

Texture Mapping and Special Effects

February 4th and 7th 2009

MAE 574, Virtual Reality Applications and
Research

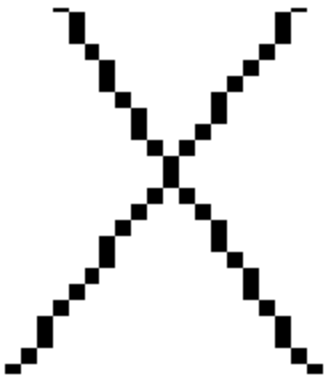
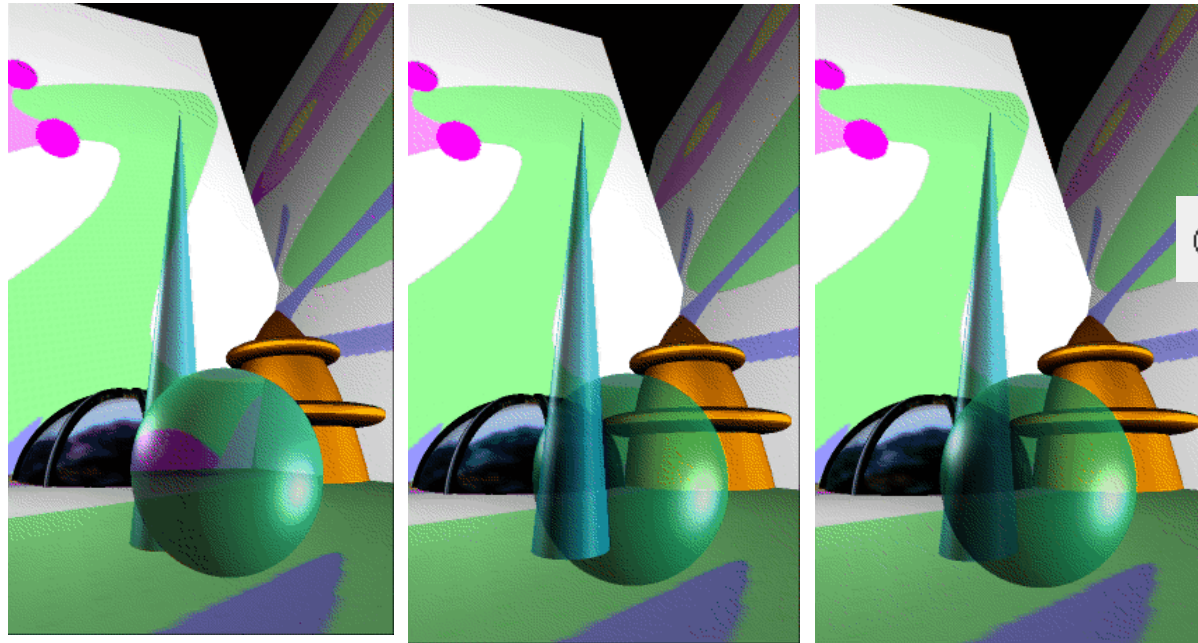
Instructor: Govindarajan Srimathveeravalli

Blending – Anti Aliasing

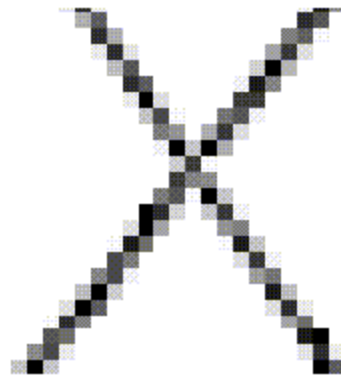
Images courtesy
Inventor Mentor

$(R_s S_r + R_d D_r, G_s S_g + G_d D_g, B_s S_b + B_d D_b, A_s S_a + A_d D_a)$

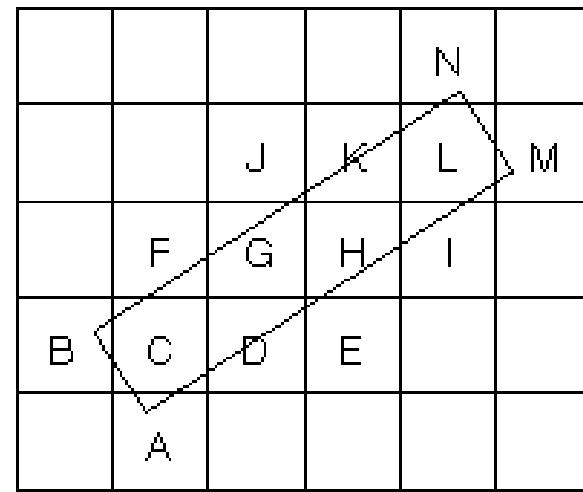
- Keyword `GL_BLEND`
- `glBlendFunc(source, dest)`
- Also, stipple and stencils



Aliased



Antialiased



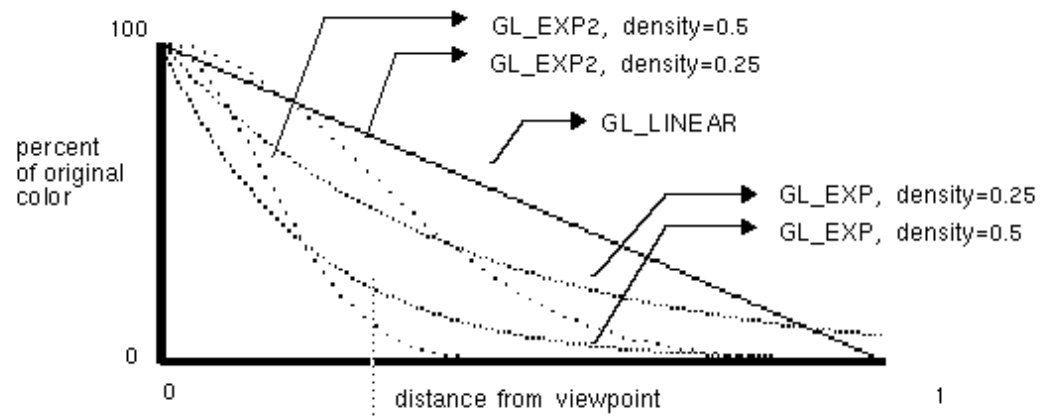
Fog, Smoke etc.

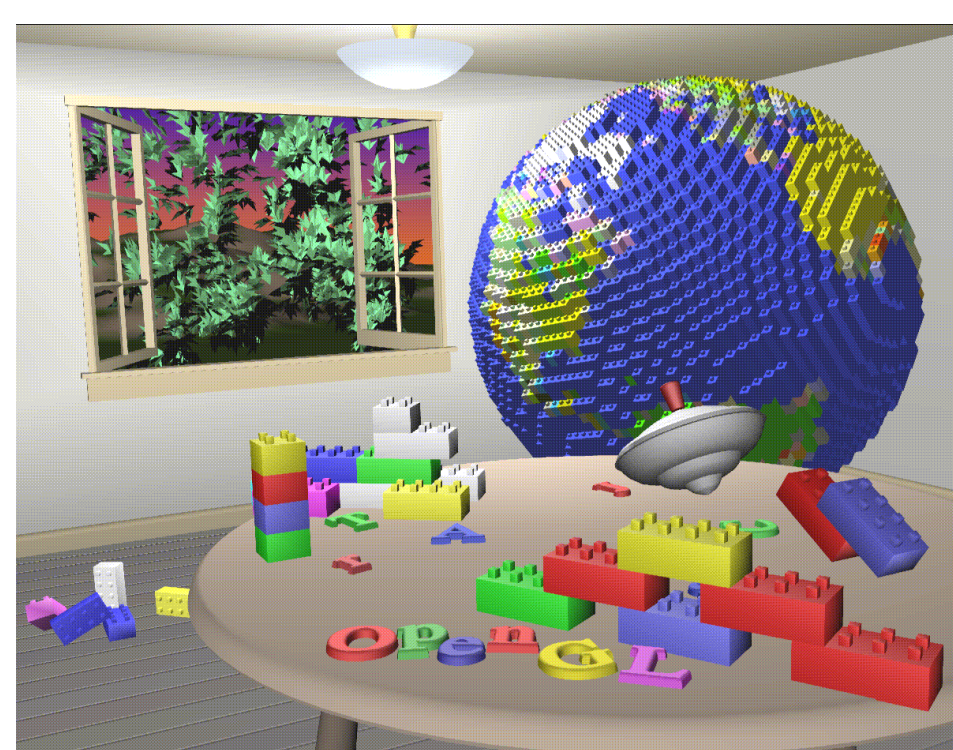
- glEnable(GL_FOG)
- Again achieved through carefully blending object pixel color with fog color
 - Color
 - Quality of fog
- Other parameters
 - Fog hint

$$f = e^{-(density \cdot z)} \quad (GL_EXP)$$

$$f = e^{-(density \cdot z)^2} \quad (GL_EXP2)$$

$$f = \frac{end - z}{end - start} \quad (GL_LINEAR)$$





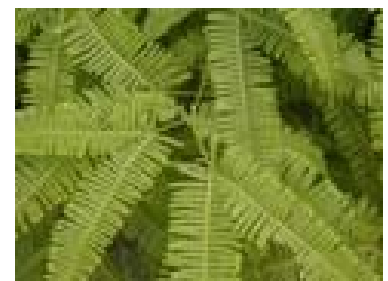
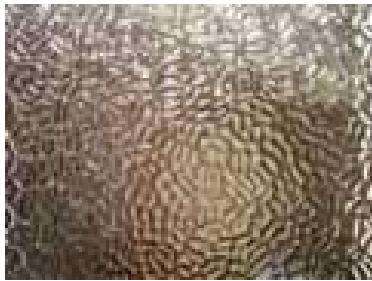
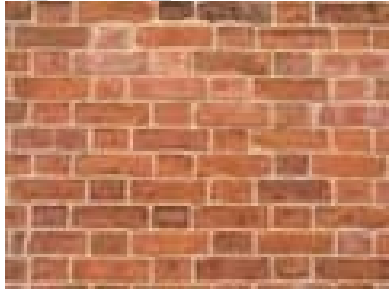
Shaded – Lit polygons

Textured polygons



How to create textures:

- ✓ Create your own in code!
- ✓ Models are available online in texture “libraries” of cars, people, construction materials, etc.



- ✓ Custom textures from scanned photographs or
- ✓ Using an interactive paint program to create bitmaps

Textures courtesy mayang.com

Texture mapping

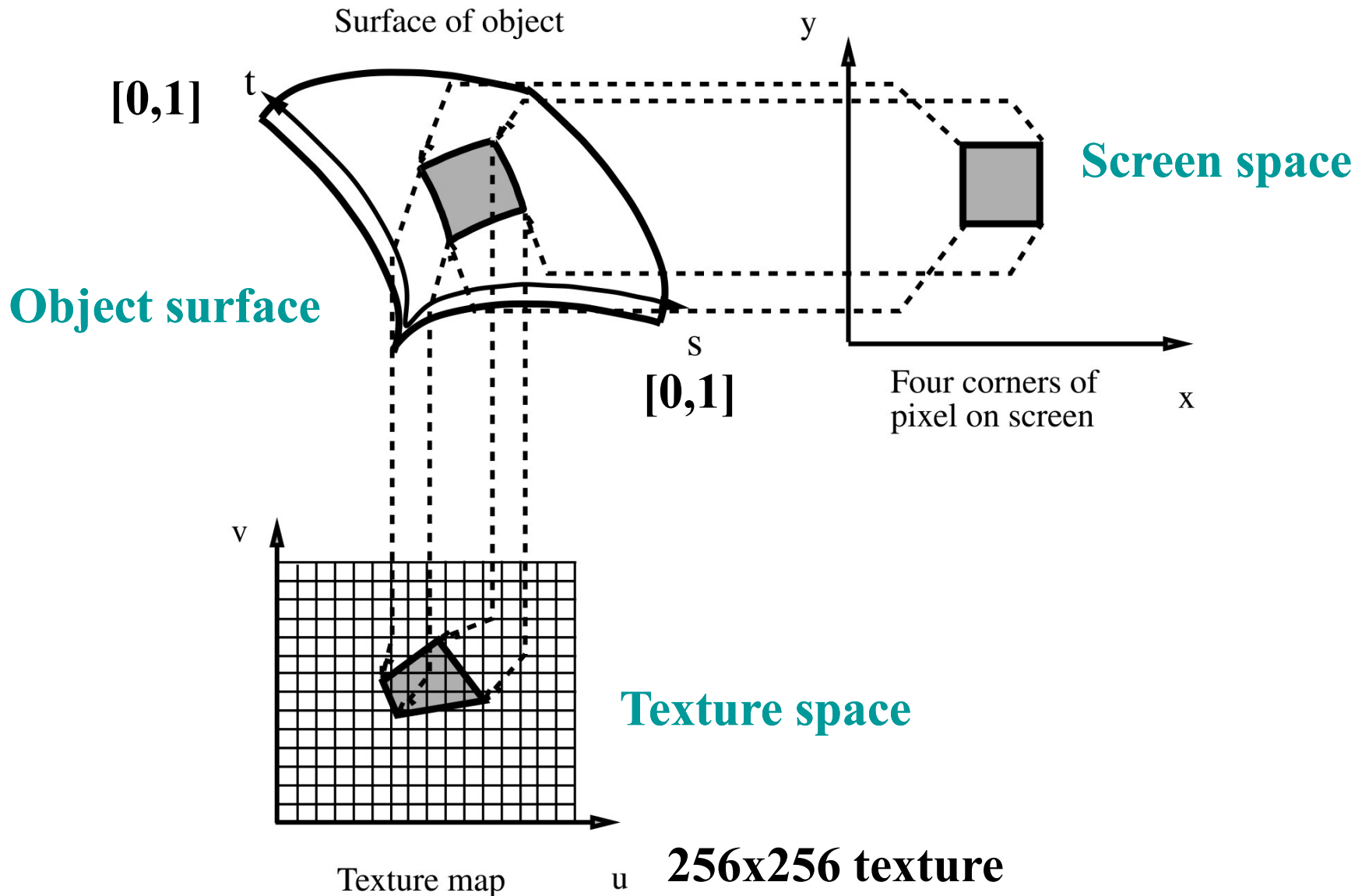
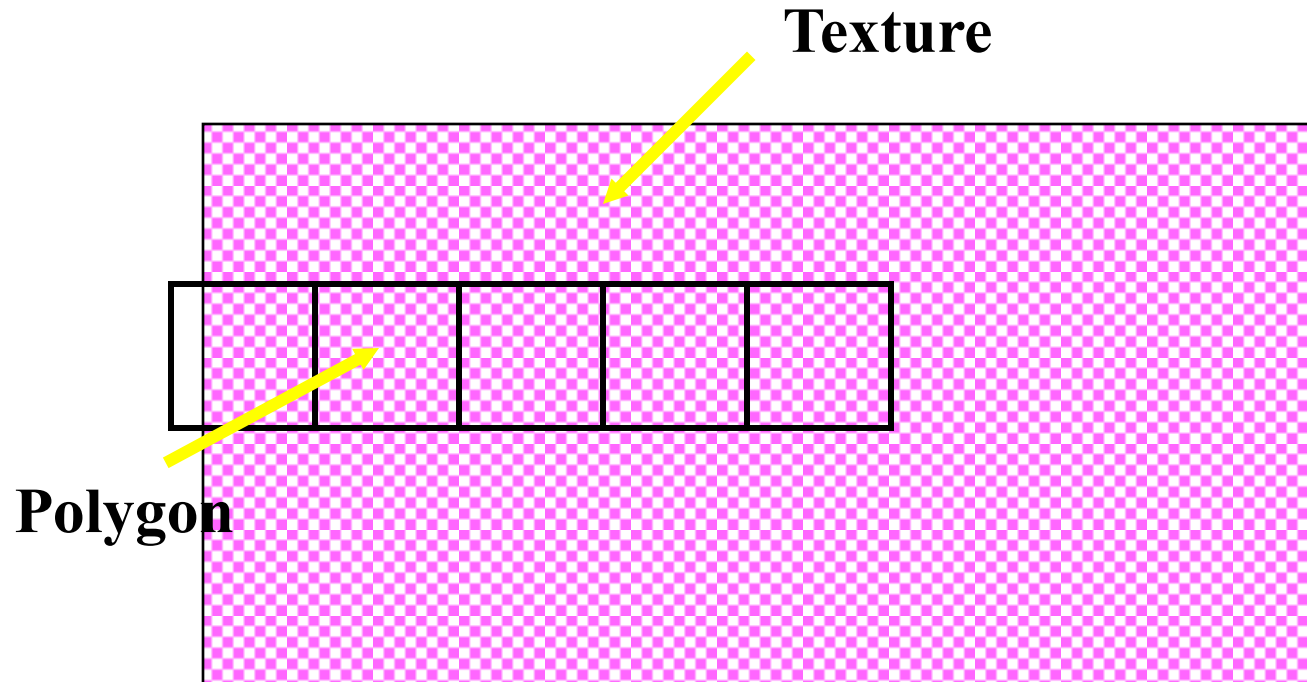


Image Texture:

- ✓ It “glues” an image to a polygon.
- ✓ Size of texture is restricted by graphics accelerators to $2^m \times 2^n$ or 2^m square.
- ✓ If the size of the polygon is much larger than the size of the texture then the OpenGL performs *magnification*
- ✓ If there are much fewer pixels than texels – *minification*
- ✓ Both techniques use bilinear interpolation to assign colors to pixels.

Texture minification:



Uses various “filters” to approximate the color of the pixel: nearest neighbor (to texel closest to the pixel center is selected, bilinear interpolation, etc.)

VR Geometric Modeling



**Tree,
higher resolution model
45,992 polygons.**



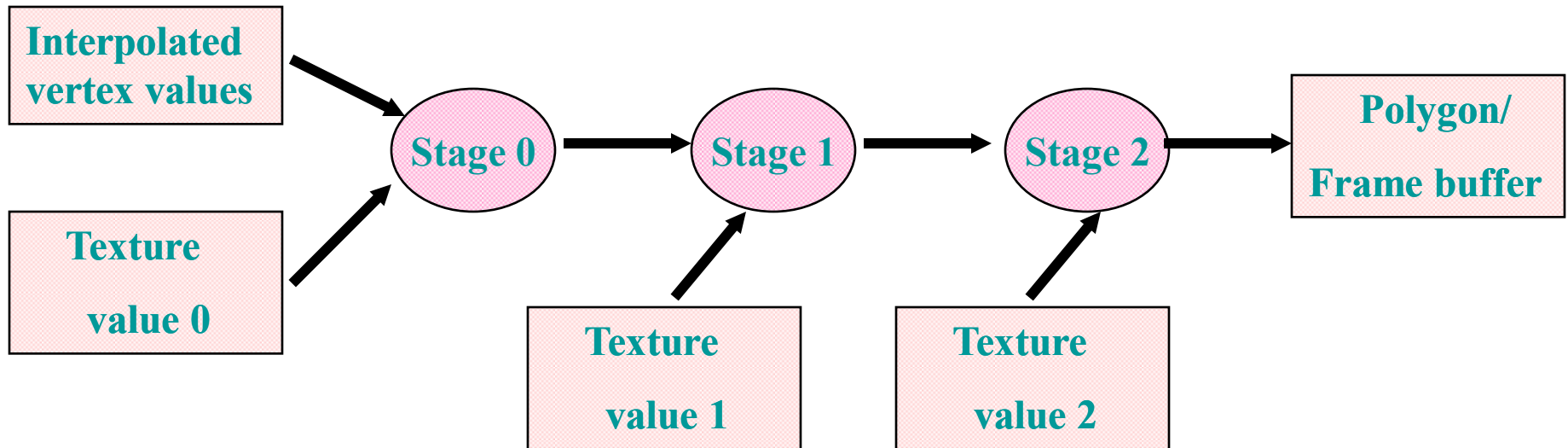
**Tree represented as a texture
1 polygon, 1246x1280 pixels
(www.imagecels.com).**

The GL Part

- Enable usage of textures first
 - glEnable(GL_TEXTURE_2D)
 - glTexEnvf(GL_TEXTURE_ENV, GL ... MODE, GL_REPLACE|MODULATE|DECAL)
- Load and initialize the texture
 - glPixelStore(GL_UNPACK_ALIGNMENT, VAL)
 - Name the texture, glGenTextures(1, name(int))
 - glBindTexture(GL_TEXTURE_2D, name)
 - glTexParameter(GL_TEXTURE_2D, parameter, value)
- Set up the drawing
 - glTexImage2D(Lots of parameters ☹)
- Define coordinates
 - glTexcoord2f()
- OpenGL requires your image be of order 2^n
 - Overcome (kind of) using gluScaleImage()
 - glCopyTexImage2D() may also be of interest

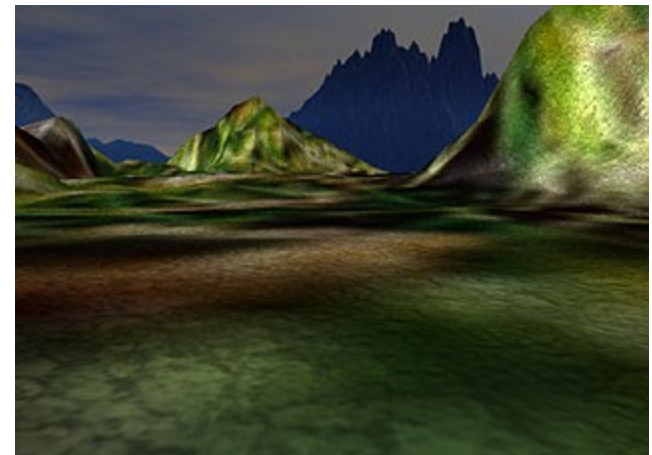
Multi-texturing:

- ✓ Several texels can be overlaid on one pixel;
- ✓ A texture *blending cascade* is made up of a series of texture stages



Class 8-9

Images courtesy,
Steven Jones

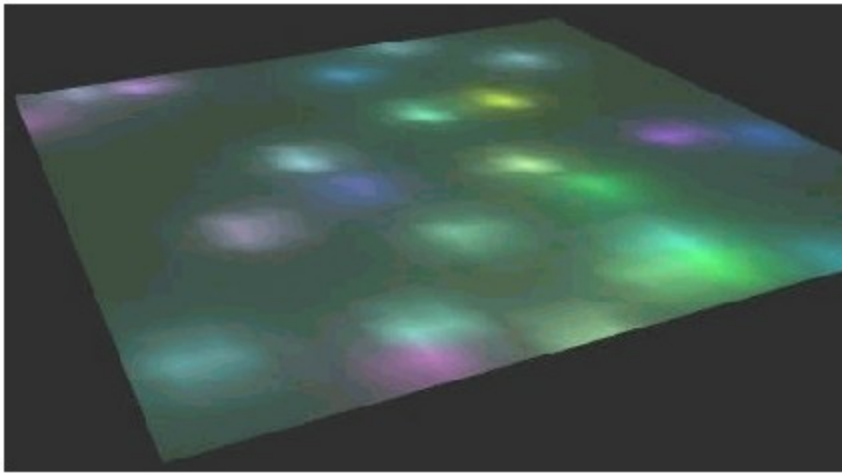


The GL Part for MultiTexturing

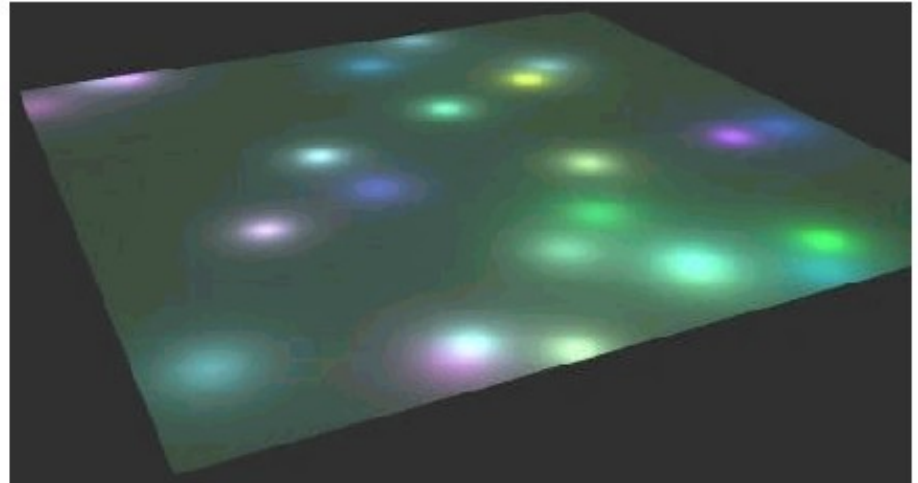
- Needs to use Glex.h
- Needs initialization and use of following two parameters
 - `PFNGLMULTITEXCOORD2FARBPROC` `glMultiTexCoord2fARB` , `glActiveTextureARB`
 - `glActiveTextureAR = (PFNGLACTIVETEXTUREARBPROC)wglGetProcAddress("glActiveTextureARB");`
- `glActiveTextureARB(GL_TEXTURE0_ARB)`
 - Typically upto 32 textures can be used at a time for multitexturing.
- Rest of the procedure is common
- Binding the texture
 - `glMultiTexCoord2fARB(GL_TEXTURE(number)_ARB, 1.0f, 0.0f);`
- Can be used for both
 - Multi-texturing
 - Lightmapping

Multi-texturing for lighting:

- ✓ Several texels can be overlaid on one pixel;
- ✓ One application in more realistic lighting;
- ✓ Polygonal lighting is real-time but requires lots of polygons (triangles) for realistic appearance



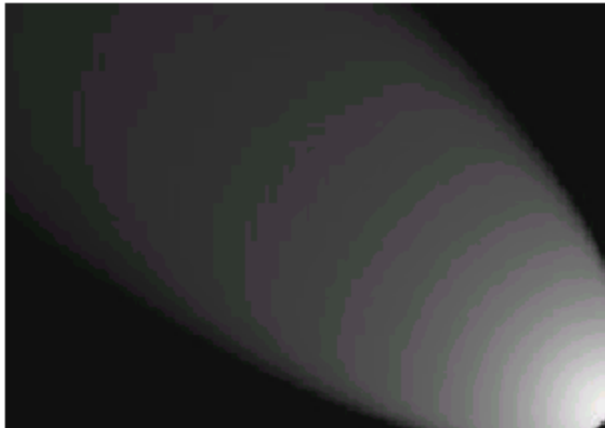
Vertex lighting of low polygon count surface – lights are diffuse – tessellated.



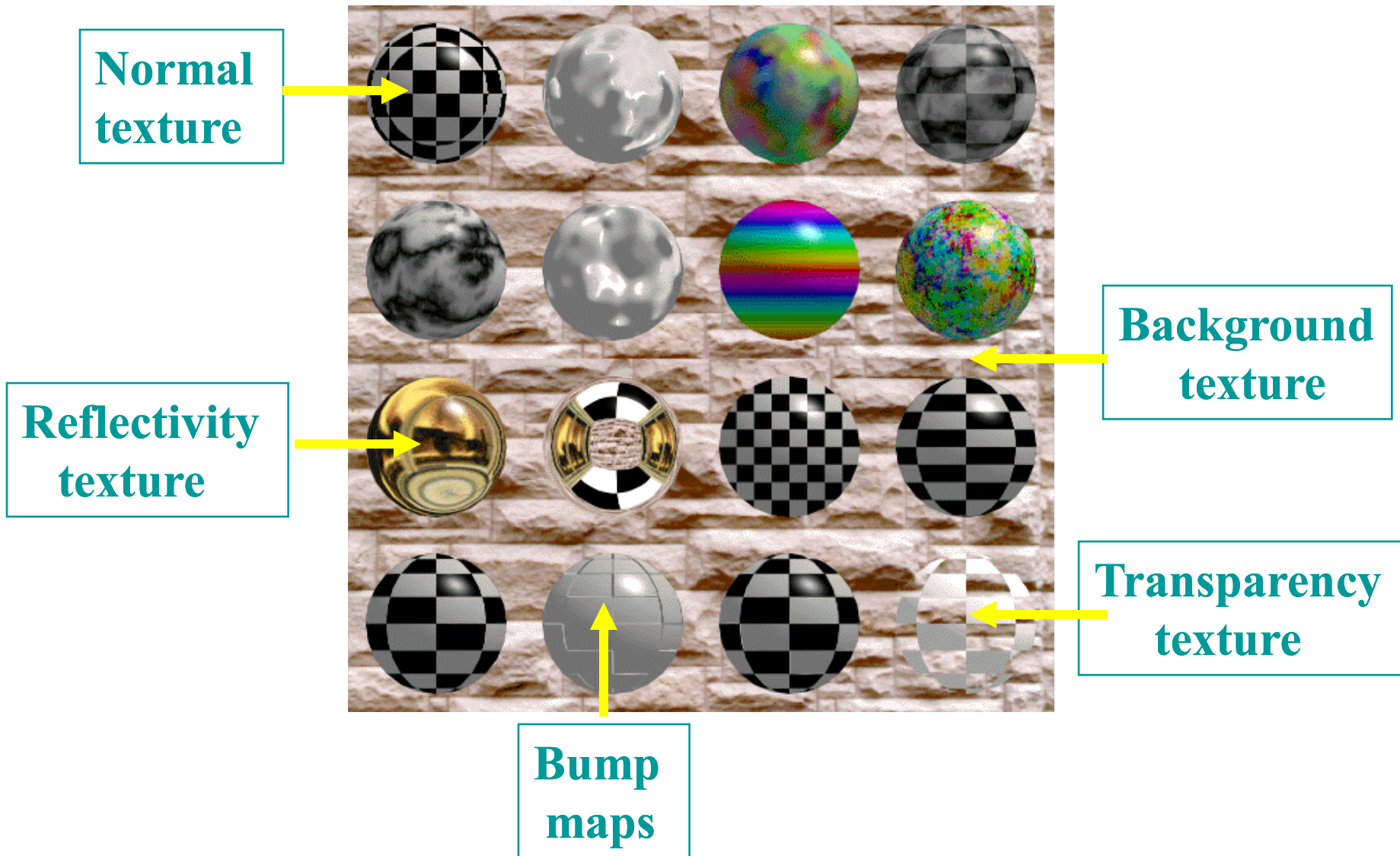
Vertex lighting of high polygon count surface – lights have realistic appearance. High computation load

Multi-texturing (texture blending):

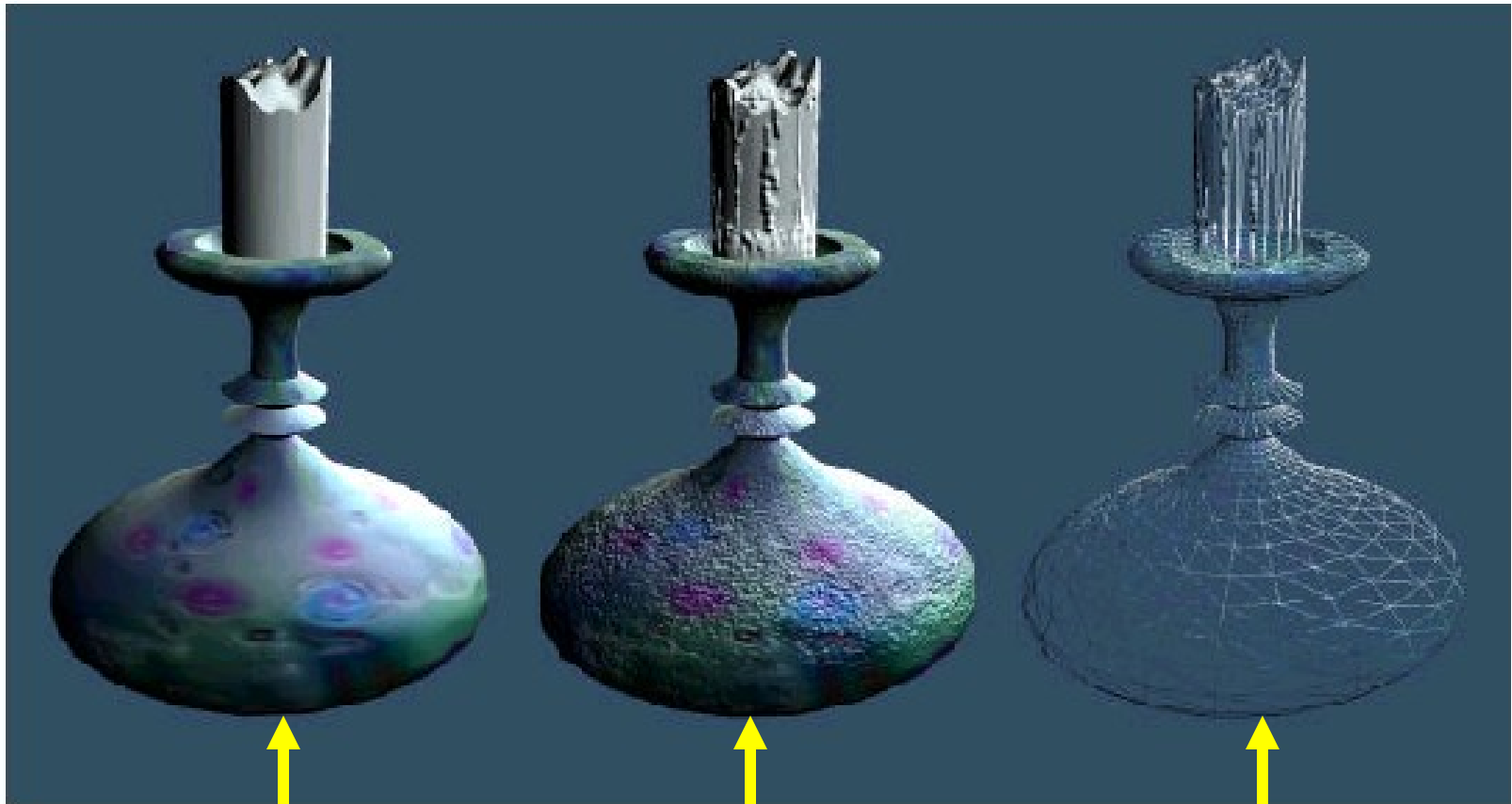
- ✓ Realistic-looking lighting can be done with 2-D textures called “light maps”
- ✓ Light maps are “independent of lighting”



Allow more complex textures



Bump mapping per-pixel shading



**Normal
texture**

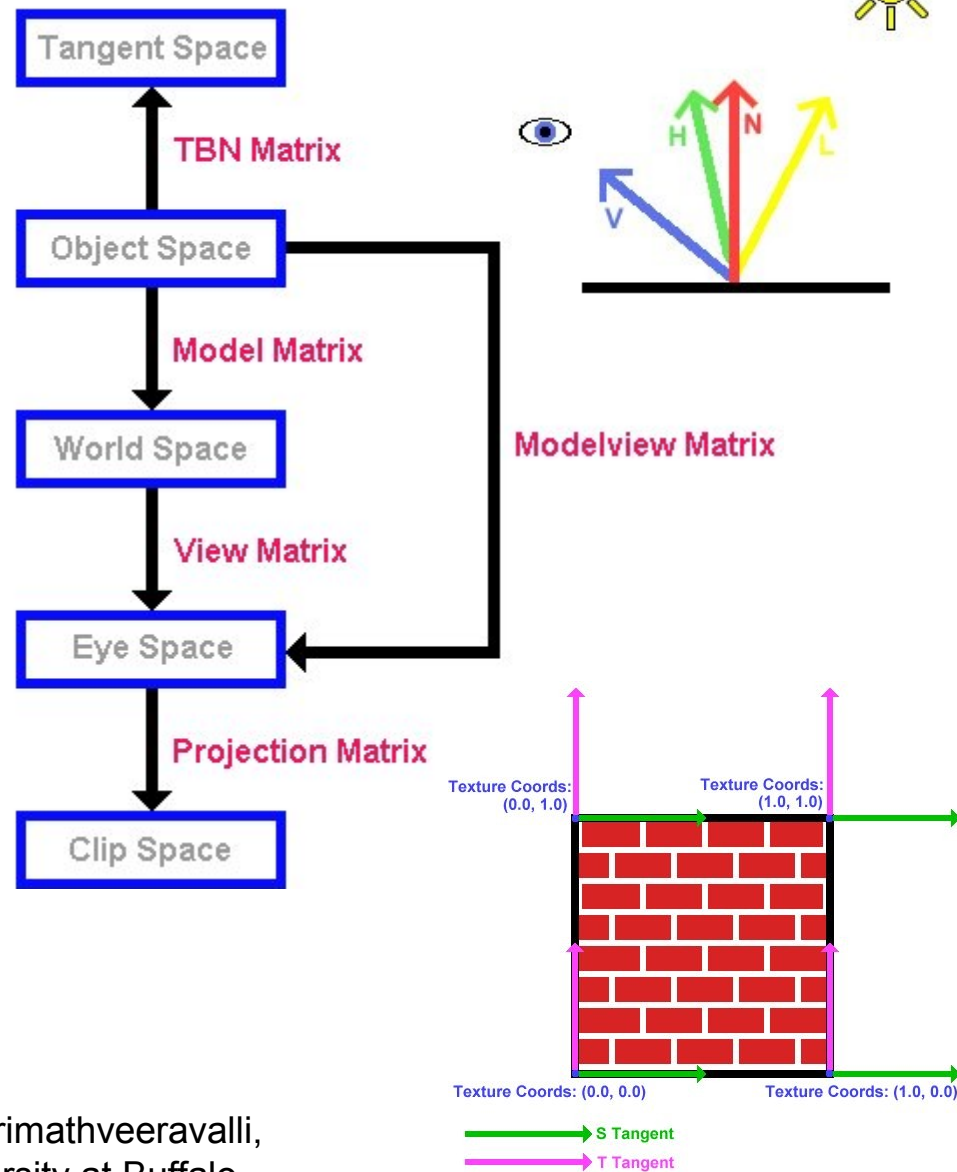
**Multi-
texture**

**Bump
texture**

Bump Mapping – Courtesy Paul's Projects

- For any lighting model you need the vectors in the image to calculate pixel color information
- Texture and tangent space (s,t)
- Need to apply normals per pixel
 - Similar to applying a texture!
- Represent vectors as follows, with condition that all values lie within [0 1]

$$r = (x+1)/2;$$
$$g = (y+1)/2;$$
$$b = (z+1)/2;$$



Bump Mapping – Courtesy Paul's Projects

- Look at a typical bump map generated using previous routine
- Cube maps
 - A 3D representation of textures
- Can be used to store normal information
- Vector passed in is returned normalized
- Why all this trouble?
 - The vector diagram from prev slide, has to be w.r.t. surface!

