Texture Mapping

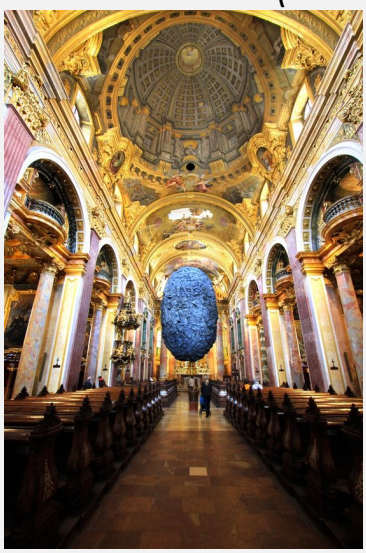
1

Texture Mapping

- A way of adding surface details
- Two ways can achieve the goal:
  - Model the surface with more polygons
    - » Slows down rendering speed
    - » Hard to model fine features
  - Map a texture to the surface
    - » This lecture
    - » Image complexity does not affect complexity of processing
- Efficiently supported in hardware

2

TROMPE L'OEIL ("DECEIVE THE EYE")



Jesuit Church, Vienna, Austria

- Windows and columns in the dome are painted, not a real 3D object
- Similar idea with texture mapping:

Rather than modeling the intricate 3D geometry, replace it with an image !

3

3

Rather than modeling the intricate 3D geometry, replace it with an image

4

MAP TEXTURES TO SURFACES

an image

texture map

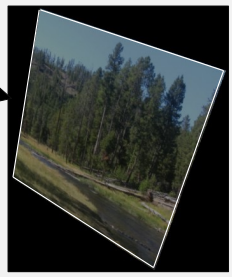


image mapped
to a 3D polygon

The polygon can have arbitrary size, shape and 3D position

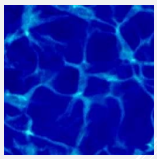
5

5

THE TEXTURE

- Texture is a bitmap image
  - Can use an image library to load image into memory
  - Or can create images yourself within the program

- 2D array:
  unsigned char texture[height][width][4]
- Or unrolled into 1D array:
  unsigned char texture[4*height*width]



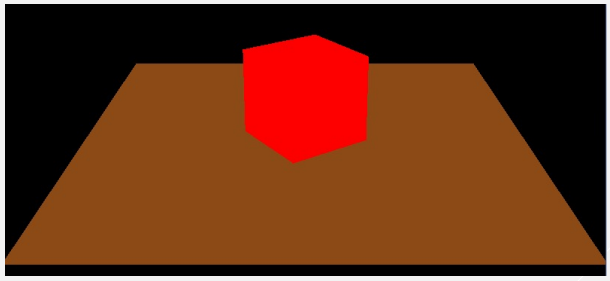
- Pixels of the texture are called *texels*
- Texel coordinates (s,t) scaled to [0,1] range

6

6

OPENGL EXAMPLE

- 拿掉了原先的線條，改採用純色的三角面組成這Block，卻沒使該物體顯得逼真



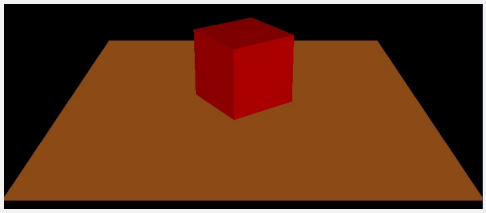
Superbible 5th edition sample project

7

OPENGL EXAMPLE

- Shading(明暗度)

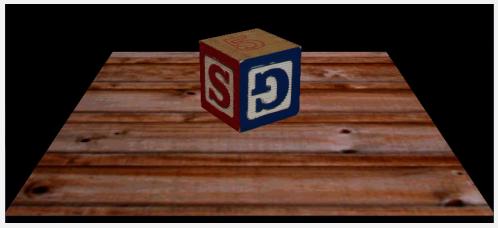
利用明暗程度的差異(打光技巧)來使原先的Block不同的面有了色差



8

OPENGL EXAMPLE

- Texture Mapping(貼圖投影)
  - 將一張圖案投影到三角形或多邊形上, 真實性提升



9

OPENGL EXAMPLE

- Blending(混合)
  - 此圖的反射效果; 預先複製一個顛倒的Block，加上半透明的效果出來，再與木版做混合

10

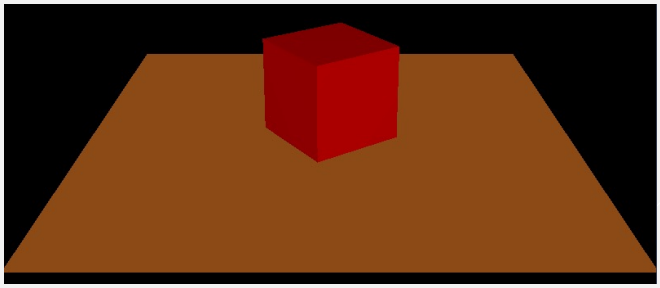
Texture Mapping(貼圖投影)


floor.tga


Block4.tga


Block5.tga

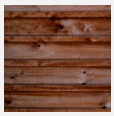

Block6.tga



11

```
GLuint textures[4];

void SetupRC()
{
    GLbyte *pBytes;
    GLint nWidth, nHeight, nComponents;
    GLenum format;

    // Black background
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f );

    glTexEnvi(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE, GL_MODULATE);
    glGenTextures(4, textures);

    // Load the texture objects
    pBytes = gltLoadTGA("floor.tga", &nWidth, &nHeight,
                                          &nComponents, &format);
    glBindTexture(GL_TEXTURE_2D, textures[0]);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexImage2D(GL_TEXTURE_2D,0,nComponents,nWidth, nHeight, 0,
                 format, GL_UNSIGNED_BYTE, pBytes);
    free(pBytes);

    //setup other images… (next page)
}
```

iComponent:
RGB8/RGBA8/GL_LUMINANCE8



12

```
pBytes = gltLoadTGA("Block4.tga", &nWidth, &nHeight, &nComponents, &format);
    glBindTexture(GL_TEXTURE_2D, textures[1]);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexImage2D(GL_TEXTURE_2D,0,nComponents,nWidth, nHeight, 0,
                 format, GL_UNSIGNED_BYTE, pBytes);
    free(pBytes);
```

```
pBytes = gltLoadTGA("Block5.tga", &nWidth, &nHeight, &nComponents, &format);
    glBindTexture(GL_TEXTURE_2D, textures[2]);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexImage2D(GL_TEXTURE_2D,0,nComponents,nWidth, nHeight, 0,
                 format, GL_UNSIGNED_BYTE, pBytes);
    free(pBytes);
```

```
pBytes = gltLoadTGA("Block6.tga", &nWidth, &nHeight, &nComponents, &format);
    glBindTexture(GL_TEXTURE_2D, textures[3]);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexImage2D(GL_TEXTURE_2D,0,nComponents,nWidth, nHeight, 0,
                 format, GL_UNSIGNED_BYTE, pBytes);
    free(pBytes);
```

13

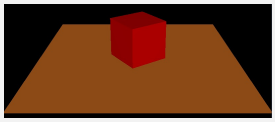
Texture Mapping(貼圖投影)

floor.tga
#0

Block4.tga
#1

Block5.tga
#2

Block6.tga
#3



14

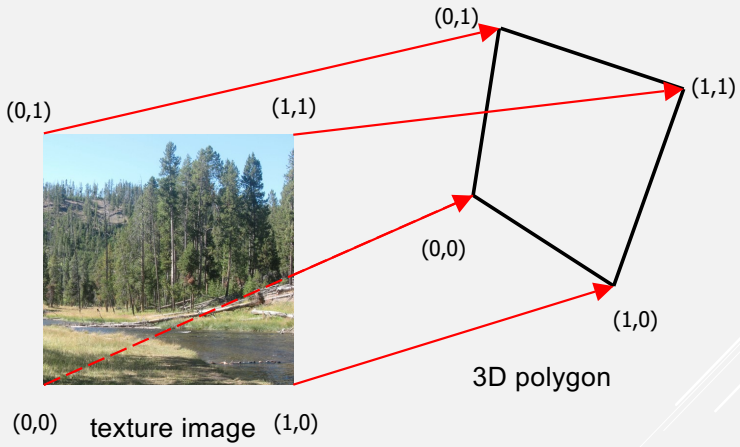
Read image using OpenCV example

```
cv::Mat image = cv::imread("textures/trashbin.png");
  //cv::Mat flipped;
  //cv::flip(image, flipped, 0);
  //image = flipped;
  if(image.empty()){
      std::cout << "image empty" << std::endl;
  }else{
      cv::flip(image, image, 0);
      glGenTextures(1, &textureTrash);
      glBindTexture(GL_TEXTURE_2D, textureTrash);

      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

        // Set texture clamping method
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);


      glTexImage2D(GL_TEXTURE_2D,     // Type of texture
                   0,                   // Pyramid level (for mip-mapping) - 0 is t
                   GL_RGB,              // Internal colour format to convert to
                   image.cols,            // Image width  i.e. 640 for Kinect in st
                   image.rows,            // Image height i.e. 480 for Kinect in st
                   0,                   // Border width in pixels (can either be 1
                   GL_BGR, // Input image format (i.e. GL_RGB, GL_RGBA, GL_BGR et
                   GL_UNSIGNED_BYTE,  // Image data type
                   image.ptr());        // The actual image data itself

      glGenerateMipmap(GL_TEXTURE_2D);
  }
```

https://stackoverflow.com/questions/16809833/opencv-image-loading-for-opengl-texture

15

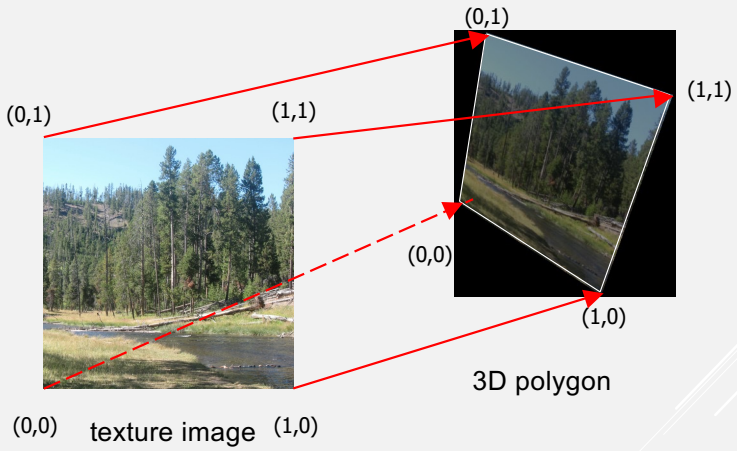
TEXTURE MAP



17

17

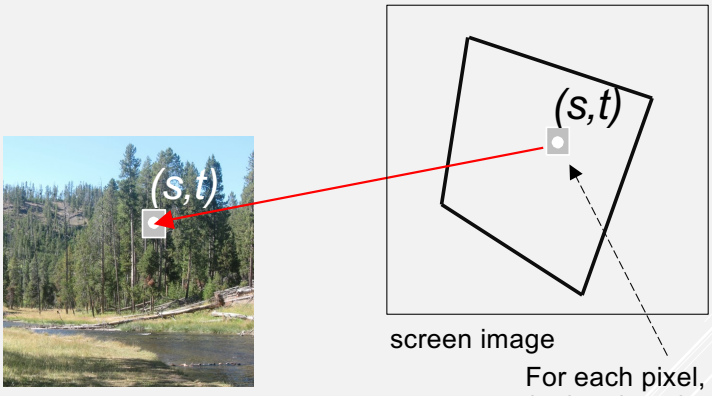
TEXTURE MAP



18

18

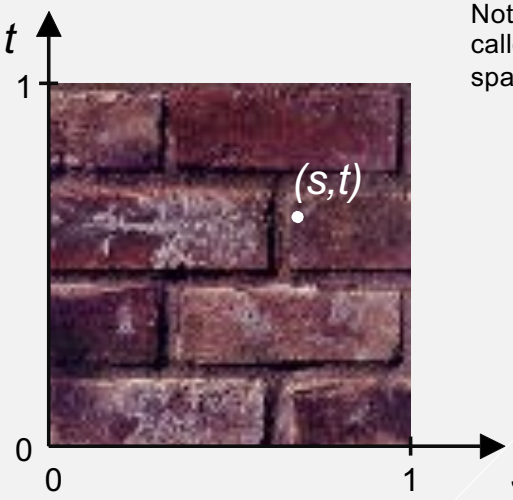
INVERSE TEXTURE MAP



19

19

THE “ST” COORDINATE SYSTEM

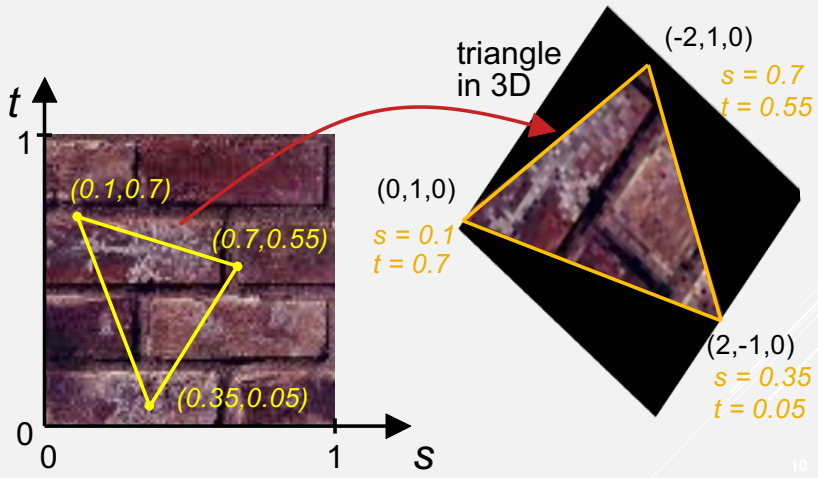


Note: also called “uv” space

20

20

TEXTURE MAPPING: KEY SLIDE

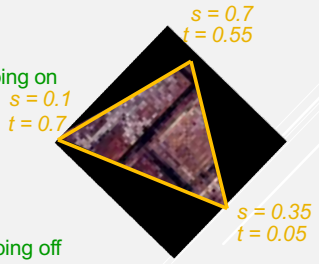


21

Specifying texture coordinates in OpenGL

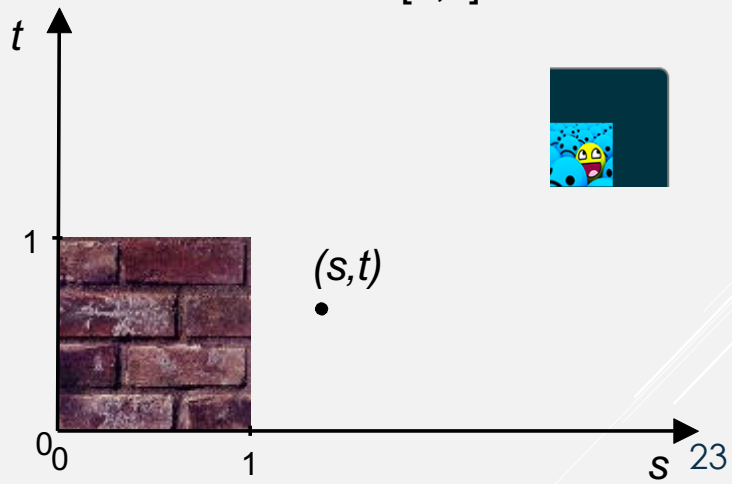
- Use glTexCoord2f(s,t)
- State machine: Texture coordinates remain valid until you change them
- Example (from previous slide) :

```
glEnable(GL_TEXTURE_2D); // turn texture mapping on
glBegin(GL_TRIANGLES);
  glTexCoord2f(0.35,0.05); glVertex3f(2.0,-1.0,0.0);
  glTexCoord2f(0.7,0.55); glVertex3f(-2.0,1.0,0.0);
  glTexCoord2f(0.1,0.7); glVertex3f(0.0,1.0,0.0);
glEnd();
glDisable(GL_TEXTURE_2D); // turn texture mapping off
```



22

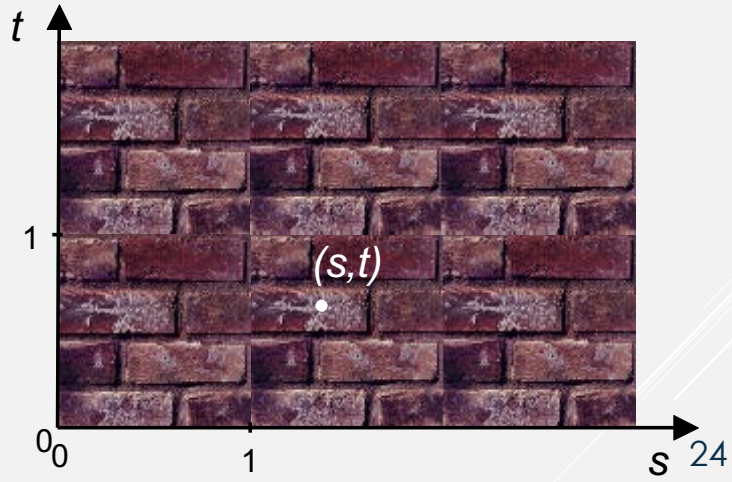
What if texture coordinates are outside of [0,1] ?



23

Solution 1: Repeat texture

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT)
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT)



24

```
pBytes = gltLoadTGA("Block4.tga", &nWidth, &nHeight, &nComponents, &format);
    glBindTexture(GL_TEXTURE_2D, textures[1]);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexImage2D(GL_TEXTURE_2D,0,nComponents,nWidth, nHeight, 0,
                 format, GL_UNSIGNED_BYTE, pBytes);
    free(pBytes);
```

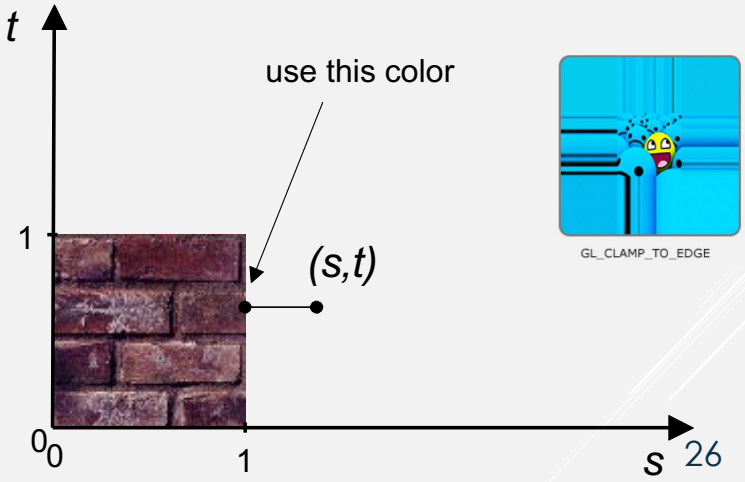
```
pBytes = gltLoadTGA("Block5.tga", &nWidth, &nHeight, &nComponents, &format);
    glBindTexture(GL_TEXTURE_2D, textures[2]);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexImage2D(GL_TEXTURE_2D,0,nComponents,nWidth, nHeight, 0,
                 format, GL_UNSIGNED_BYTE, pBytes);
    free(pBytes);
```

```
pBytes = gltLoadTGA("Block6.tga", &nWidth, &nHeight, &nComponents, &format);
    glBindTexture(GL_TEXTURE_2D, textures[3]);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexImage2D(GL_TEXTURE_2D,0,nComponents,nWidth, nHeight, 0,
                 format, GL_UNSIGNED_BYTE, pBytes);
    free(pBytes);
```

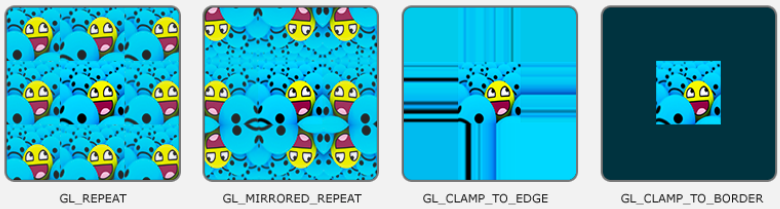
25

Solution 2: Clamp to [0,1]

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP)
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP)

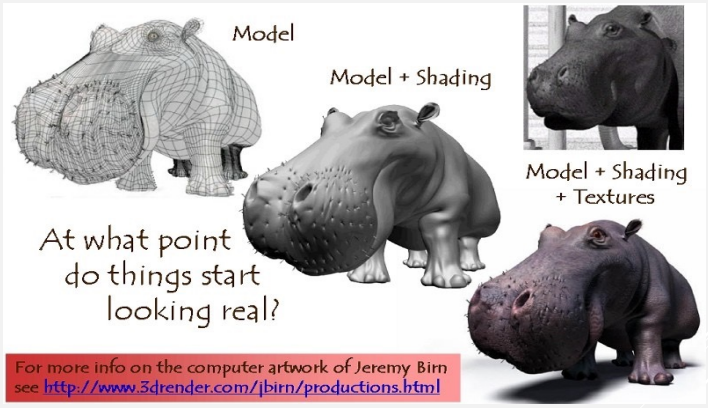


26



27

Combining texture mapping and shading

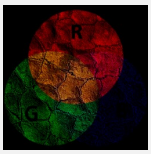


28

28

Combining texture mapping and shading

- Final pixel color = a combination of texture color and color under standard OpenGL Phong lighting



- GL_MODULATE:
  multiply texture and Phong lighting color
- GL_BLEND:
  linear combination of texture and Phong lighting color
- GL_REPLACE:
  use texture color only (ignore Phong lighting)

- Example:
  glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, **GL_REPLACE**);

29

29

```
glTexEnvi(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE, GL_MODULATE);
```

```
void glTexEnvf( GLenum target,
                GLenum pname,
                GLfloat param );
```

target

Specifies a texture environment. May be GL_TEXTURE_ENV, GL_TEXTURE_FILTER_CONTROL or GL_POINT_SPRITE.

pname

Specifies the symbolic name of a single-valued texture environment parameter. May be either GL_TEXTURE_ENV_MODE, GL_TEXTURE_LOD_BIAS, GL_COMBINE_RGB, GL_COMBINE_ALPHA, GL_SRC0_RGB, GL_SRC1_RGB, GL_SRC2_RGB, GL_SRC0_ALPHA, GL_SRC1_ALPHA, GL_SRC2_ALPHA, GL_OPERAND0_RGB, GL_OPERAND1_RGB, GL_OPERAND2_RGB, GL_OPERAND0_ALPHA, GL_OPERAND1_ALPHA, GL_OPERAND2_ALPHA, GL_RGB_SCALE, GL_ALPHA_SCALE, or GL_COORD_REPLACE.

param

Specifies a single symbolic constant, one of GL_ADD, GL_ADD_SIGNED, GL_INTERPOLATE, GL_MODULATE, GL_DECAL, GL_BLEND, GL_REPLACE, GL_SUBTRACT, GL_COMBINE, GL_TEXTURE, GL_CONSTANT, GL_PRIMARY_COLOR, GL_PREVIOUS, GL_SRC_COLOR, GL_ONE_MINUS_SRC_COLOR, GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA, a single boolean value for the point sprite texture coordinate replacement, a single floating-point value for the texture level-of-detail bias, or 1.0, 2.0, or 4.0 when specifying the GL_RGB_SCALE or GL_ALPHA_SCALE.

30

OUTLINE

- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

31

31

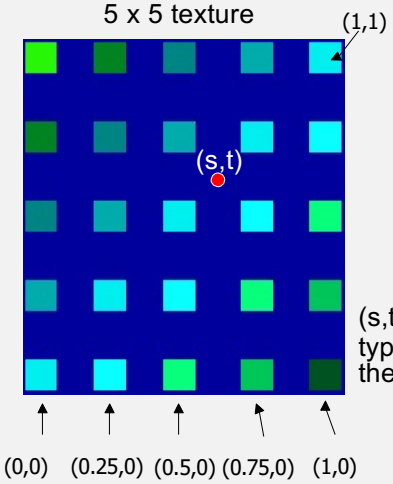
OUTLINE

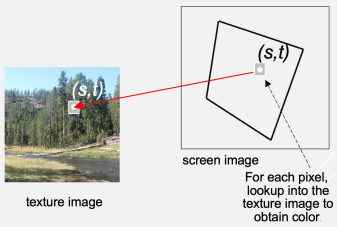
- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

32

TEXTURE INTERPOLATION

For given texture coordinates (s,t), we can find a unique image value, corresponding to the texture image at that location

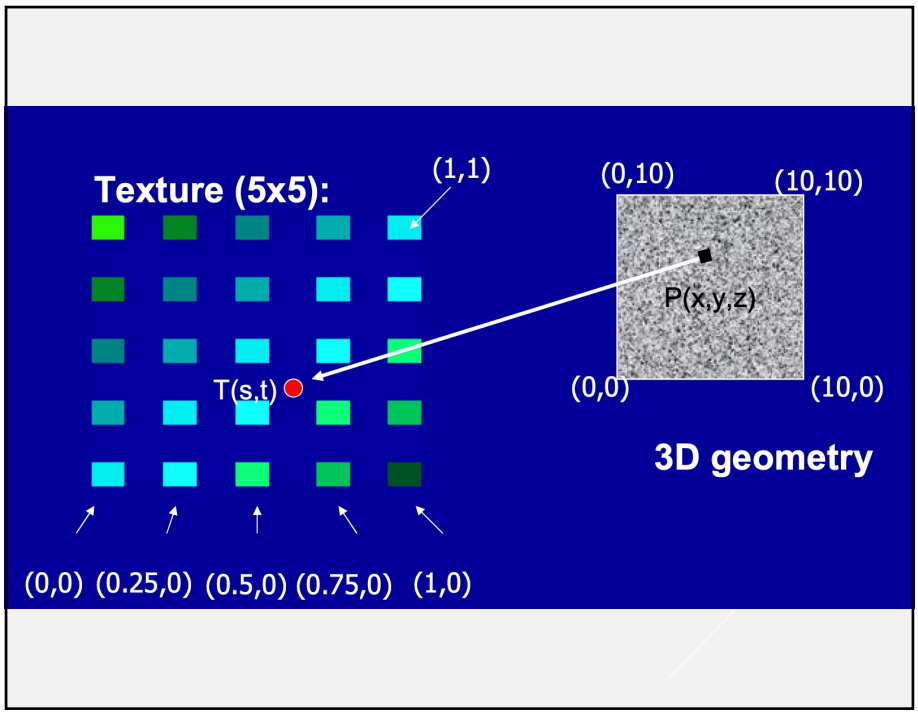




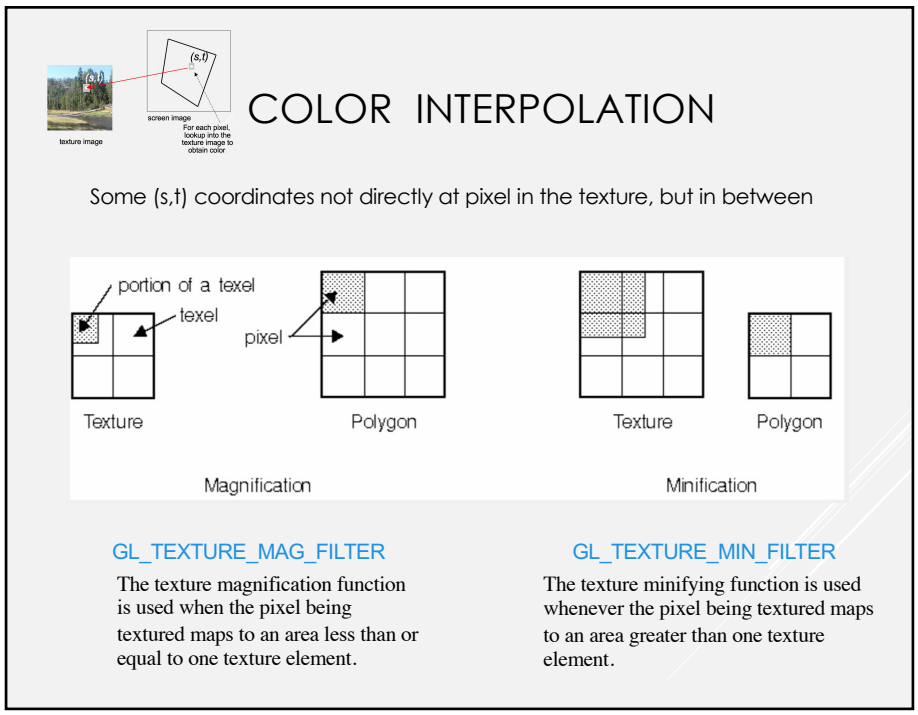
(s,t) coordinates typically not directly at pixel in the texture, but in between

23

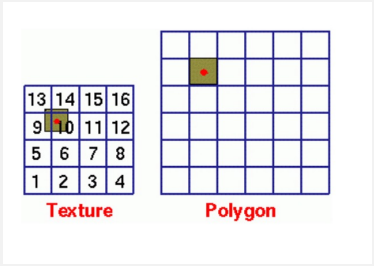
33


Texture (5x5):
(1,1)
(0,10)
(10,10)
P(x,y,z)
T(s,t)
(0,0)
(10,0)
3D geometry
(0,0) (0.25,0) (0.5,0) (0.75,0) (1,0)

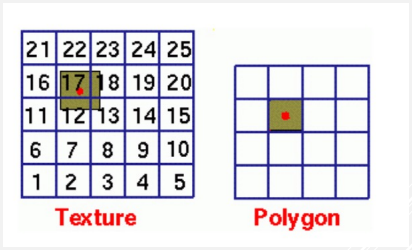
34


COLOR INTERPOLATION
texture image
screen image
For each pixel, lookup into the texture image to obtain color
Some (s,t) coordinates not directly at pixel in the texture, but in between
portion of a texel
texel
pixel
Texture
Polygon
Magnification
Texture
Polygon
Minification
GL_TEXTURE_MAG_FILTER
The texture magnification function is used when the pixel being textured maps to an area less than or equal to one texture element.
GL_TEXTURE_MIN_FILTER
The texture minifying function is used whenever the pixel being textured maps to an area greater than one texture element.

35



- Pixels map to less than one texel



- Pixels map to more than one texel

36

TEXTURE INTERPOLATION

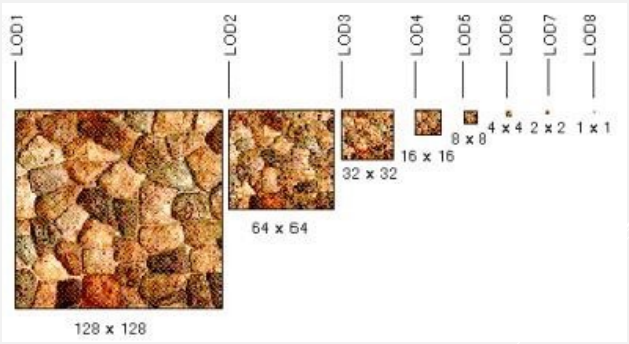
- (s,t) coordinates typically not directly at pixel in the texture, but in between
- Solutions:
  - Use the nearest neighbor to determine color
    - » Faster, but worse quality
    - » glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
  - Linear interpolation
    - » Incorporate colors of several neighbors to determine color
    - » Slower, better quality
    - » glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)

38

38

MIPMAPPING

- Pre-calculate how the texture should look at various distances, then use the appropriate texture at each distance
- Reduces / fixes the aliasing problem

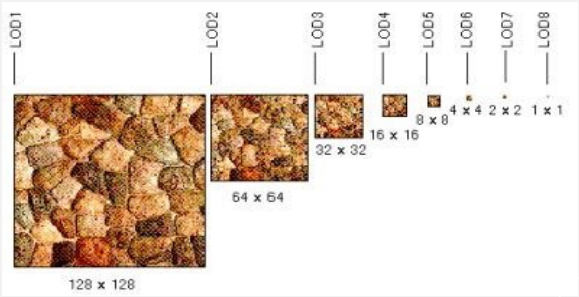


41

41

MIPMAPPING

- Each mipmap (each image below) represents a level of depth (LOD).
- Powers of 2 make things much easier.



42

42

MIPMAPPING IN OPENGL

- gluBuild2DMipmaps(GL_TEXTURE_2D, components, width, height, format, type, data)
  - This will generate all the mipmaps automatically

```
gluBuild2DMipmaps( GL_TEXTURE_2D, 3, 32,
32, GL_RGB, GL_UNSIGNED_BYTE,
texImage32 );
```

- glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST_MIPMAP_NEAREST)
  - This will tell GL to use the mipmaps for the texture

43

43

TEXTURE MAPPING IN OPENGL

- During your initialization:
  1. Read texture image from file into an array in memory, or generate the image using your program
  2. Specify texture mapping parameters
     - » Wrapping, filtering, etc.
  3. Initialize and activate the texture
- In display():
  1. Enable OpenGL texture mapping
  2. Draw objects: Assign texture coordinates to vertices
  3. Disable OpenGL texture mapping

44

44

INITIALIZING THE TEXTURE

- Do once during initialization, for each texture image in the scene, by calling glTextImage2D
- The dimensions of texture images must be powers of 2
  - if not, rescale image or pad with zero
  - or can use OpenGL extensions
- Can load textures dynamically if GPU memory is scarce

45

45

GLTEXIMAGE2D

glTexImage2D(GL_TEXTURE_2D, level, internalFormat, width, height, border, format, type, data)

- GL_TEXTURE_2D: specifies that it is a 2D texture
- Level: used for specifying levels of detail for mipmapping (default: 0)
- InternalFormat
  - Often: GL_RGB or GL_RGBA
  - Determines how the texture is stored internally
- Width, Height
  - The size of the texture must be powers of 2
- Border (often set to 0)
- Format, Type
  - Specifies what the input data is (GL_RGB, GL_RGBA, …)
  - Specifies the input data type (GL_UNSIGNED_BYTE, GL_BYTE, …)
  - Regardless of Format and Type, OpenGL convertes the data to internalFormat
- Data: pointer to the image buffer

20

46

ENABLE/DISABLE TEXTURE MODE

- Must be done before rendering any primitives that are to be texture-mapped
- glEnable(GL_TEXTURE_2D)
- glDisable(GL_TEXTURE_2D)

- Successively enable/disable texture mode to switch between drawing textured/non-textured polygons

- Changing textures:
  - Only one texture is active at any given time (with OpenGL extensions, more than one can be used simultaneously; this is called *multitexturing*)
  - Use glBindTexture to select the active texture

47

47

OUTLINE

- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

48

48

Complete example

```
void initTexture()
{
  load image into memory; // can use libjpeg, libtiff, or other image library
  // image should be stored as a sequence of bytes, usually 3 bytes per
    pixel (RGB), or 4 bytes (RGBA); image size is 4 * 256 * 256 bytes in
    this example
  // we assume that the image data location is stored in pointer
    "pointerToImage"

  // create placeholder for texture
  glGenTextures(1, &texName); // must declare a global variable in
    program header: GLUint texName
  glBindTexture(GL_TEXTURE_2D, texName); // make texture "texName"
    the currently active texture

  (continues on next page)
```

30

49

Complete example (part 2)

```
  // specify texture parameters (they affect whatever texture is active)
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
  GL_REPEAT); // repeat pattern in s
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
    GL_REPEAT); // repeat pattern in t

  // use linear filter both for magnification and minification
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
    GL_LINEAR);
  glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
    GL_LINEAR);

  // load image data stored at pointer "pointerToImage" into the currently
    active texture ("texName")
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 256, 256, 0,
          GL_RGBA, GL_UNSIGNED_BYTE, pointerToImage);

} // end init()
```

31

50

Complete example (part 3)

```
void display()
{
  …
  // no modulation of texture color with lighting; use texture color directly
  glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
    GL_REPLACE);

  // turn on texture mapping (this disables standard OpenGL lighting,
    unless in GL_MODULATE mode)
  glEnable(GL_TEXTURE_2D);

  (continues on next page)
```

32

51

Complete example (part 4)

```
  glBegin(GL_QUADS); // draw a textured quad
    glTexCoord2f(0.0,0.0); glVertex3f(-2.0,-1.0,0.0);
    glTexCoord2f(0.0,1.0); glVertex3f(-2.0,1.0,0.0);
    glTexCoord2f(1.0,0.0); glVertex3f(0.0,1.0,0.0);
    glTexCoord2f(1.0,1.0); glVertex3f(0.0,-1.0,0.0);
  glEnd();

  // turn off texture mapping
  glDisable(GL_TEXTURE_2D);

  // draw some non-texture mapped objects
    (standard OpenGL lighting will be used if it is enabled)
  …
  // switch back to texture mode, etc.
  …
} // end display()
```

33

52

OUTLINE

- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

53

53

Textures do not have to represent color

- Specularity (patches of shininess)
- Transparency (patches of clearness)
- Normal vector changes (bump maps)
- Reflected light (environment maps)
- Shadows
- Changes in surface height (displacement maps)

54

54

BUMP MAPPING

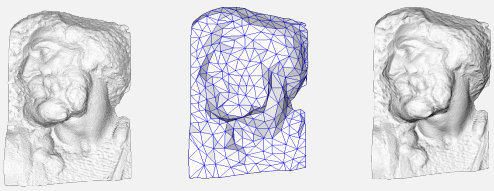
- How do you make a surface look *rough*?
  - Option 1: model the surface with many small polygons
  - Option 2: perturb the normal vectors before the shading calculation
    - » Fakes small displacements above or below the true surface
    - » The surface doesn't actually change, but shading makes it look like there are irregularities!
    - » A texture stores information about the "fake" height of the surface

55

55

BUMP MAPPING

- We can perturb the normal vector without having to make any actual change to the shape.
- This illusion can be seen through—how?



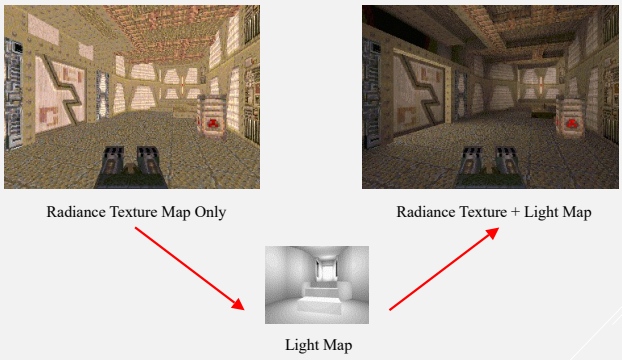
Original model (5M) | Simplified (500) | Simple model with bump map

57

57

LIGHT MAPPING

- *Quake* uses *light maps* in addition to texture maps. Texture maps are used to add detail to surfaces, and light maps are used to store pre-computed illumination. The two are multiplied together at run-time, and cached for efficiency.



58

58

SUMMARY

- Introduction
- Texture mapping in OpenGL
- Filtering and Mipmaps
- Example
- Non-color texture maps

59

59