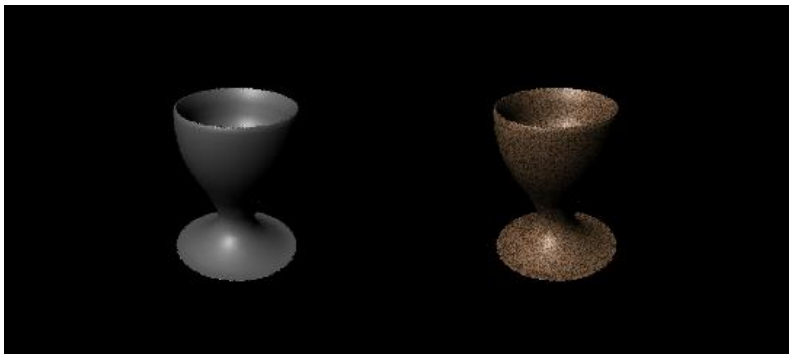
Texture

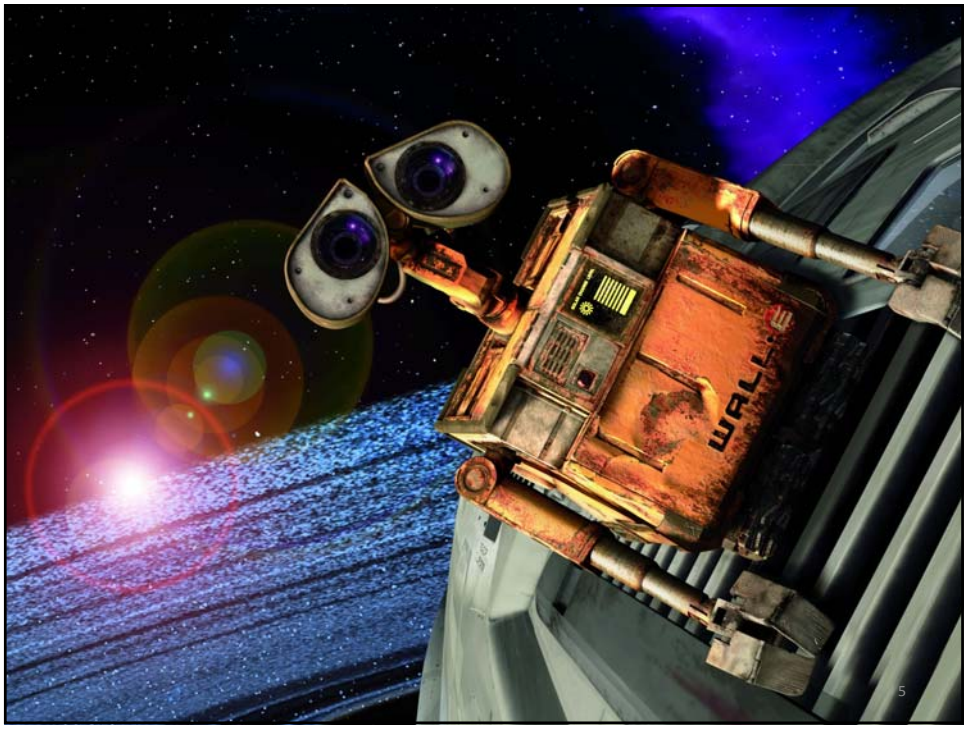
Texture Mapping

- desire for heightened realism



Texture Mapping

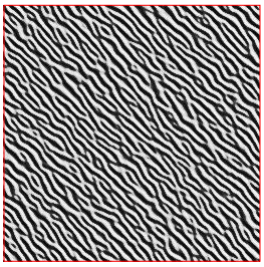
- Means of adding visual detail to a scene without adding geometric detail
- For each point on the surface, there is a corresponding point in a texture image: fetch surface characteristics from the texture
- Characteristics can be color, lighting parameters, surface normal, displacement



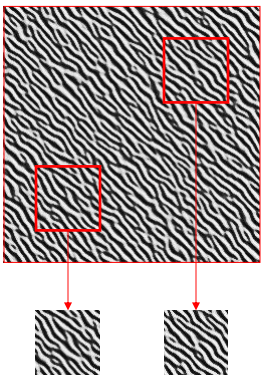
Texture Definitions

- Regular speech: conventional definition – visual and tactile properties of a surface
- Graphics: operational definition – any image used in texture mapping
- Vision: mathematical definition – a signal with the property of 'stationarity' (any part is like another part)

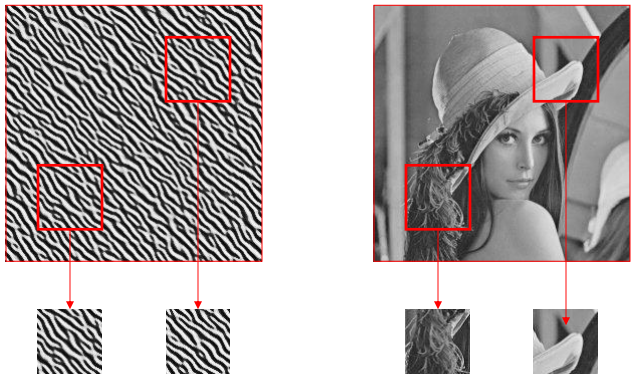
Stationarity



Stationarity



Stationarity

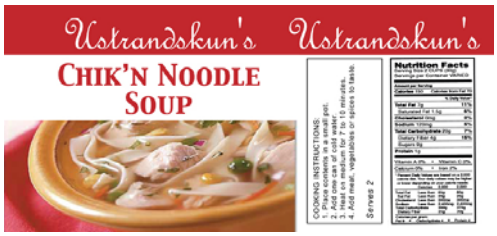


Texture Mapping

- Cover a surface with an image
- Each point has a texture coordinate (u,v)
- Typically, the texture coordinates are given at vertices and interpolated like other parameters

Texture Mapping

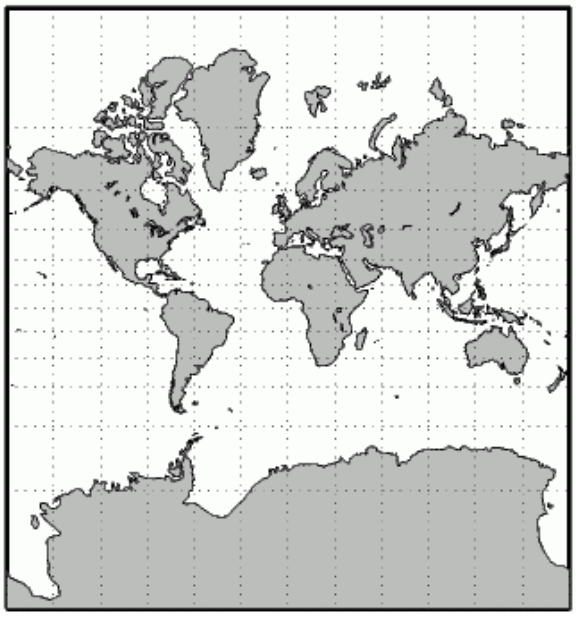
- Simplest object: cylinder (sides only)
  - u coordinate is angle (normalized to 0..1)
  - v coordinate is height (normalized ditto)
- "label on a can of soup"

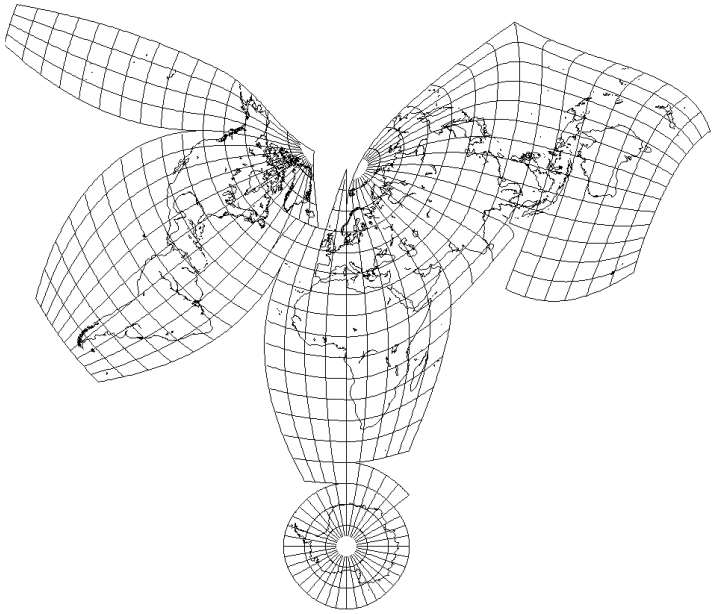


- Not all shapes can be parameterized so easily

Texture Mapping

- Map a 2D surface (in 3D) to a plane
- On the one hand – no problem! The plane is big enough
- On the other hand -- we want to satisfy constraints:
  - no distortion of areas
  - no distortion of angles
  - no seams
- Impossible!





Solutions

- Remove one or more of the constraints:
  - Allow angles to be distorted
  - Allow areas to be distorted
  - Allow seams
- Change the problem
  - Make a mapping from a surface to a surface
  - Make a mapping from a volume to a surface

Relaxing Constraints

- Usual solution – just put up with distortion
- A little bit of distortion is not too visible

- Allow distortion, but use it: the planar texture is pre-distorted

Relaxing Constraints

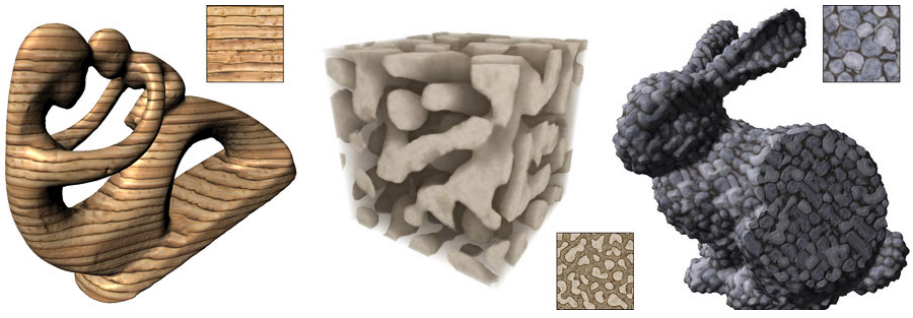
- Permit seams
- Instead of a single "chart", we have a whole "atlas" – the surface is locally flat, so this can work
- Need to hide the seams
  - easy if texture has lots of high-frequency detail
  - align seams with model features (e.g., hairline)
  - put seams in rarely-seen locations

Solid Texture

- Instead of a 2D plane, use a 3D volume of texture
- Mapping is trivial
- Object appears to be carved from textured material

- Drawback: way more texture needed

Solid Texture



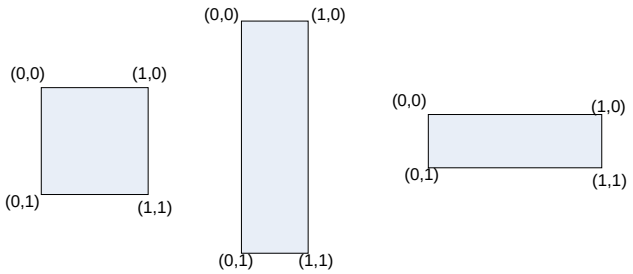
Traditional Texture Mapping

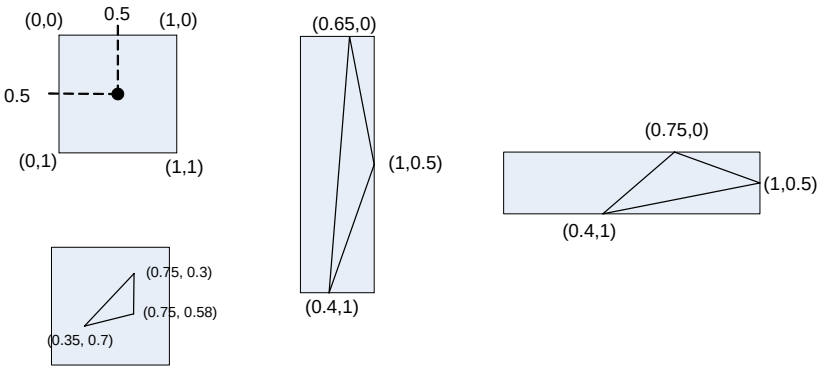
- Specify (u,v) coordinates for vertices
- (u,v) in ({0,1},{0,1}) – canonical values

- Need to specify various characteristics of using the texture
  - what to do with coordinates outside range
  - how to combine texture and lighting
  - mip mapping

Texture coordinate system

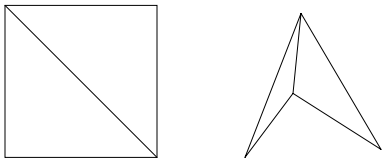
- A normalized coordinate system
  - the range [0,1] x [0,1]
  - Independent of image size or shape





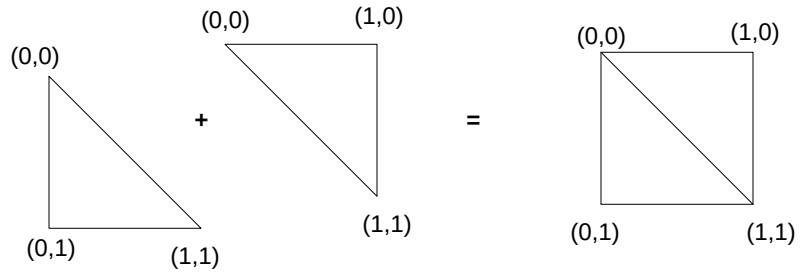
Quads

- A quad is a pair of triangles that share an edge



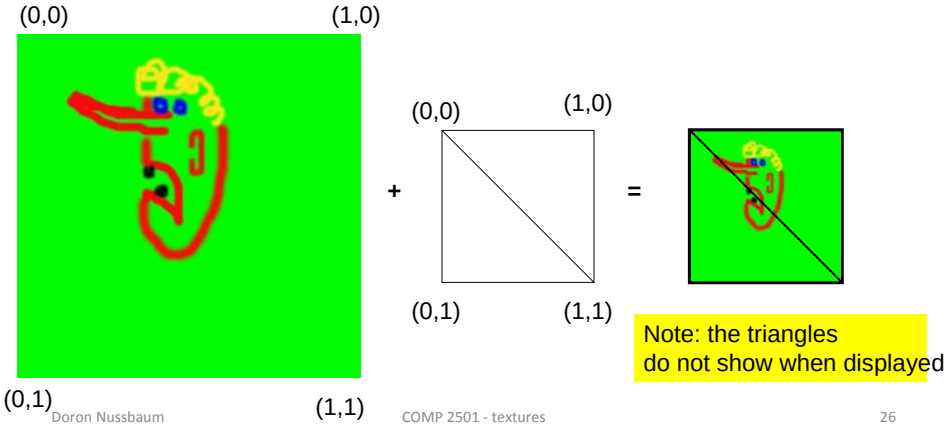
Quads

- A quad is used to render an image



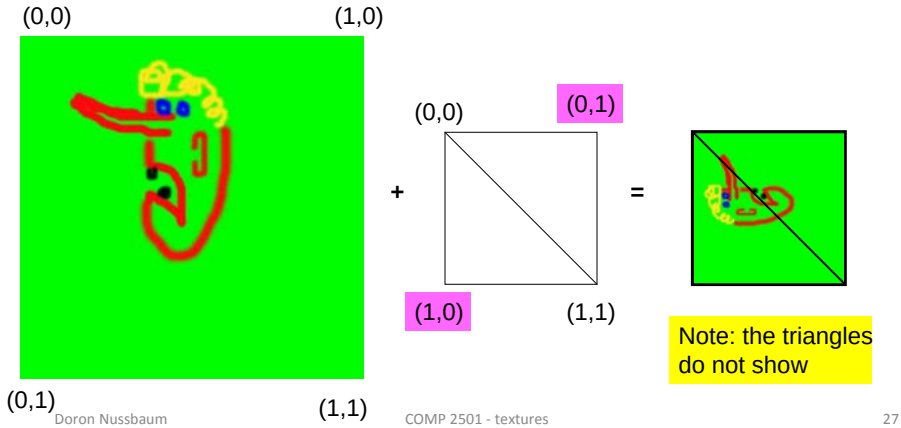
Quads

- A quad is used to render an image



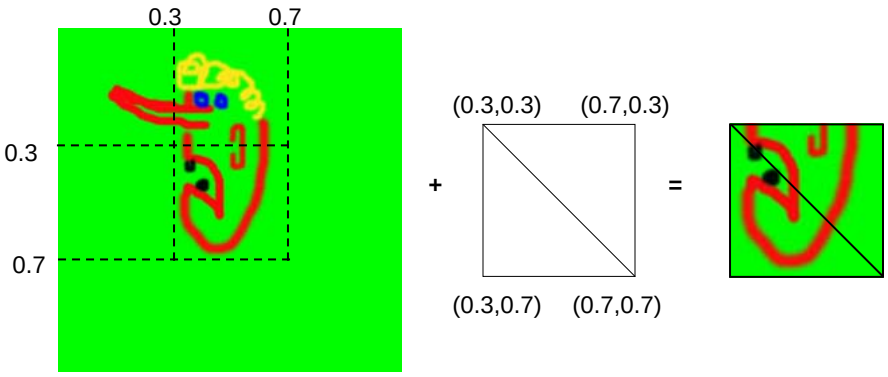
Quads

- A quad is used to render an image



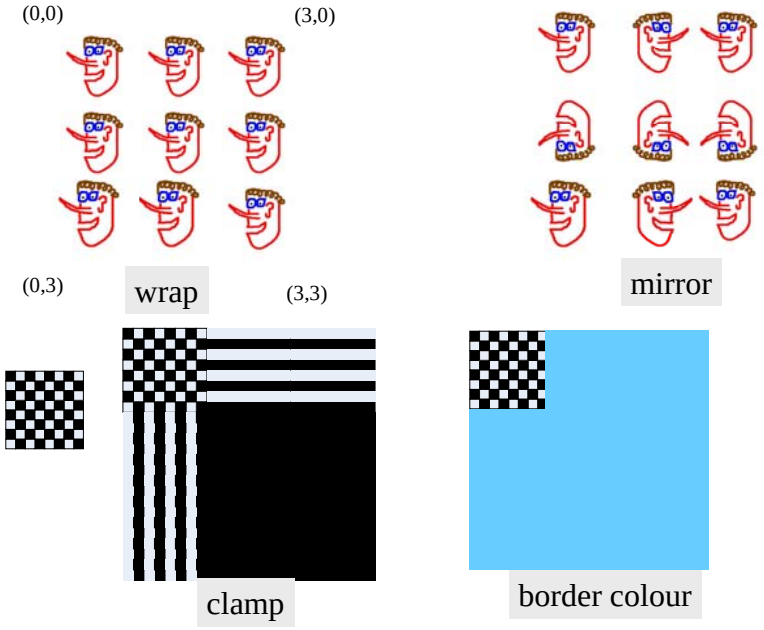
Quads

- Rendering part of the image



Boundary Conditions

- One-dimensional texture for simplicity
- What does T(1.1) evaluate to?
- Three options:
  - clamp: T(t) = T(1), t>1
  - mirror: T(t) = T(1-t), 2>t>1
  - repeat: T(t) = T(floor(t))
- May introduce visible discontinuities



Combining Texture and Lighting

- Texture supplies a color (value in texture map)
- Illumination calculation also supplies a color (calculated by local lighting)
- Combining them:
  - use texture only ("decal")
  - use constant fraction of texture
  - modulate (multiply values together)

Bump Mapping

- Jim Blinn, 1981
- Modify normal direction according to texture
- No extra geometry, illusion of high geometric complexity
- $N = N_{interpolated} * (1-b) + b * N_{texture}$
  - and renormalize

Normal Maps

- No underlying normal to perturb – entire normal comes from texture
- No 3D geometry needed! Can give illusion of geometry on sprites (just use fake normal in lighting calculation)

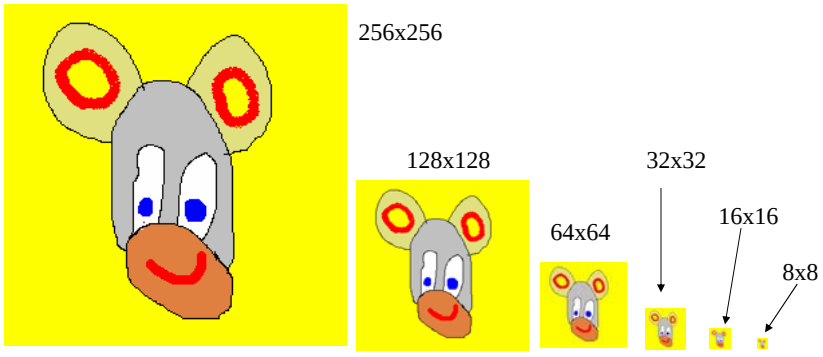
Magnification and Minification

- Texture rarely appears onscreen at the stored resolution
- If it appears smaller: minification filtering
- If it appears larger: magnification filtering

- Filtering options:
  - nearest neighbour
  - linear interpolation
  - cubic interpolation (smoother, more expensive)

MIPmaps

- A mipmap is a chain of bitmaps (surfaces) where each bitmaps is ¼ of the previous image (babushka dolls)
  - Chain stops when images are of size 1x1 or required level was obtained
  - Filters are point and linear



the TextureSampler

- In HLSL, reading a value from a texture is done with the TextureSampler
  - auxiliary function defined in shader
  - specifies what interpolation to use
  - given coordinates, returns texture value (color)
- Shader interprets texture
  - used for color, normal (bump mapping), position (displacement mapping) , ...

Working With Textures C++ code

- Create a texture

```
HRESULT D3DXCreateTextureFromFile (
LPDIRECT3DDEVICE9 pDevice,
LPCTSTR pSrcFile,
LPDIRECT3DTEXTURE9 * ppTexture
 );
```

- Assign it to a handle

```
IDirect3DTtxture9 *mMyTexture = NULL;

D3DXCreateTextureFromFile (d3dDev, "earth.jpg", &mMyTexture);
…

// assign it to a the texture in the effect file
effect-> SetTexture(mhTexture, mMyTexture);
```

```
// in texture file

uniform extern texture gTexture;
```

Sampling filters

- image triangle > screen triangle
  - Image must be reduced - Minified
- image triangle < screen triangle
  - Image must be enlarged - Magnified

- Direct3D provides three types of sampling operations
  - Nearest neighbour filter
  - Linear filter
  - Anisotropic filter

- Instructing Direct3D
  - d3dDevice->SetSampleState(stage, type, value);

Nearest Point Filter

- Operation
  - For each pixel on the screen
    - Map it back to the image (find the texel)
    - Use the value of the nearest texel
- Properties
  - Poorest quality
  - Fastest and simplest
  - The default filter

Nearest Point Filter

- Operation
  - For each pixel on the screen
    - Map it back to the image (find the texel)
    - Use the value of the nearest texel
- Properties
  - Poorest quality
  - Fastest and simplest
  - The default filter

- Commands

Enlarging the image

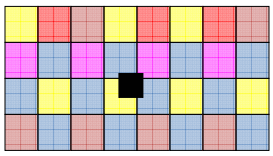
```
d3dDevice->SetSampleState(0, D3DSAMP_MAGFILTER, D3DTEXF_POINT);
```

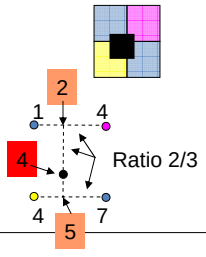
Reducing the image

```
d3dDevice->SetSampleState(0, D3DSAMP_MINFILTER, D3DTEXF_POINT);
```


Linear Filter

- Operation
  - For each pixel on the screen
    - Map the pixel back to the image
    - Use bilinear interpolation to compute a value
    - Use the computed value
- Properties
  - Better quality
  - Slower than nearest-point
  - Most graphic cards are optimized to using it





Linear Filter

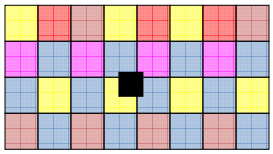
- Operation
  - For each pixel on the screen
    - Map the pixel back to the image
    - Use bilinear interpolation to compute a value
    - Use the computed value
- Properties
  - Better quality
  - Slower than nearest-point
  - Most graphic cards are optimized to using it
- Commands

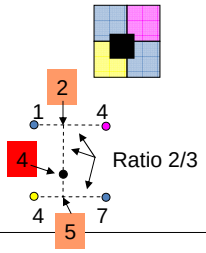
Enlarging the image

```
d3dDevice->SetSampleState(0, D3DSAMP_MAGFILTER, D3DTEXF_LINEAR);
```

Reducing the image

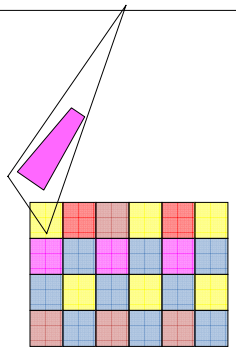
```
d3dDevice->SetSampleState(0, D3DSAMP_MINFILTER, D3DTEXF_LINEAR);
```





Anisotropic filter

- Operation
  - For each pixel on the screen
    - Map the pixel back to the image
    - Measure the distortion as a result of a 3D
    - Compute the new value using the distortion
    - Use the computed value
- Properties
  - Better quality
  - Time consuming
  - For 3D
  - Most graphic cards are optimized to using it



Anisotropic filter

- Commands

Enlarging the image

```
d3dDevice->SetSamplerState(0, D3DSAMP_MAGFILTER, D3DTEXF_ANISOTROPIC);
```

Reducing the image

```
d3dDevice->SetSamplerState(0, D3DSAMP_MINFILTER, D3DTEXF_ANISOTROPIC);
```

Also requires the setting of a maximum level

```
d3dDevice->SetSampleState(0, D3DSAMP_MAXANISOTROPY, 4);
```


Mipmaps

- DirectX uses the mipmaps mechanism to increase the accuracy of texel sampling
- A mipmap chain is created automatically when the texture is created using d3dDevice->CreateTextureFromFile();
- The mipmap chain is a complete chain
  - Smallest image is of size 1x1

Mipmap filters

- A mipmap filter is used to determine which mipmap level is to be used to compute the pixel value from the texels.
- Command
  - SetSampleState(stage, D3DSAMP_MIPFILTER, filter)
  - The filter can be
    - None – do not use mipmap (D3DTEXF_NONE)
    - Point
    - Linear

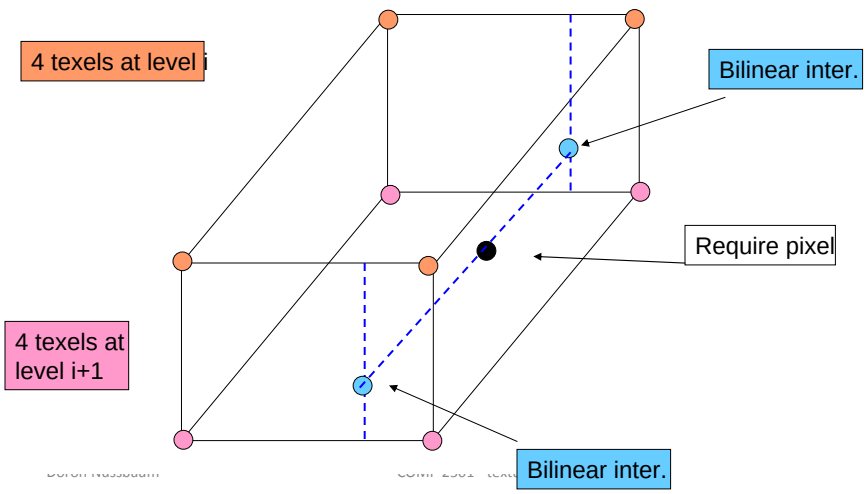
Mipmap – point filter

- Operates in conjunction with MIN/MAG filters
- Operation
  - Chooses the level that is closest to the triangle size
  - Use the bitmap of that level and operate the corresponding MIN/MAG filters (e.g., bilinear)

Mipmap – Linear filter

- Performs trilinear interpolation of the texels.
- Operation
  - Chooses the two levels that are closest to the triangle size (the triangle is in between)
  - Selects 4 pixels in each level
  - Compute a pixel value from the 8 texels

Mipmap – Linear filter



Texture behaviour

- What happens when using coordinates outside the range [0,1]?
  - For example using coordinates (u,v) = (0,3)
- Is it allowed?
- Yes
- There are four types of behaviours
  - Wrap
  - Mirror
  - Clamp
  - Colour

Working With Textures Effect File

- Samplers
  - Define how to use the
  - Filters
  - Address modes

```
uniform extern texture gTexture;

Sampler TexS = sampler_state {
    texture <gTexture>;                  // the texture

    // "access instructions"

};
```

```
// in texture file
uniform extern texture gTexture;

Sampler TexS = sampler_state {
    texture <gTexture>;                  // the texture
    MinFilter = LINEAR;
    MaxFilter = LINEAR;

    MipFilter = POINT;

    AddressU = WRAP
    AddressV = MIRROR;
};
```

Working With Textures Effect File

- Access the texture data

float4 tex2D(sampler s, float2 t)
s - input sampler
t – texture coordinate

float4 tex3D(sampler s, float3 t)
s - input sampler
t – texture coordinate

```
// in texture file
uniform extern texture gTexture;

Sampler TexS = sampler_state {
   texture <gTexture>;          // the texture
   MinFilter = LINEAR;
   MaxFilter = LINEAR;
   AddressU = WRAP
   AddressV = MIRROR;
};


float4 textureLightPS(float4 col COLOR0, float2 texCoord: TEXCOORD0)
{
    float4 texColor = tex2D(TexS, texCoord).rgb;  // extract colour of pixel from texture

    texColor.a = 1.0;                    // manipulate color

    return(texColor.a );                        // return the color
}
```