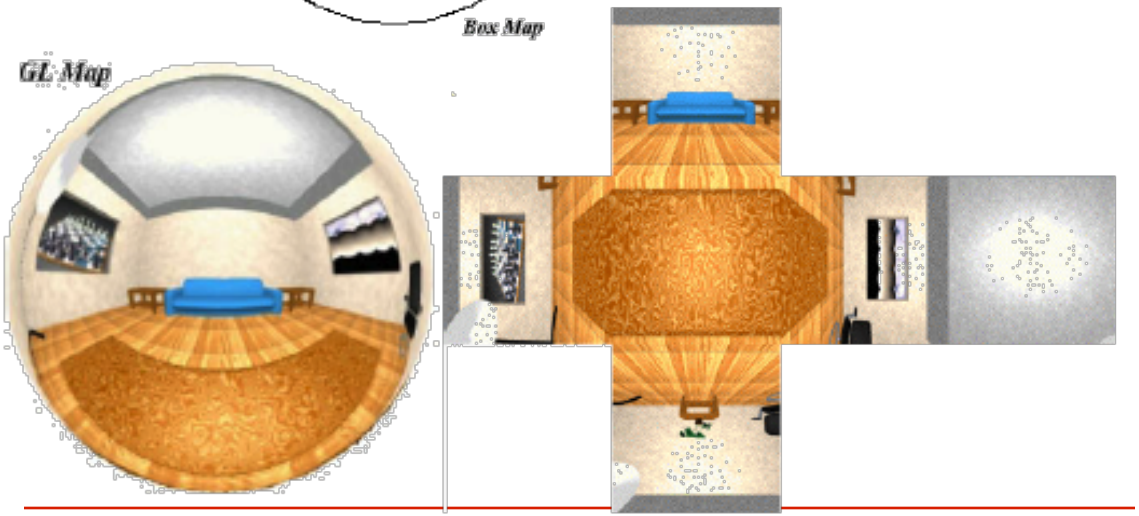
    glTexGenf(S, TEXTURE_GEN_MODE, SPHERE_MAP)

  or using sphere map for environment mapping.

Environmental mapping

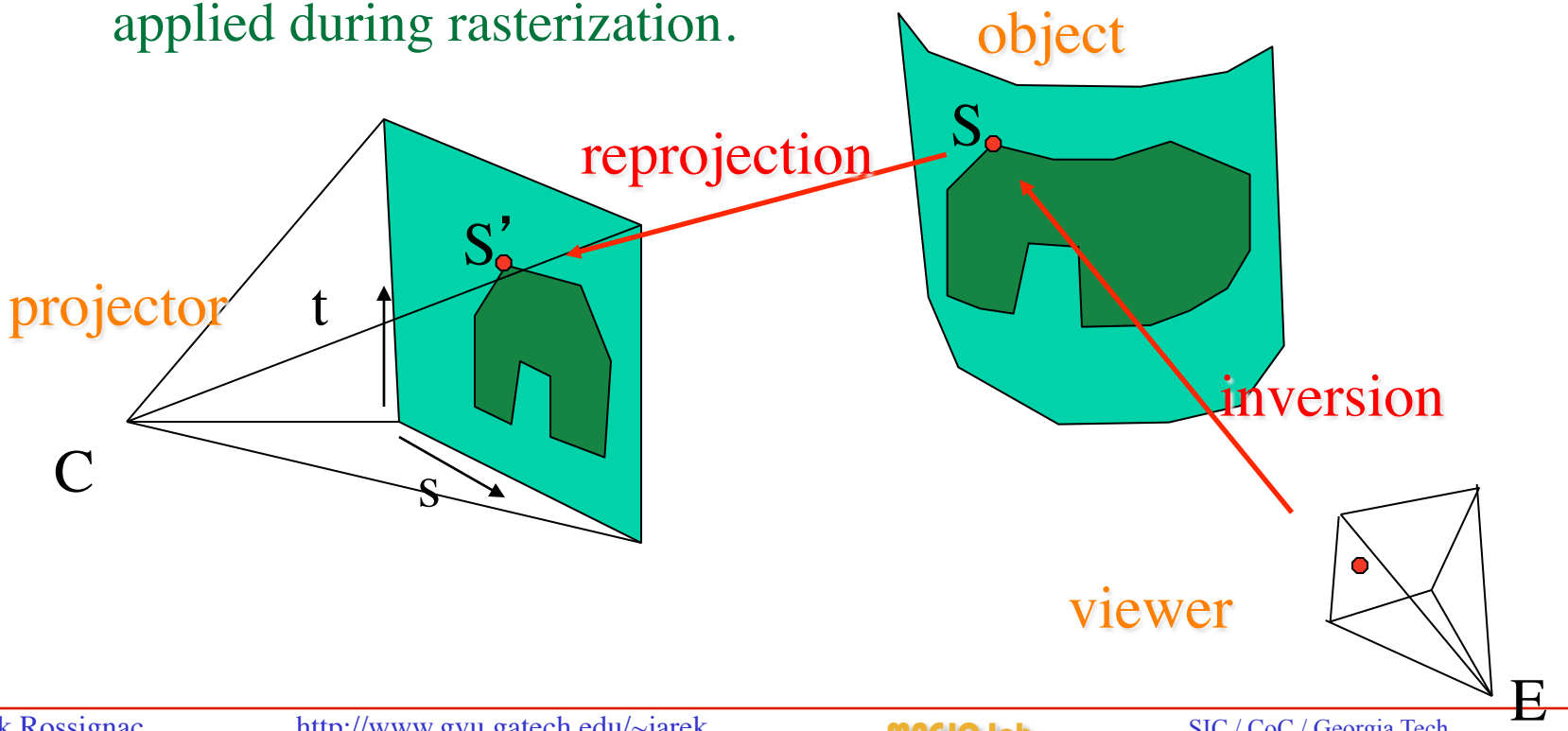
- Developer specifies surrounding shape (cube, sphere) + texture
- GPU uses as texture the color where reflected ray hits the surrounding shape



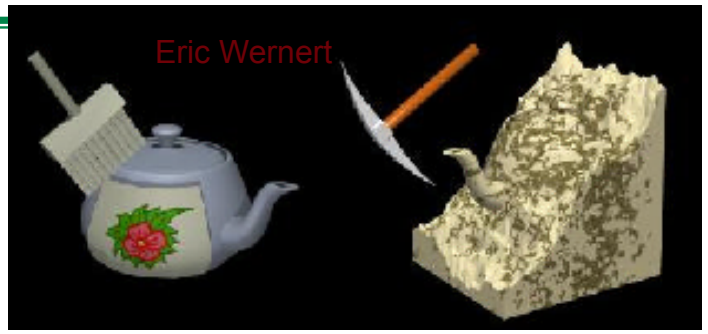


Texture coordinates generation (texgen)

- OpenGL also provides a 4×4 texture matrix
  - It can be used to transform the per-vertex texture coordinates, whether supplied explicitly or implicitly through texture coordinate generation.
  - Rescale, translate, project texture coordinates before the texture is applied during rasterization.

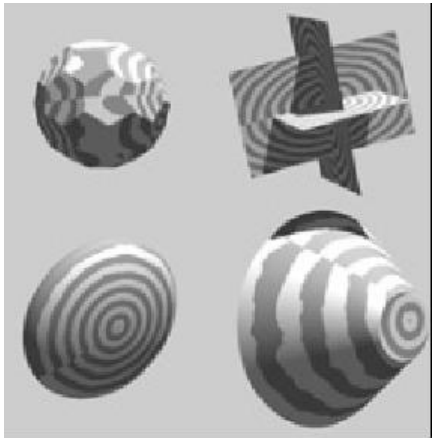


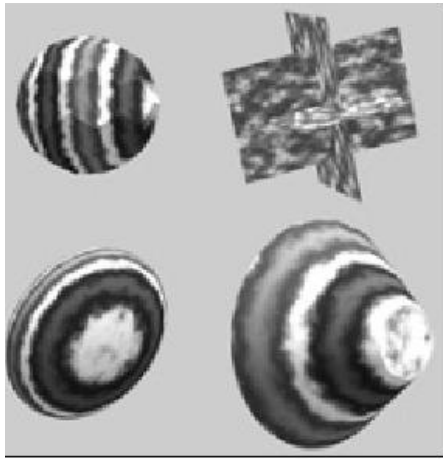
3D (solid) textures



- 3D array of texels T[s,t,r]
- Great for procedural 3D textures
  - objects carved out of wood, marble
- Mipmap must be provided by application 2×2×2 averaging
- Enable 3D texture mapping

  glEnableGL_TEXTURE_3D_EXT(GL_TEXTURE_3D_EXT)
- Transform texture by texture matrix as desired





Resources

Mark Kilgard (Nvidia) with sample program

http://www.opengl.org/resources/code/samples/mjktips/projtex/index.html

"Fast Shadows and Lighting Effects Using Texture Mapping" Segal, Korobkin, van Widenfelt, Foran, Haeberli. SIGGRAPH 1992.

Eric Wernert (Indiana): http://www.avl.iu.edu/~ewernert/b581/lectures/15.1/index.html

Texture generation:

http://www.opengl.org/resources/code/samples/redbook/texgen.c

http://www.opengl.org/documentation/specs/version1.1/glspec1.1/node27.html