

# **Texture Mapping**

**CSCI 4229/5229**

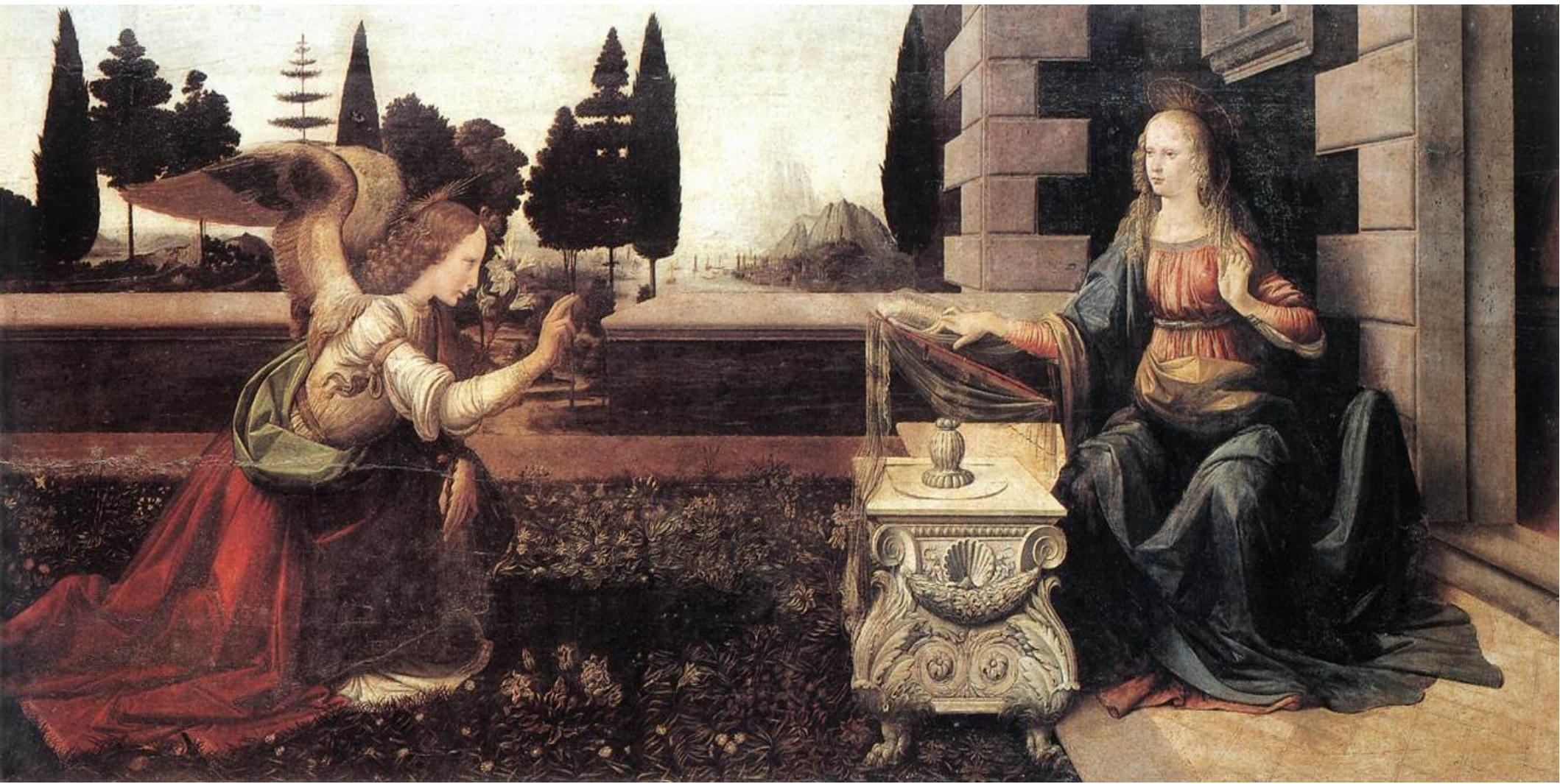
**Computer Graphics**

**Summer 2016**

# What are texture maps?

- Bitmap images used to assign fine texture to displayed surfaces
- Used to make surfaces appear more realistic
- Must move with the surfaces
- Can be stretched or repeated
- Simple in concept, but hardware intensive

# *Annunciation* Leonardo da Vinci (1472)



# OpenGL Texture Types

- Images are draped over polygon surfaces
- 1D, 2D and 3D textures
  - $(s,t,r,q)$  coordinates
  - 2D uses  $(s,t)$ ,  $q$  is the homogeneous  $w$
- 1D, 2D and 3D textures set separately
- 2D textures most commonly used

# OpenGL Texture Calls

- `glGenTextures`
  - Returns unused texture name(s)
- `glBindTexture`
  - Sets the active (current) texture
- `glTexImage*`
  - Copies image to texture memory
- `glTexCoord*`
  - Sets texture coordinates for vertex
- `glTexEnv*`, `glTexParameter*`
  - Control application of textures

# Creating a Texture

- `glGenTextures(1,&texname);`
  - Returns unique texture name
- `glBindTexture(GL_TEXTURE_2D,texname);`
  - First use - allocates memory and makes current
- `glTexImage2D(GL_TEXTURE_2D,0,3,dx,dy,0,GL_RGB,GL_UNSIGNED_BYTE,image);`
  - Copies RGB *image* to texture memory (size *dx xdy*)
  - Image size must be power of two before OpenGL 2

# Setting the Texture Properties

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

- How to magnify texture

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

- How to minify texture

```
glTexEnvf(GL_TEXTURE_2D,  
    GL_TEXTURE_ENV_MODE, GL_MODULATE);
```

- How textures interact with underlying surface

# Applying a Texture Map

```
glBindTexture(GL_TEXTURE_2D, texname);
glBegin(GL_POLYGON);
for (i=0; i<n; i++)
{
    glTexCoord2d(s[i], t[i]);
    glVertex3d(x[i], y[i], z[i]);
}
glEnd();
```



# MIPmaps

- *multum in parvo* (much in little)
- Textures adapted to great distances
  - Level 0=64x64, Level 1=32x32, ... ,  
Level 6=1x1
- Can be generated manually or automatically
  - gluBuild2DMipmaps()
  - gluBuild2DMipmapLevels()

# Multiple Textures

- `glActivateTexture(GL_TEXTUREn);`
  - Call BEFORE `glBindTexture()` etc
- Specify multiple texture coordinates per vertex
  - `glMultiTexCoord2f(GL_TEXTURE0,r0,s0);`
  - `glMultiTexCoord2f(GL_TEXTURE1,r1,s1);`
  - `glMultiTexCoord2f(GL_TEXTURE2,r2,s2);`
  - `glVertex3d(x,y,z);`

# Automatic Texture Coordinates

- `glTexGen*`()
  - Can generate textures automatically for polygons
- `glutSolidTeapot()`
  - Textures coordinates generated
- `gluQuadric` objects
  - `gluQuadricTexture(obj,bool)` controls automatic texture coordinate generation

# Creating a Texture

- `glGenTextures(n, texname[]);`
  - Returns  $n$  unique texture names
- `glBindTexture(GL_TEXTURE_2D, texname);`
  - First use – allocates memory and makes current
  - Subsequent uses just makes it current
  - All operations applies to current texture
    - Current texture is applied to surfaces
    - Current texture is modified by `glTexImage`, etc

```
glTexImage2D(GL_TEXTURE_2D,0,3,dx,dy,  
0,GL_RGB,GL_UNSIGNED_BYTE,image);
```

- `GL_TEXTURE_2D` or `GL_PROXY_TEXTURE_2D`
- Level 0 (or higher for MIPmaps)
- Internal representation 3 (or one of many others)
- Size  $dx \times dy$  [must be  $2^n$  before OpenGL 2.0]
- Border 0 (none) or 1 (pixel width)
- Source image is RGB (or one of many others)
- Source data is unsigned char (or short, etc)
- Image data pointer (can be freed after call)

```
glTexParameter*(GL_TEXTURE_2D,par,val);
```

- **GL\_TEXTURE\_MAG\_FILTER** (magnification)
  - GL\_LINEAR (interpolate)
  - GL\_NEAREST
- **GL\_TEXTURE\_MIN\_FILTER** (minification)
  - GL\_LINEAR (interpolate)
  - GL\_NEAREST
  - GL\_NEAREST\_MIPMAP\_NEAREST
  - GL\_NEAREST\_MIPMAP\_LINEAR
  - GL\_LINEAR\_MIPMAP\_NEAREST
  - GL\_LINEAR\_MIPMAP\_LINEAR

```
glTexParameter*(GL_TEXTURE_2D,par,val);
```

- GL\_TEXTURE\_WRAP\_S (horizontal)
- GL\_TEXTURE\_WRAP\_T (vertical)
- GL\_TEXTURE\_WRAP\_R (depth)
  - GL\_REPEAT (ignore integer part of  $s,t$  )
  - GL\_MIRRORED\_REPEAT (odds backward)
  - GL\_CLAMP - limit to (0,1)
  - GL\_CLAMP\_TO\_EDGE - limit to  $\frac{1}{2}$  pixel in
  - GL\_CLAMP\_TO\_BORDER - limit to  $\frac{1}{2}$  pixel out

```
glTexParameter*(GL_TEXTURE_2D,par,val);
```

- `GL_TEXTURE_BORDER_COLOR`
  - Set border RGBA (4 component float vector)
- `GL_TEXTURE_PRIORITY` (0-1)
- *and many more ...*



# glTexEnvi(GL\_TEXTURE\_2D, val, par)

- GL\_TEXTURE\_ENV\_MODE
  - GL\_MODULATE (multiply)
  - GL\_REPLACE
  - GL\_DECAL (transparent combine)
  - GL\_BLEND
  - GL\_COMBINE
  - GL\_ADD (arithmetic)
- *and many more ...*