Lecture 07: Buffers and Textures

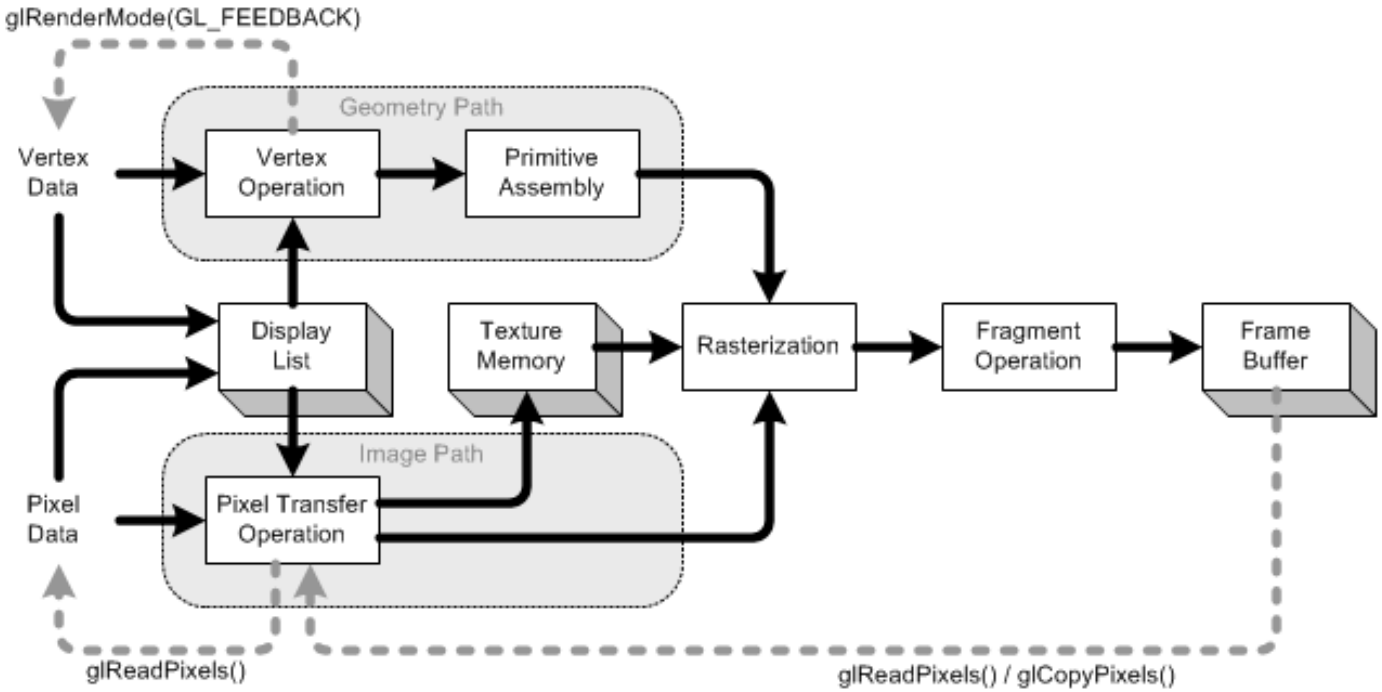
CSE 40166 Computer Graphics

Peter Bui

University of Notre Dame, IN, USA

October 26, 2010

OpenGL Pipeline



Today's Focus

- **Pixel Buffers**: read and write image data to and from OpenGL buffers.
- **Textures**: map images to OpenGL polygons/surfaces.

Buffer

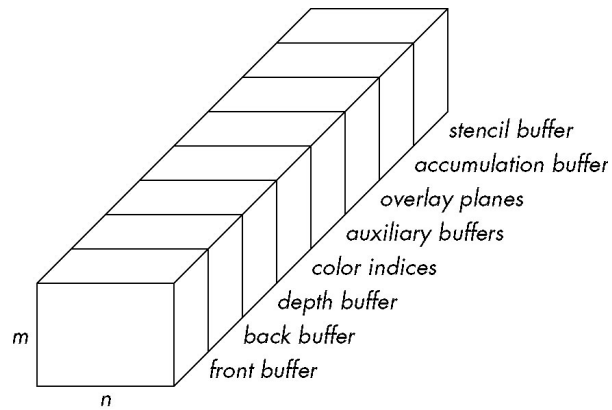
A block of memory with $n \times m$ k-bit elements.

OpenGL Buffers

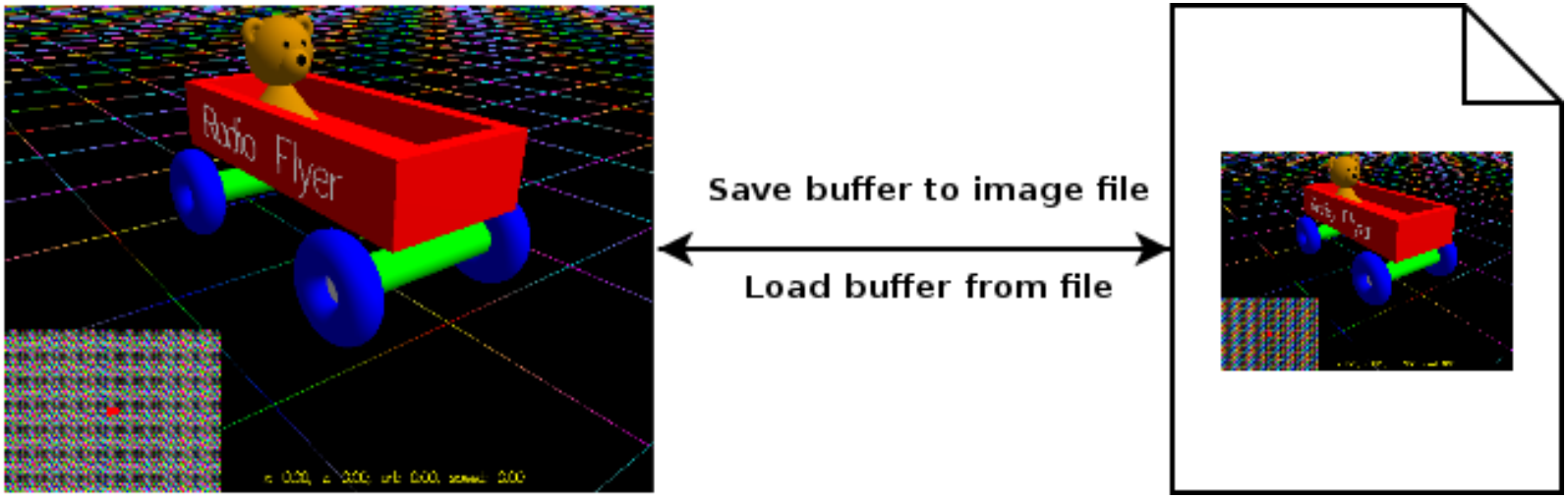
- Front + Back buffer
- Depth buffer

Common Properties

- **Precision**: numerical accuracy (determined by number of bits).
- **Bitplane**: any of the k $n \times m$ planes in a buffer.
- **Pixel**: all k of the bits at a particular spacial location.



Save and Load Buffers



- **Save buffer**: retrieve data from buffer and save as image file.
- **Load buffer**: retrieve data from image buffer and load buffer.

Must be careful that data is reformatted to be compatible with the frame buffer.

Digital Images

External Format

Images are arrays of pixels, but can be encoded in a variety of formats: *EPS, PNG, JPG, BMP, PPM*.

Internal Representation

In OpenGL, images are usually stored as 2D RGB arrays.

```
1 GLubyte          image[width][height][3];
2 unsigned char    image[width][height][3];
```

Portable Pixel Map PPM

Format

Simple compression-less image format consisting of a short header and body:

```
P6
# Simple PPM
512 512
255
<r><g><b>....
```

Library

```
unsigned char *buffer;
size_t width, height;

ppm_read("image.ppm", &buffer, &width, &height);
ppm_write("image2.ppm", buffer, width, height);
```

Saving Buffer to Image File

Steps

1. Select appropriate buffer.
2. Retrieve pixels from GPU to CPU.
3. Write buffer to image file.

Code

```
1 glReadBuffer(GL_BACK);
2 glReadPixels(0, 0, width, height,
3                GL_RGB, GL_UNSIGNED_BYTE, buffer);
4 ppm_write("image.ppm", buffer, width, height);
```

As shown in **Example 18**.

Loading Buffer from Image File

Steps

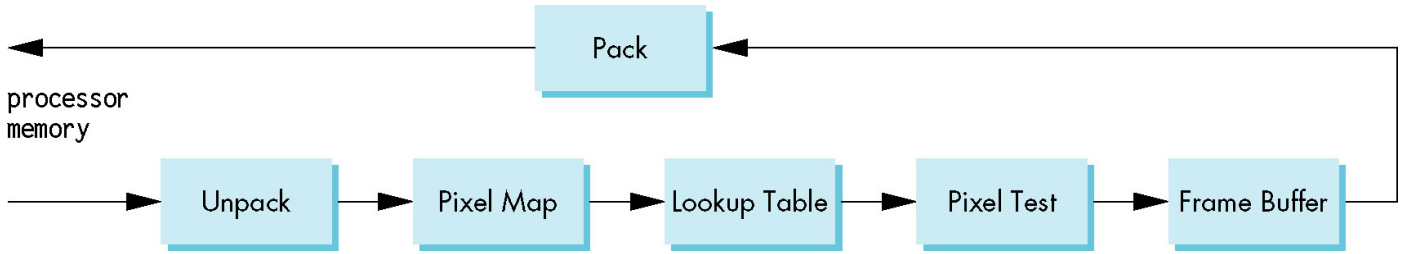
1. Read image file into buffer.
2. Select appropriate buffer.
3. Position raster and configure alignment.
4. Write pixels from CPU to GPU.

Code

```
1 ppm_read("image.ppm", &buffer, &width, &height);
2 glDrawBuffer(GL_BACK);
3 glRasterPos2i(0, 0);
4 glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
5 glReadPixels(width, height, GL_RGB,
6                GL_UNSIGNED_BYTE, buffer);
```

As shown in **Example 20**.

OpenGL Pixel Pipeline

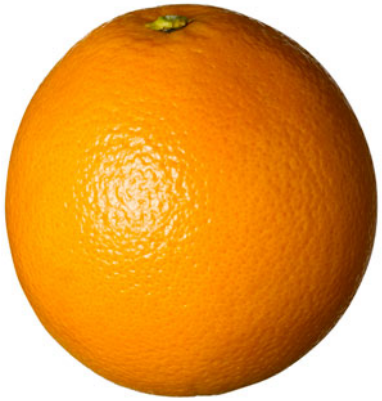


1. **Unpacking**: converts pixels from CPU buffer to internal OpenGL format.
2. **Mapping**: pixel values mapped using user-defined lookup tables.
3. **Testing**: check if pixel should be written, and how.
4. **Buffer**: store pixels in graphics buffer.
5. **Packing**: convert data from internal OpenGL format to application data format.

Textures

Motivation

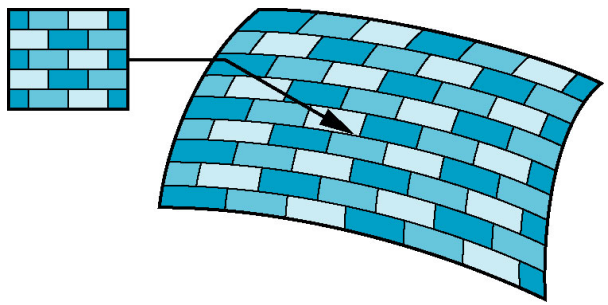
Modeling all details of a surface would require far too much computation and complexity.



Method

Map a pattern to a OpenGL polygon or surface.

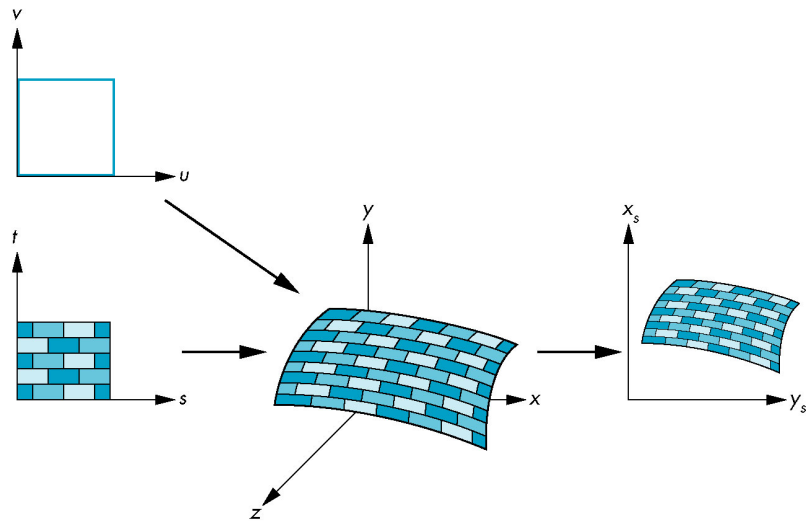
- Textures are patterns.
- Patterns are just digital images.
- Pin texture to polygon or surface.



Texture Mapping

Properties

- Texture is indexed with *UV/ST coordinates*: $(0, 0) - (1, 1)$.
- Texture usually represents color data, but could be used for other things.
- A single element is called a **texel**.



Mapping

Each vertex gets a set of a texture coordinates, which are used to pin the image onto each polygon.

Texture Mapping Techniques

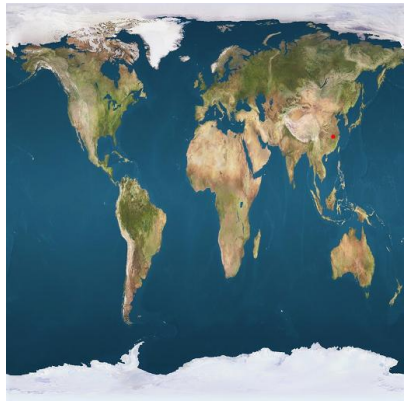


Figure: **Spherical**

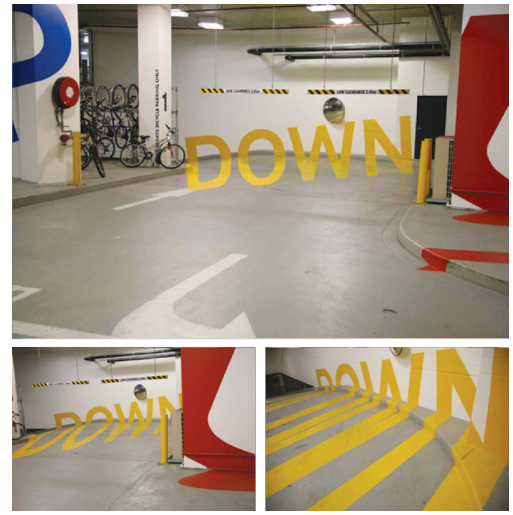


Figure: **Projections**

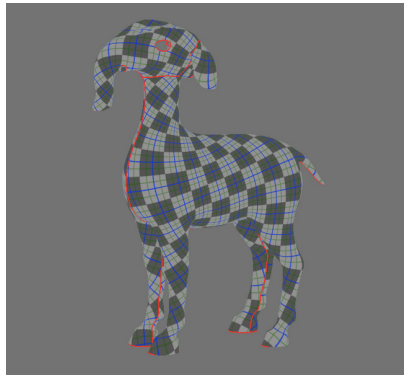


Figure: **UV Unwraping**

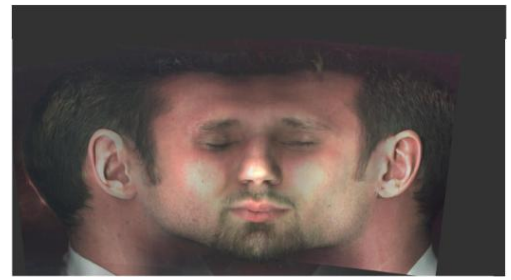


Figure: **Cylindrical**

Basic Texture Mapping in OpenGL

Setup

1. **Load Image**: read texture image from file.
2. **Texture Handle**: get unique id for texture using `glGenTextures`.
3. **Bind Texture**: tell state machine to use particular texture using `glBindTexture`.
4. **Set Texture Parameters**: use `glTexEnvf` and `glTexParameteri` to configure texture settings.
5. **Copy Image to Texture**: use `glTexImage2D`.

Usage

1. **Enable and Bind Texture**: use `glEnable(GL_TEXTURE_2D)`.
2. **Map Texture Coordinates**: use `glTexCoord2f(s, t)`.

Texture Setup Example

```
1  GLuint TextureId;
2
3  void
4  register_texture(GLuint *tid, GLubyte *buffer, size_t width, size_t height)
5  {
6      glGenTextures(1, tid);
7      glBindTexture(GL_TEXTURE_2D, *tid);
8      glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
9      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
10     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
11     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
12     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
13     glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
14     glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0,
15                  GL_RGB, GL_UNSIGNED_BYTE, buffer);
16 }
17
18 int
19 main(int argc, char *argv[])
20 {
21     unsigned char *buffer;
22     size_t width, height;
23
24     ppm_read("texture.ppm", &buffer, &width, &height);
25     register_texture(&TextureId, buffer, width, height);
26 }
```

Texture Usage Example

```
1  void
2  display()
3  {
4      double width  = (GLdouble)ImageWidth/(GLdouble)ImageHeight;
5      double height = 1.0;
6
7      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
8
9      glMatrixMode(GL_MODELVIEW);
10     glLoadIdentity();
11     gluLookAt(EyeX, EyeY, EyeZ, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
12
13     glEnable(GL_TEXTURE_2D);
14     glBindTexture(GL_TEXTURE_2D, ImageTextureId);
15
16     glColor3f(1.0, 1.0, 1.0);
17     glBegin(GL_QUADS); {
18         glTexCoord2f(0.0, 0.0);
19         glVertex3f(-width, -height, 0.0);
20
21         glTexCoord2f(1.0, 0.0);
22         glVertex3f( width, -height, 0.0);
23
24         glTexCoord2f(1.0, 1.0);
25         glVertex3f( width,  height, 0.0);
26
27         glTexCoord2f(0.0, 1.0);
28         glVertex3f(-width,  height, 0.0);
29     } glEnd();
30 }
```

As shown in **Example 21**.

Texture Parameters

TEXTURE_ENV_MODE

- **GL_MODULATE**: texture colors $\times$ object colors.
- **GL_REPLACE**: disable lighting and just use texture colors.
- **GL_DECAL**: when there's alpha, draw texture with α and let object's colors bleed through.

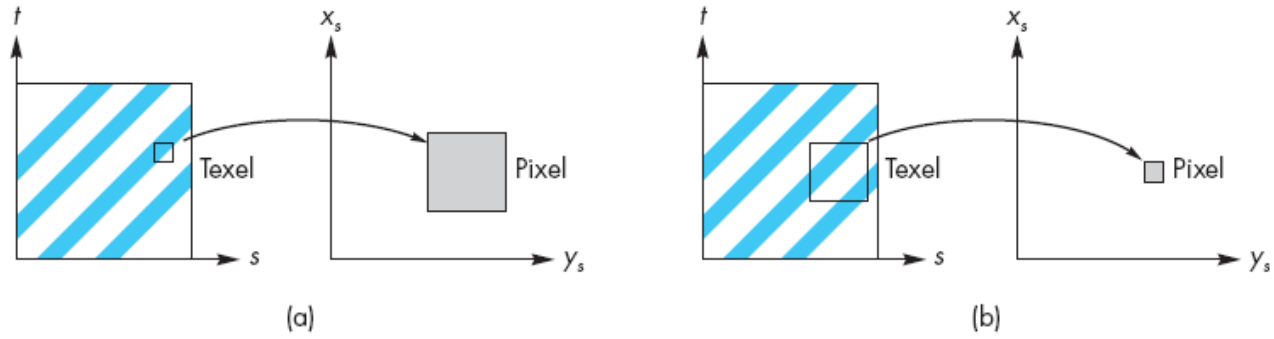
TEXTURE_MIN—MAG_FILTER

- **GL_NEAREST**: simplest nearest neighbor replacement.
- **GL_LINEAR**: linear interpolation for smoother values.

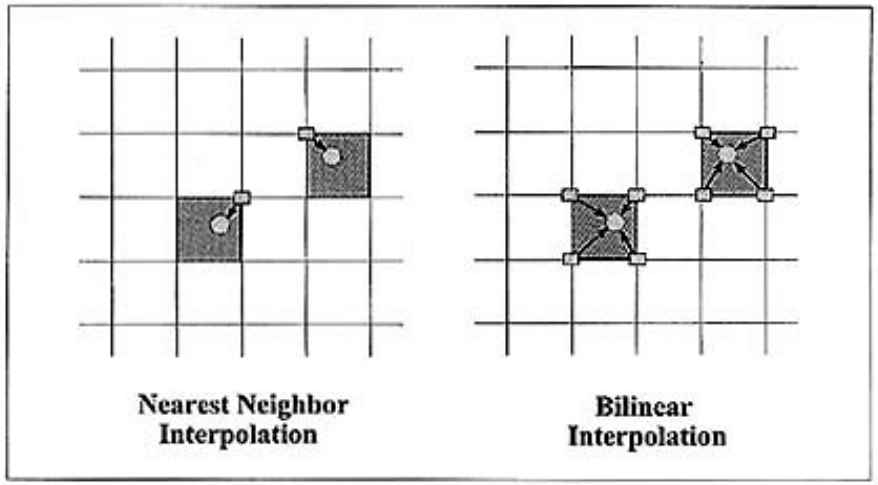
TEXTURE_WRAP_S—T

- **GL_REPEAT**: repeat pattern.
- **GL_CLAMP**: use edge values.

Texture Magnification/Minification Filtering



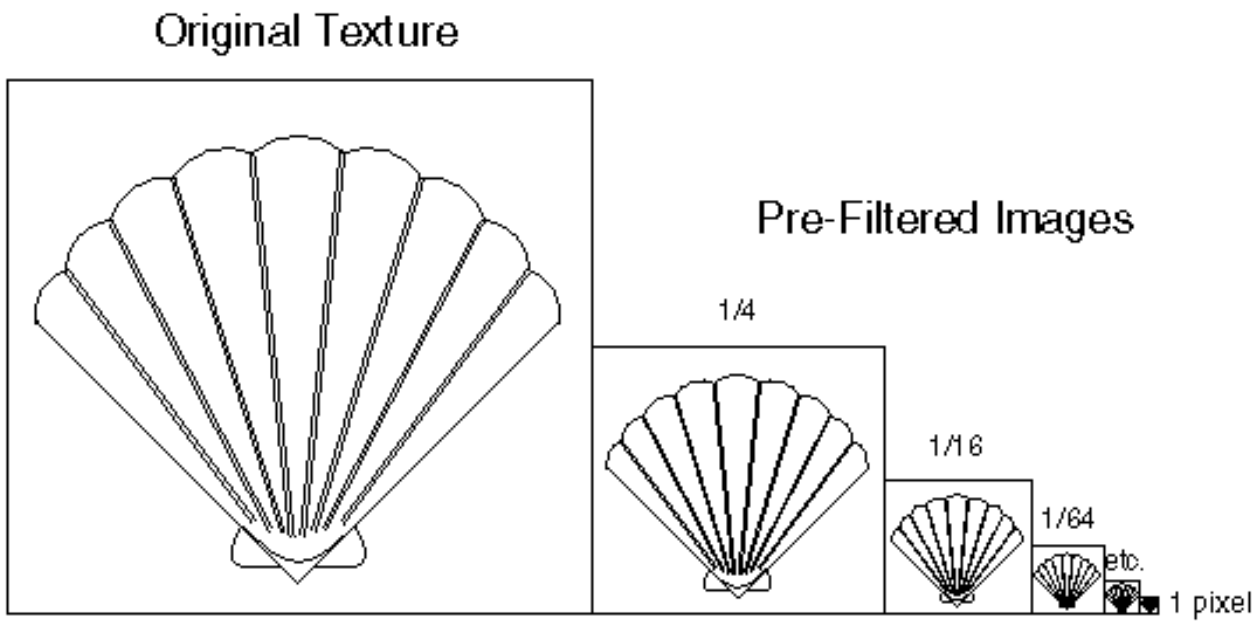
- **Magnification**: texel is larger than 1 pixel.
- **Minification**: texel is smaller than 1 pixel.



Texture Mipmapping

Mipmapping

Create copies of the texture at smaller resolutions and then index into those when the depth at the pixel is far away to avoid *antialiasing*.



Texture Wrapping

What happens if we map texture coordinates $> (1, 1)$?

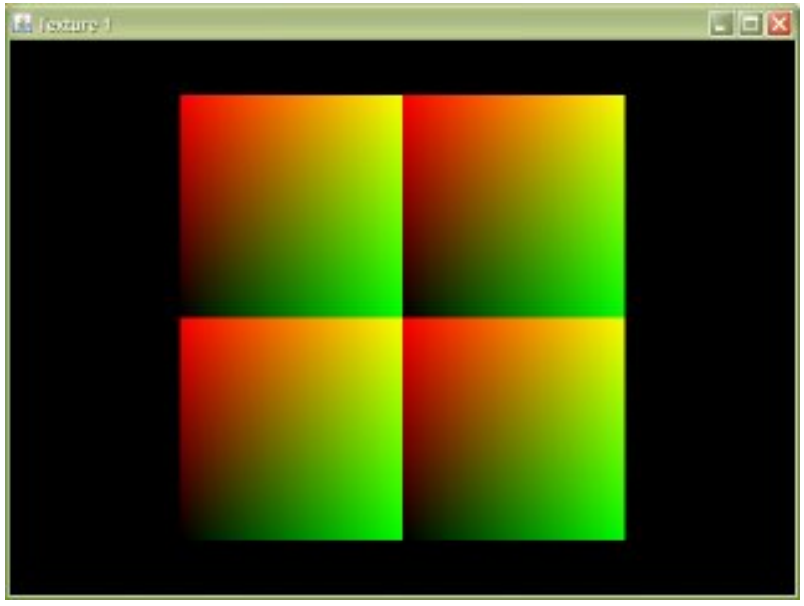


Figure: Repeat

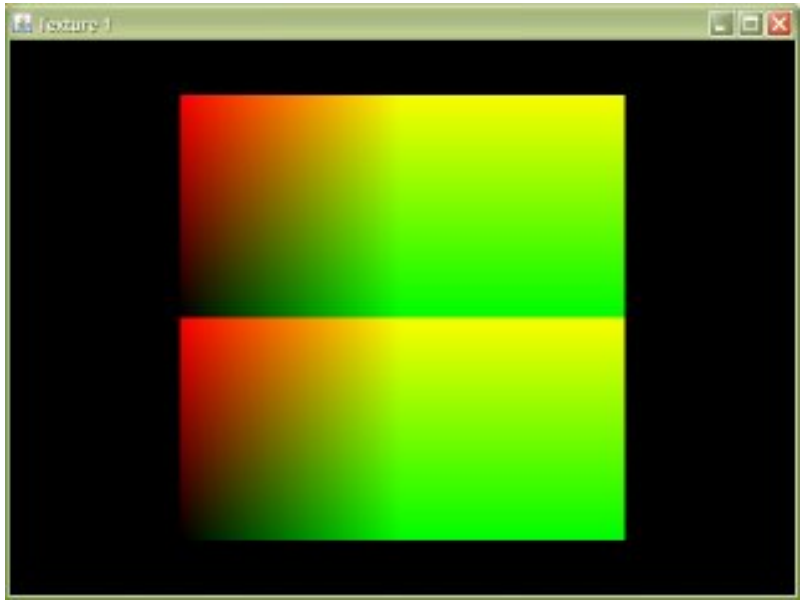


Figure: Clamp

Bump Mapping

Distorts the normal vectors during shading process to make the surface appear to have small variations in shape.

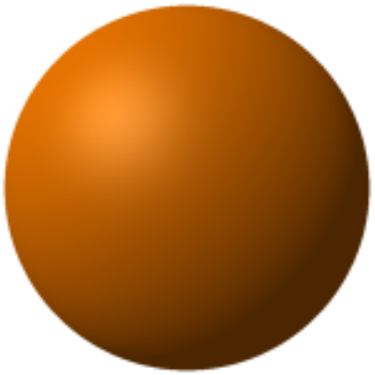


Figure: Smooth Sphere

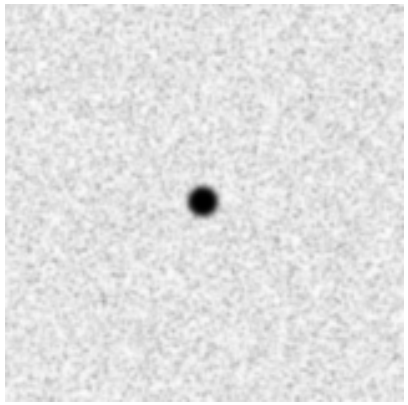


Figure: Bump Map

Figure: Orange