

VGP351 – Week 7

⇒ Agenda:

- Quiz #3
- Last day of texture mapping
 - Reflection mapping
 - Projective texturing
 - Texture atlases
 - Texture compression



26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping

- ⇒ Simulate reflections of the environment using a texture and texture coordinate calculations
 - Can either be called “environment mapping” or “reflection mapping”



26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping

- Forms of reflection mapping are classified by the shape used to simulate the environment
 - Cylindrical
 - Hemispherical
 - Spherical
 - Cube
 - Dual-paraboloid

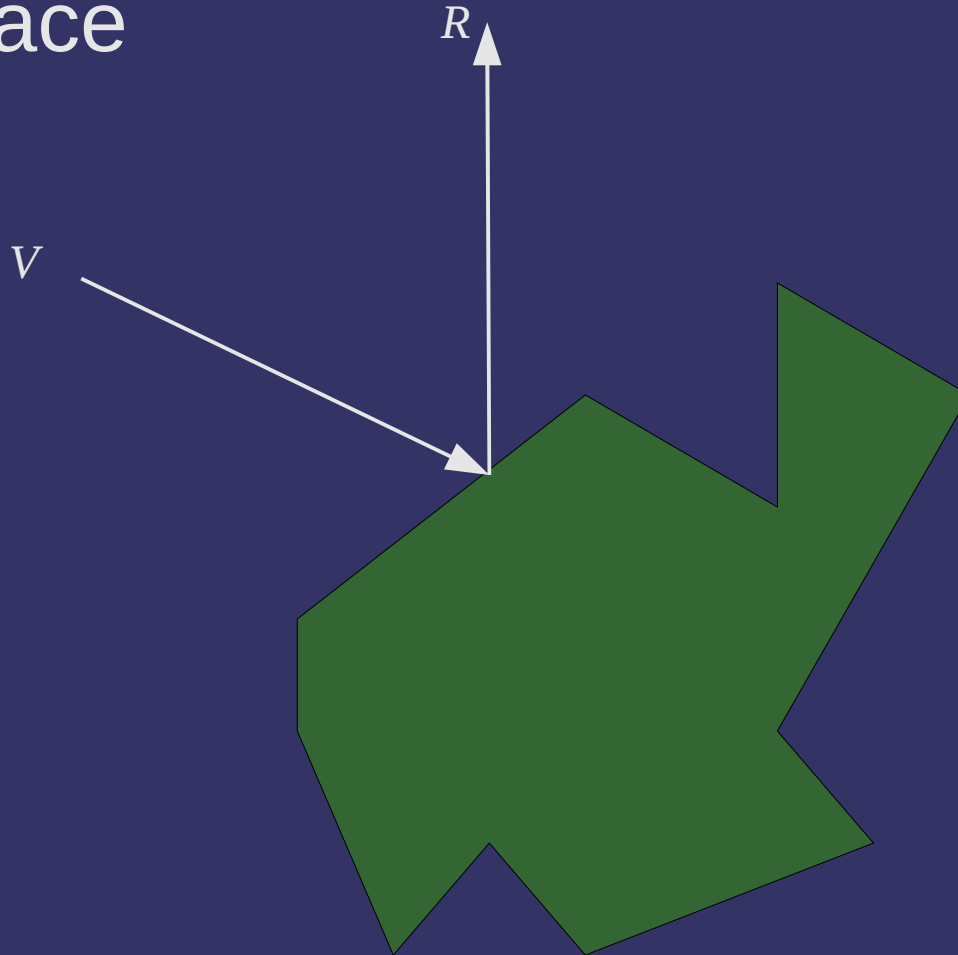


26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping

- Calculate the reflection vector and map to texture space

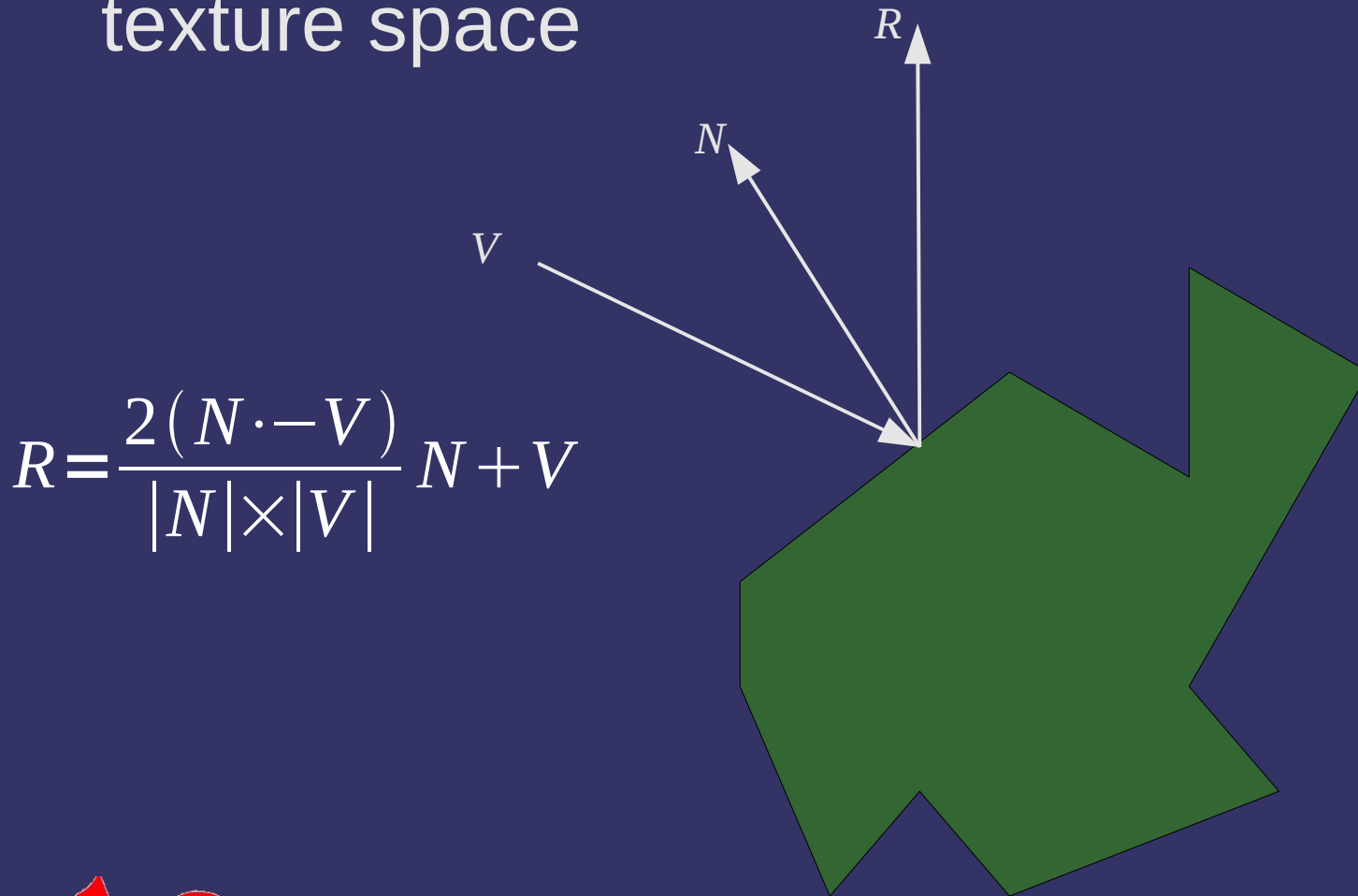


26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping

- Calculate the reflection vector and map to texture space



$$R = \frac{2(N \cdot -V)}{|N| \times |V|} N + V$$



26-February-2009

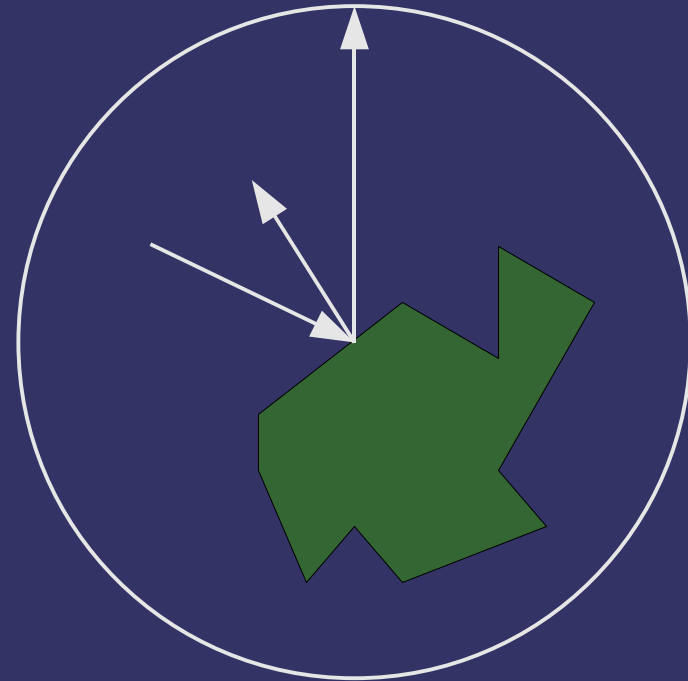
© Copyright Ian D. Romanick 2009

Reflection Mapping – Cylindrical

⇒ *Exactly* like cylindrical projection

- Use the reflection vector instead of the position

$$s = \frac{\text{atan}(x/z)}{2\pi}$$
$$t = y$$



26-February-2009

© Copyright Ian D. Romanick 2009

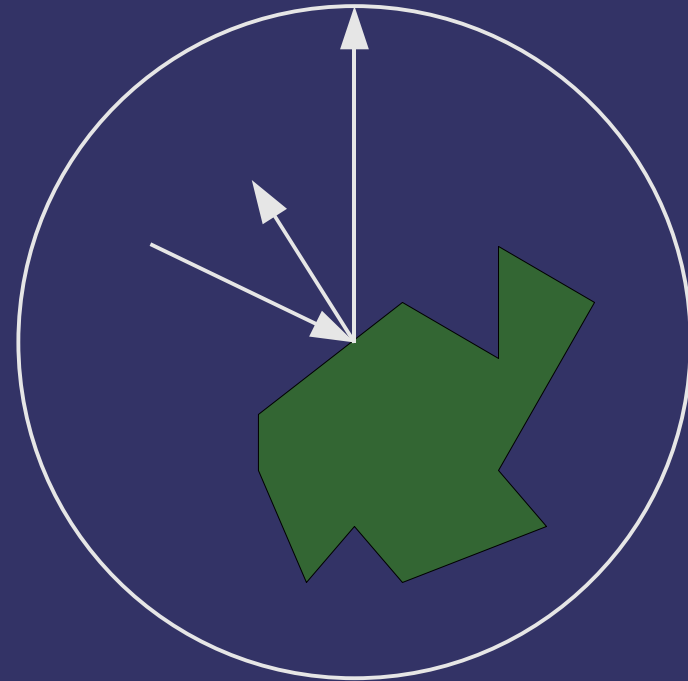
Reflection Mapping – Cylindrical

➤ Pros:

- Easy to implement
- Easy to get source images
- Only one texture image

➤ Cons:

- Distortion increases away from horizon
- Can't reflect sky or ground (i.e., $R = (0, \pm 1, 0)$)



26-February-2009

© Copyright Ian D. Romanick 2009

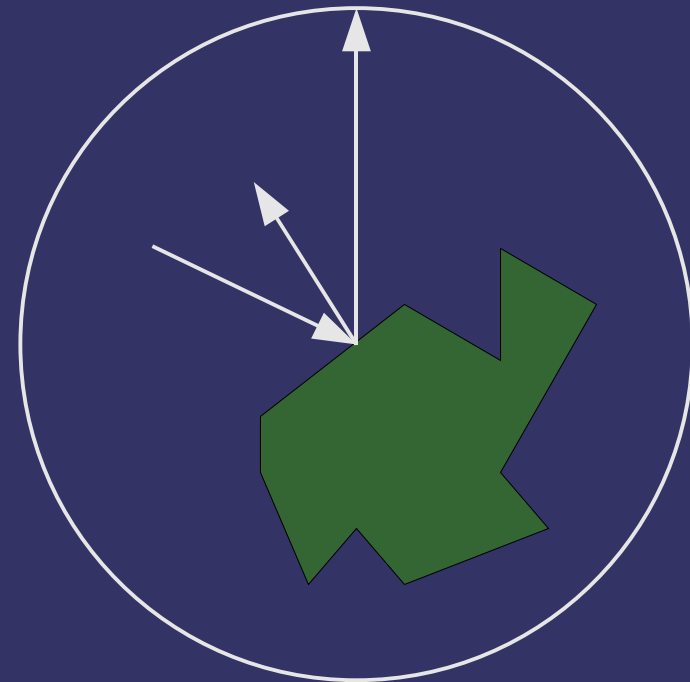
Reflection Mapping – Cylindrical

➤ Pros:

- Easy to implement
- Easy to get source images
- Only one texture image

➤ Cons:

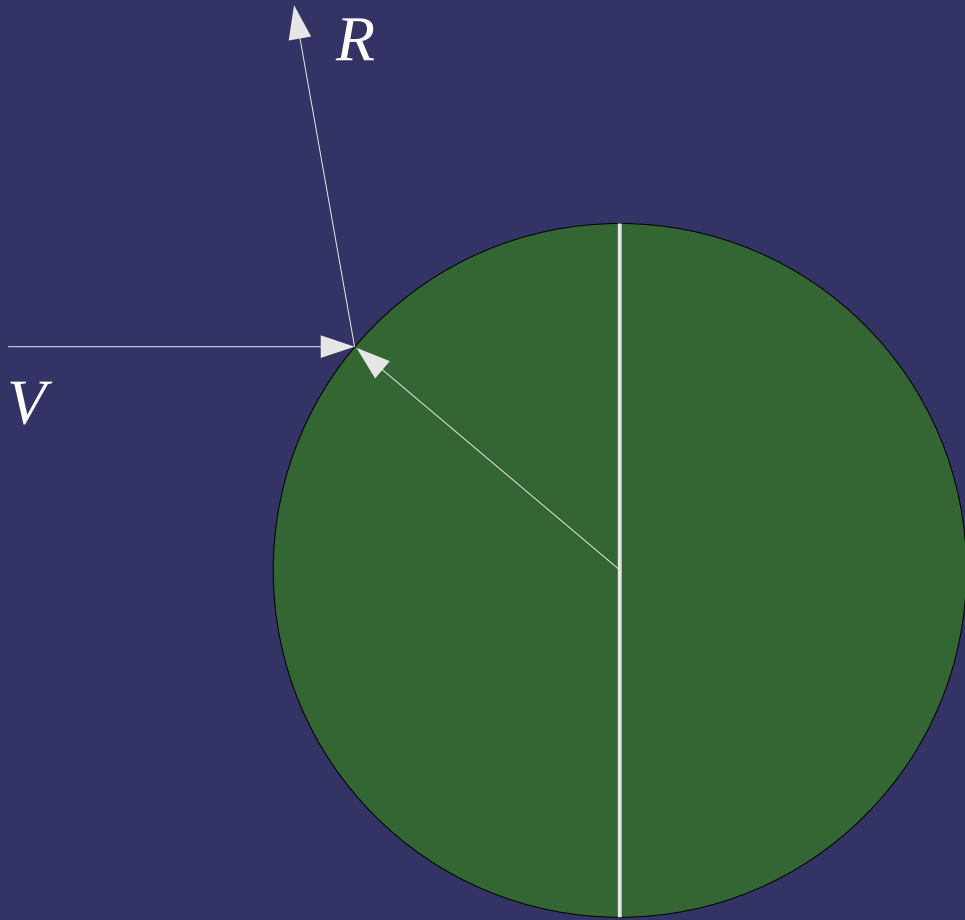
- Distortion increases away from horizon
- Can't reflect sky or ground (i.e., $R = (0, \pm 1, 0)$)



Caused by arctangent!
 $\arctan(x/z) \rightarrow \pm\infty$ as $z \rightarrow 0$,
and has a discontinuity
when $z = 0$.



Reflection Mapping – Hemispherical



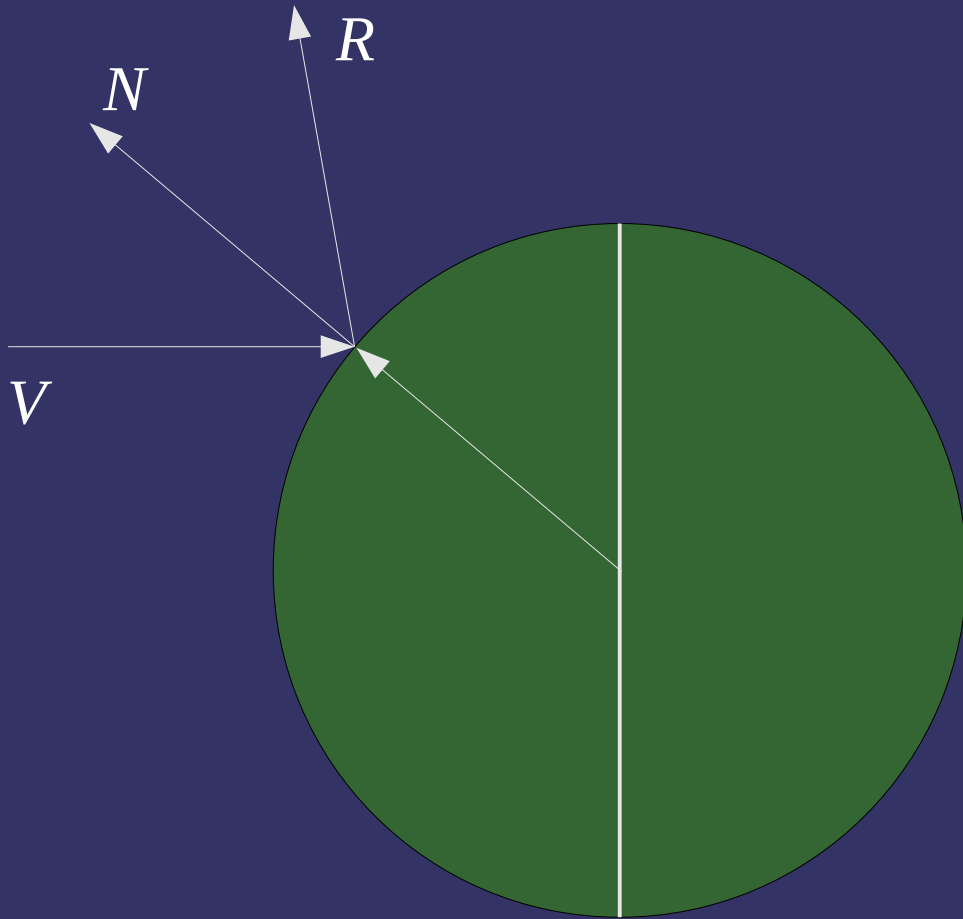
- Find location with matching infinite view vector and R



26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping – Hemispherical



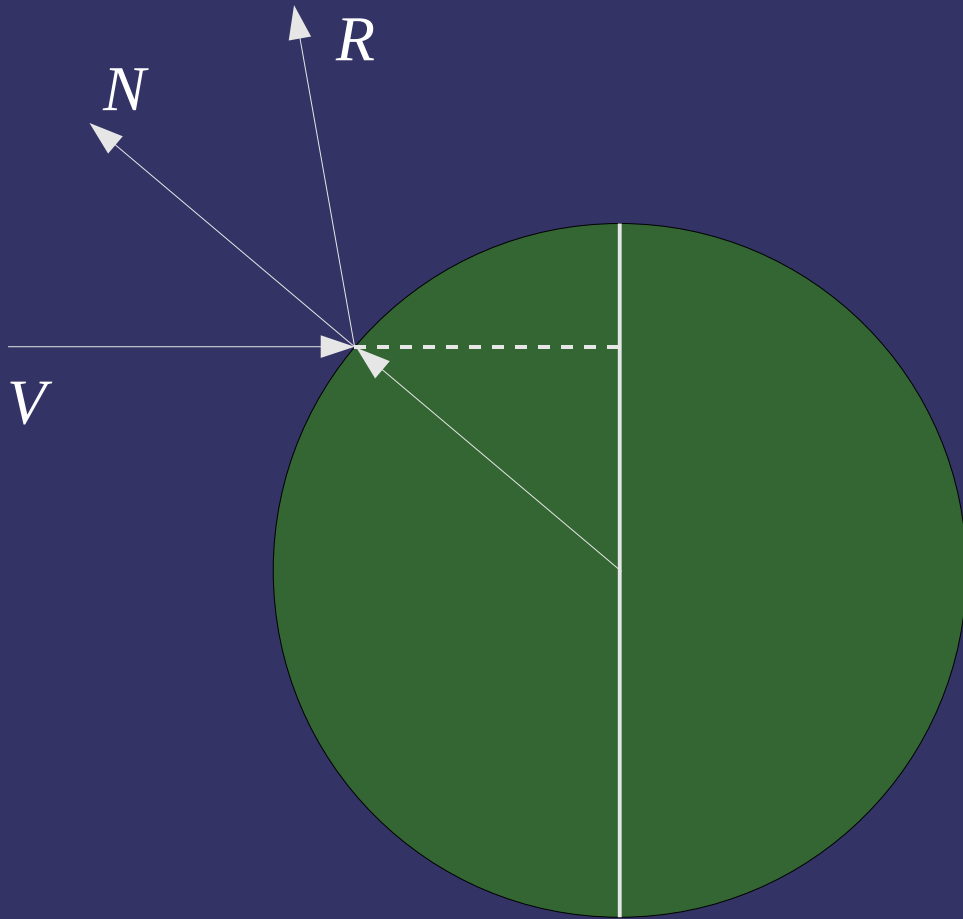
- ⇒ Find location with matching infinite view vector and R
 - This is the normal of the sphere at that location



26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping – Hemispherical



⇒ Find location with matching infinite view vector and R

- This is the normal of the sphere at that location
- Texture coordinate is the projection of this vector onto the image

$$s = \frac{R_x}{\sqrt{2(R_z + 1)}}$$

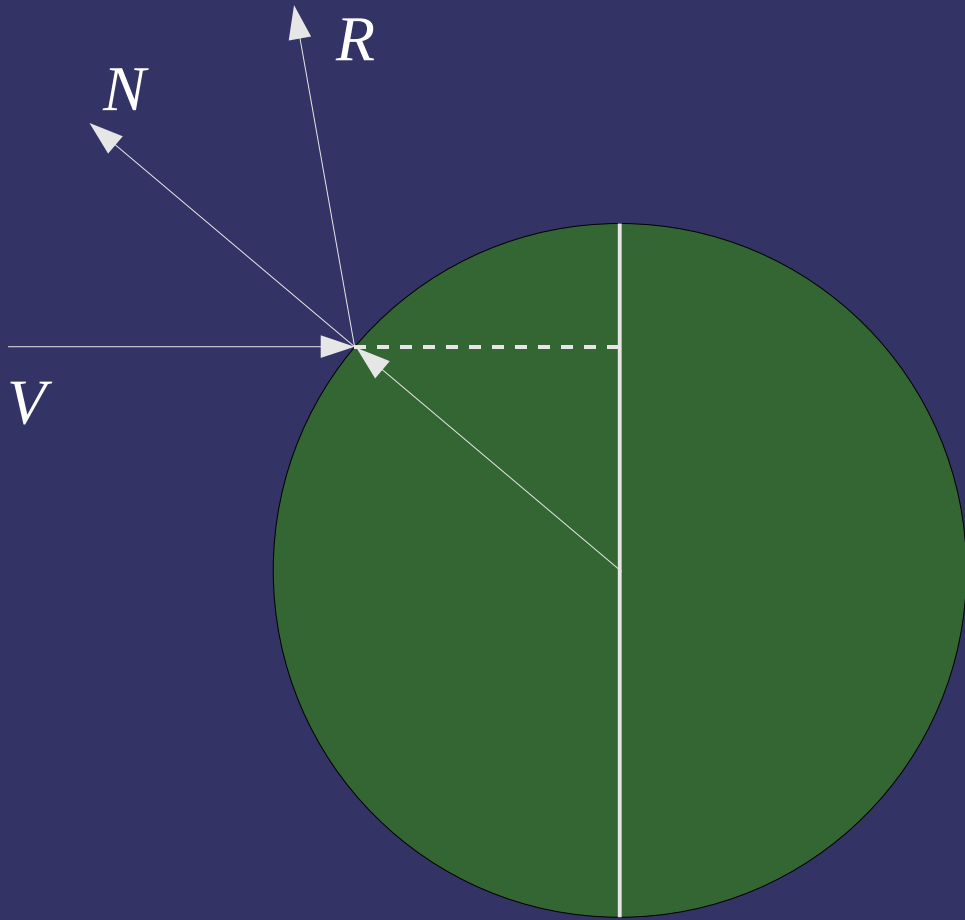
$$t = \frac{R_y}{\sqrt{2(R_z + 1)}}$$



26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping – Hemispherical



⇒ Find location with matching infinite view vector and R

- This is the normal of the sphere at that location
- Texture coordinate is the projection of this vector onto the image

$$s = \frac{R_x}{\sqrt{2(R_z + 1)}}$$
$$t = \frac{R_y}{\sqrt{2(R_z + 1)}}$$

Range [-1, 1]



26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping – Hemispherical

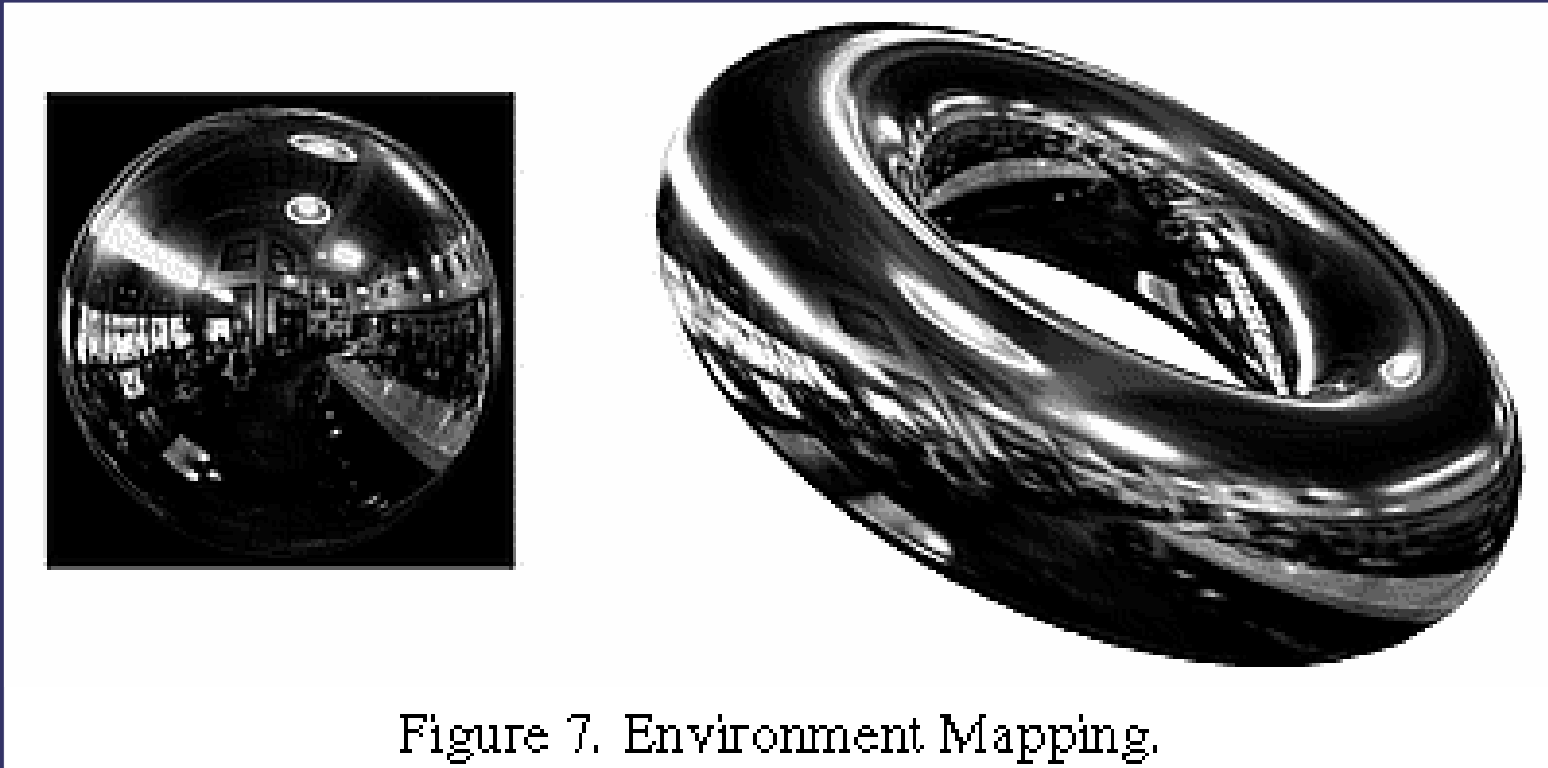
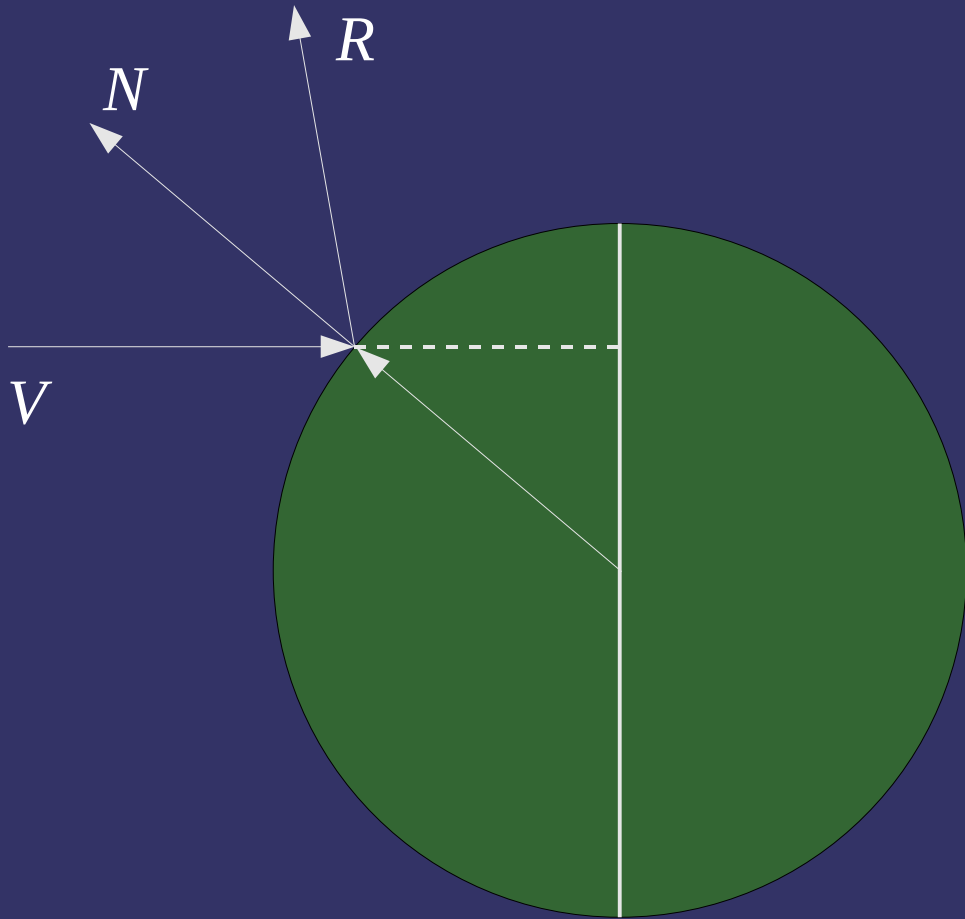


Image from <http://www.graficaobscura.com/texmap/index.html>

26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping – Hemispherical



➤ Pros:

- Easy to implement
- Easy to get source images
- Only one texture image

➤ Cons:

- Reflection map is viewpoint dependent
- Difficult to render to reflection map

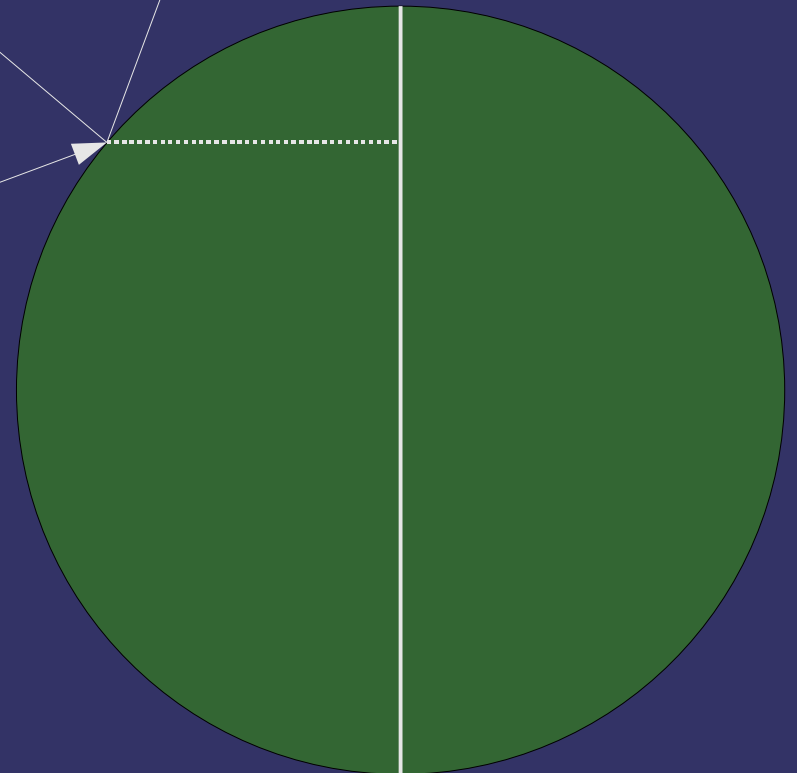


26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping – Spherical

- ⇒ Similar to hemispherical, but uses a local view
 - Note how the same position in the reflection map is now a reflection *behind* the sphere



26-February-2009

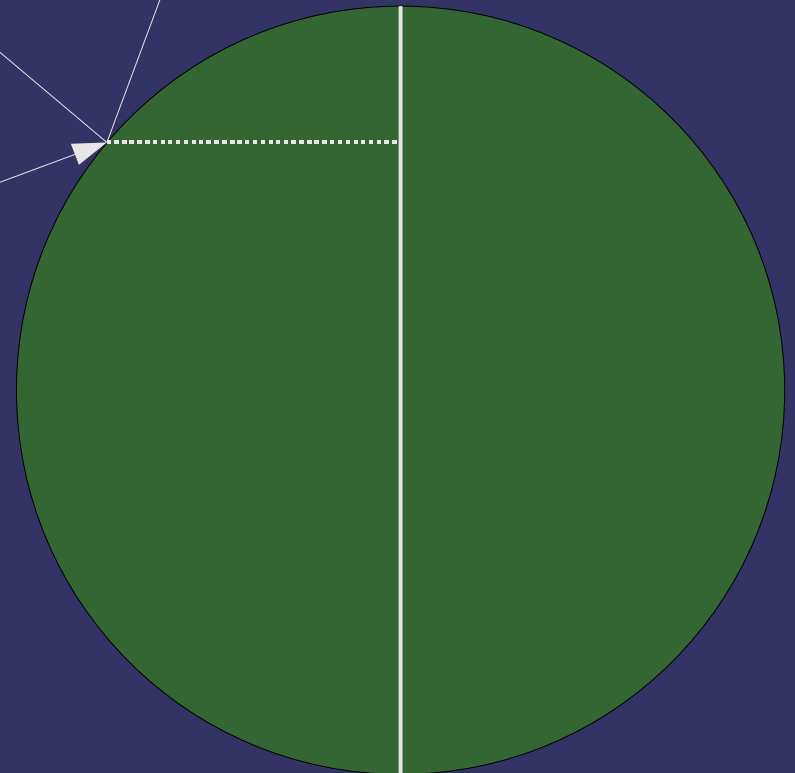
© Copyright Ian D. Romanick 2009

Reflection Mapping – Spherical

- ⇒ Similar to hemispherical, but uses a local view
- Note how the same position in the reflection map is now a reflection *behind* the sphere

$$s = \frac{R_x}{\sqrt{R_x^2 + R_y^2 + (R_z + 1)^2}}$$

$$t = \frac{R_y}{\sqrt{R_x^2 + R_y^2 + (R_z + 1)^2}}$$

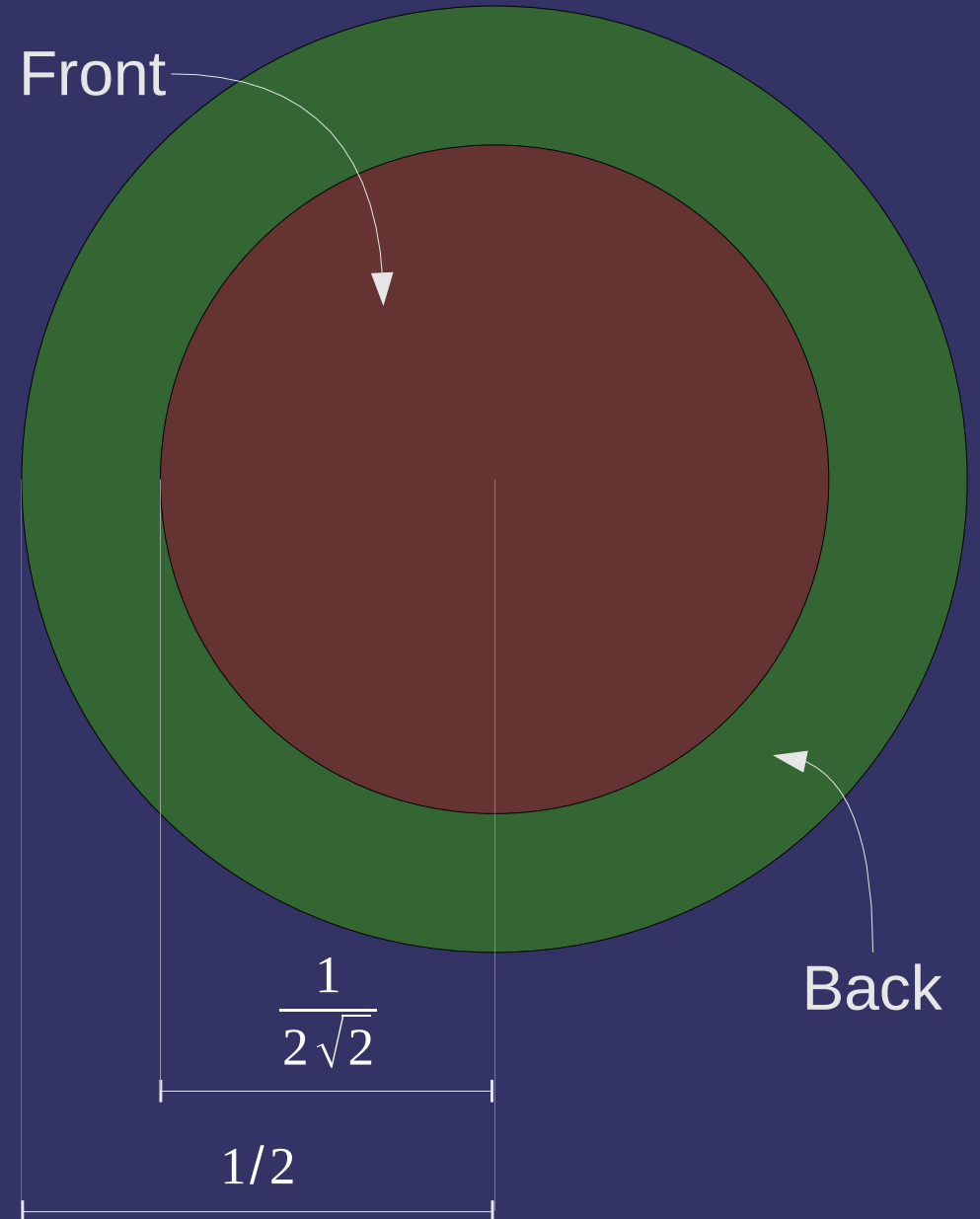


26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping – Spherical

- Single image for full 360° view



26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping – Spherical

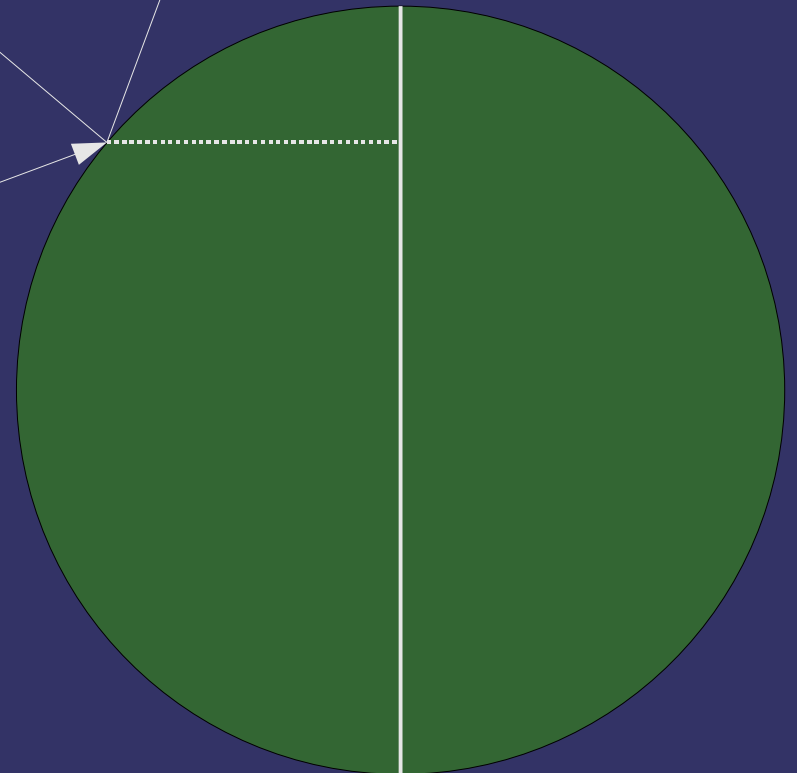
➤ Pros:

- Easy to implement
- Only one texture
- Local viewer and view independent

➤ Cons:

- Distortion increases as R diverges from V
- Difficult to get source images

Difficult to render
reflection map

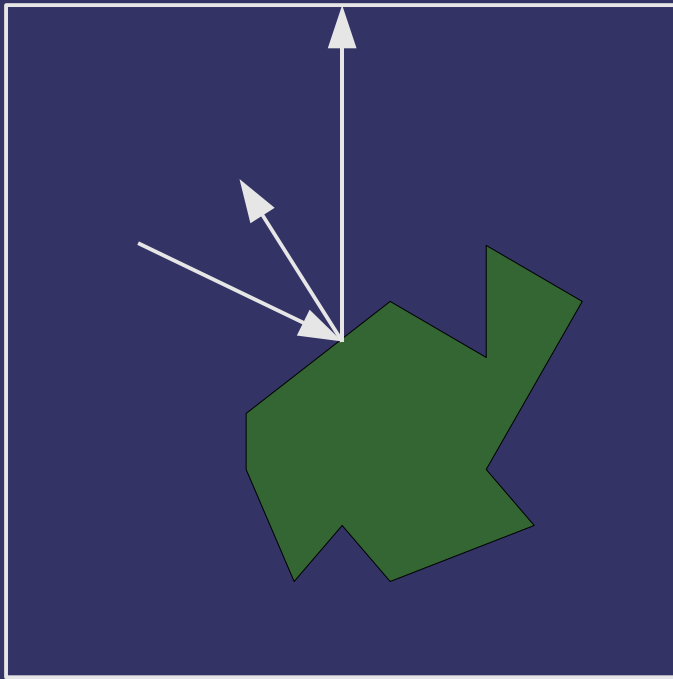


26 February 2009

© Copyright Ian D. Romanick 2009

Reflection Mapping – Cube

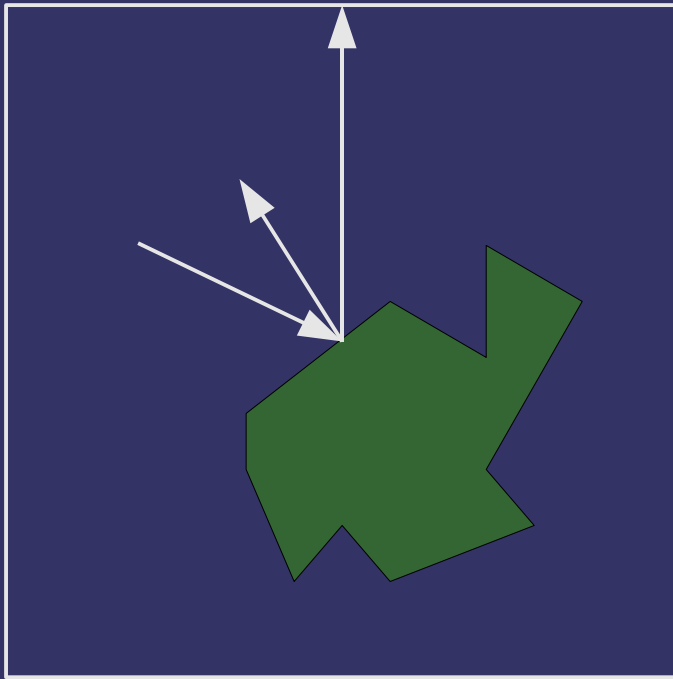
- Extend R to intersect unit cube surrounding point



26-February-2009

© Copyright Ian D. Romanick 2009

Reflection Mapping – Cube



➤ Pros:

- Trivial to implement
- Easy to render to reflection map

➤ Cons:

- Requires hardware support
- More difficult to get source images
- Discontinuities at cube-face boundaries



26-February-2009

© Copyright Ian D. Romanick 2009

Cube Maps

- ⇒ Consist of 6 equal sized, square textures
- ⇒ Accessed using a 3-component texture coordinate
 - Hardware uses largest magnitude component to select cube face
 - Intersection of vector with face determines 2D texture coordinate within that face

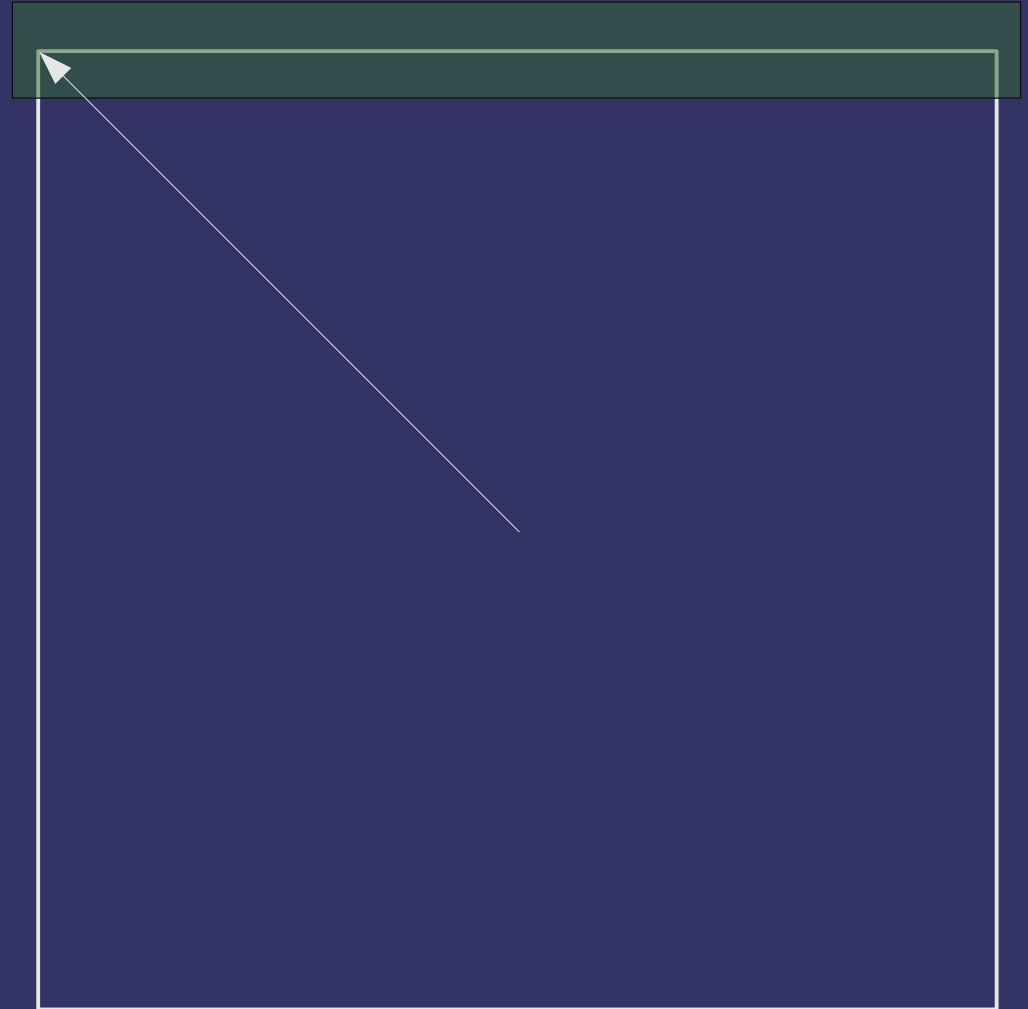


26-February-2009

© Copyright Ian D. Romanick 2009

Cube Maps

- Most hardware samples from *one* cube map face
 - What happens when the texture coordinate hits the edge texel of one face?

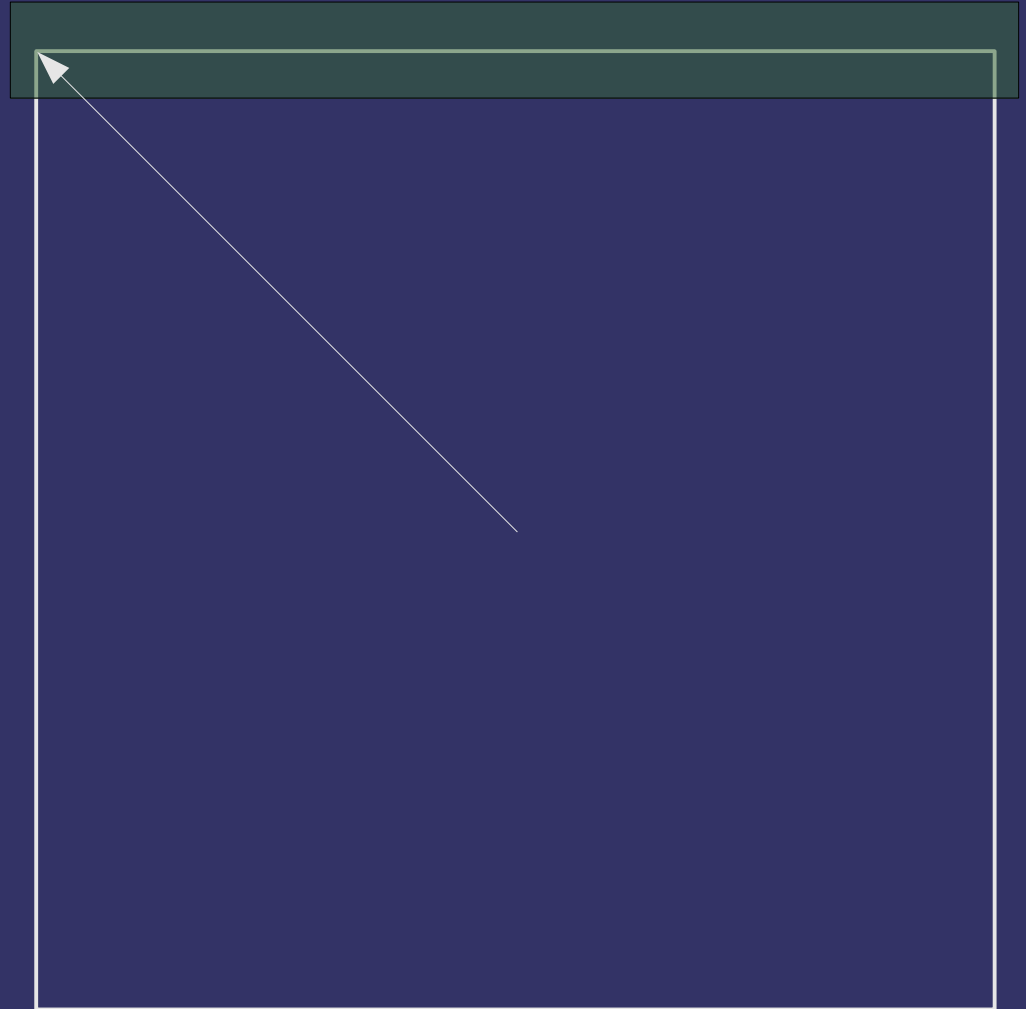


26-February-2009

© Copyright Ian D. Romanick 2009

Cube Maps

- Most hardware samples from *one* cube map face
 - Texel wrap modes are applied *within* the face
 - Use clamp-to-edge
 - Discontinuity at the boundaries
 - Can be fixed by using a texture border
 - Most hardware doesn't support this!



26-February-2009

© Copyright Ian D. Romanick 2009

Cube Maps

- ⇒ Bind texture to `GL_TEXTURE_CUBE_MAP`
- ⇒ Set texture data for specific cube faces using per-face targets:
 - `GL_TEXTURE_CUBE_MAP_POSITIVE_X`
 - `GL_TEXTURE_CUBE_MAP_NEGATIVE_X`
 - `GL_TEXTURE_CUBE_MAP_POSITIVE_Y`
 - `GL_TEXTURE_CUBE_MAP_NEGATIVE_Y`
 - `GL_TEXTURE_CUBE_MAP_POSITIVE_Z`
 - `GL_TEXTURE_CUBE_MAP_NEGATIVE_Z`



26-February-2009

© Copyright Ian D. Romanick 2009

Cube Maps

- Cube map textures must be *cube map complete*
 - If a mipmap filter mode is used, each face must be mipmap complete
 - Data must be available for all six faces
 - Level 0 of all six faces must be the same size
 - All faces must be square



26-February-2009

© Copyright Ian D. Romanick 2009

References

<http://www.debevec.org/ReflectionMapping/>

- Historical overview of reflection mapping. Includes references to many seminal papers and some good images.

<http://www.graficaobscura.com/texmap/index.html>

- The section on “Environment Mapping” provides additional background on the hemispherical technique.

<http://www.reindelsoftware.com/Documents/Mapping/Mapping.html>

- Good survey of most of the techniques discussed tonight.

<http://local.wasp.uwa.edu.au/~pbourke/miscellaneous/cube2cyl/>

- Description of a program to convert cubic environment maps to cylindrical environment maps or Blinn / Newell spherical environment maps. The pictures are worth well more than 1,000 words.



26-February-2009

© Copyright Ian D. Romanick 2009

Break



26-February-2009

© Copyright Ian D. Romanick 2009

Projective Texturing

- Applying a texture to a scene as though it were “projected” from a slide projector
 - Useful for various lighting effects
 - Complex shaped spot lights (i.e., flash light)
 - The basis of shadow mapping
 - You'll have to wait until VGP353



26-February-2009

© Copyright Ian D. Romanick 2009

Projective Texturing

- Fundamental problem: given a projector in world space and a point in world space, determine where the point is in texture space
- What does this sound like?



26-February-2009

© Copyright Ian D. Romanick 2009

Projective Texturing

- Fundamental problem: given a projector in world space and a point in world space, determine where the point is in texture space
- What does this sound like?
 - Projecting from world space (through camera space) to screen space
 - So we need a projector position, projection direction, a reference up direction, and the usual assortment of projection frustum parameters



26-February-2009

© Copyright Ian D. Romanick 2009

Projective Texturing

- Process is similar to viewing transformations:
 - Construct a transformation from world-space to projector-space
 - Construct a projection transformation for the projector's frustum
 - Transform each vertex by these matrices
 - Divide by Z
 - Result is the texture coordinate...almost



26-February-2009

© Copyright Ian D. Romanick 2009

Projective Texturing

- ⇒ View coordinates have a range $[-1, 1]$, but texture coordinates have range $[0, 1]$



26-February-2009

© Copyright Ian D. Romanick 2009

Projective Texturing

- View coordinates have a range $[-1, 1]$, but texture coordinates have range $[0, 1]$

$$s = (x - 1.0) / 2.0$$

$$t = (y - 1.0) / 2.0$$



26-February-2009

© Copyright Ian D. Romanick 2009

Projective Texturing

- View coordinates have a range $[-1, 1]$, but texture coordinates have range $[0, 1]$

$$s = (x - 1.0) / 2.0$$
$$t = (y - 1.0) / 2.0$$

Translation

Scale



26-February-2009

© Copyright Ian D. Romanick 2009

Projective Texturing

- View coordinates have a range $[-1, 1]$, but texture coordinates have range $[0, 1]$

$$M_{\text{bias}} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



26-February-2009

© Copyright Ian D. Romanick 2009

Projective Texturing

- What happens if a point *behind* the projection point is projected?



26-February-2009

© Copyright Ian D. Romanick 2009

Projective Texturing

- What happens if a point *behind* the projection point is projected?
 - It gets *inverted* onto the image plane
 - This is called an *anti-projection*



26-February-2009

© Copyright Ian D. Romanick 2009

References

http://en.wikipedia.org/wiki/Projective_texture_mapping

http://developer.nvidia.com/object/Projective_Texture_Mapping.html

http://www.ozone3d.net/tutorials/glsl_texturing_p08.php



26-February-2009

© Copyright Ian D. Romanick 2009

Cost of State Changes

- ⇒ Changing state can be expensive
 - At the very least, most hardware will have to flush internal data cache
 - One of the more expensive pieces of state to change is a texture binding



26-February-2009

© Copyright Ian D. Romanick 2009

Cost of State Changes

- Most common strategy to reduce state changes is sorting
 - Objects are sorted by common state and rendered in batches
 - This is a hassle and may not always be possible



26-February-2009

© Copyright Ian D. Romanick 2009

Texture Atlases

- ⇒ The number of texture binding changes can be reduced by packing multiple images into a single texture
 - When multiple texture maps are combined into a single, larger texture, it is called a *texture atlas*

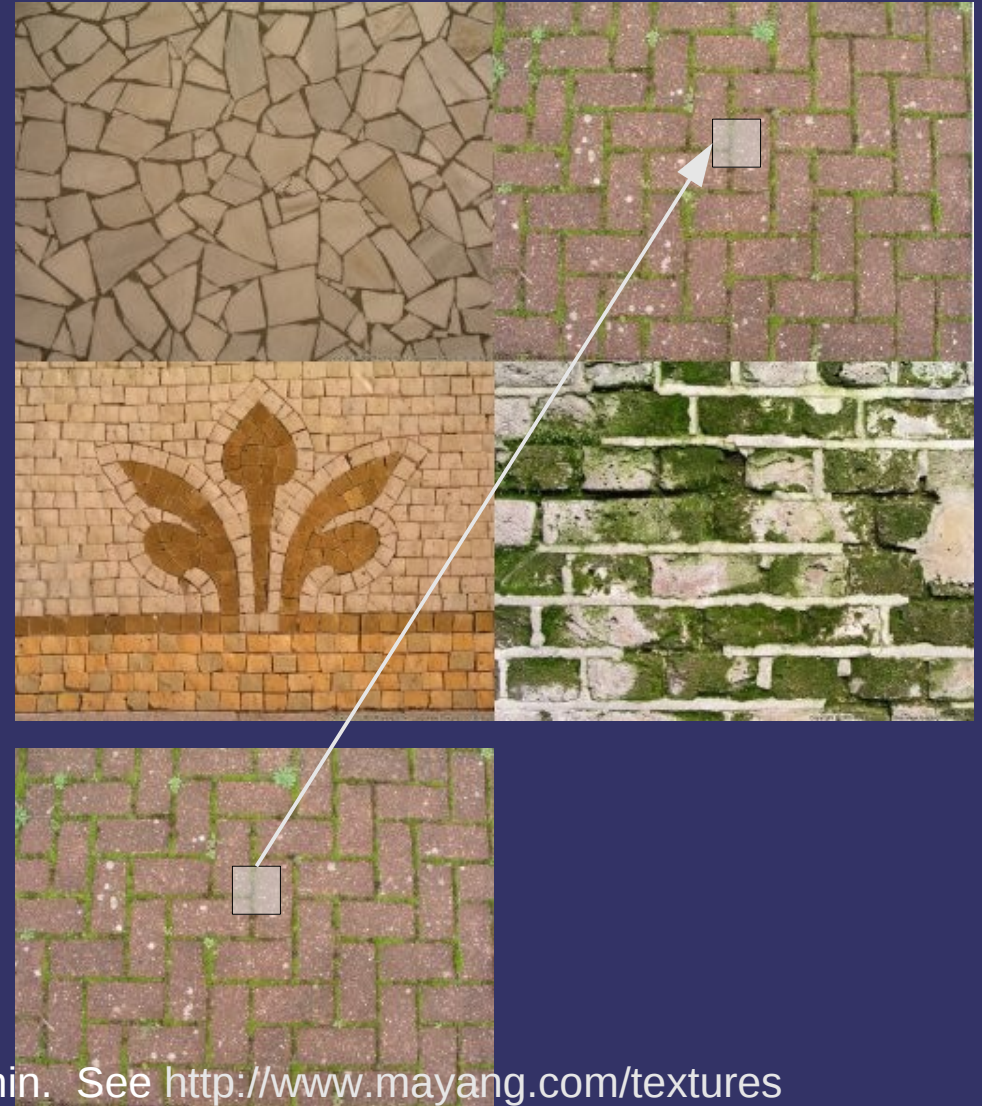


26-February-2009

© Copyright Ian D. Romanick 2009

Texture Atlases

- Texture coordinates must be updated for use with an atlas
 - Scale to the relative size within the atlas
 - Bias to the base position within the atlas



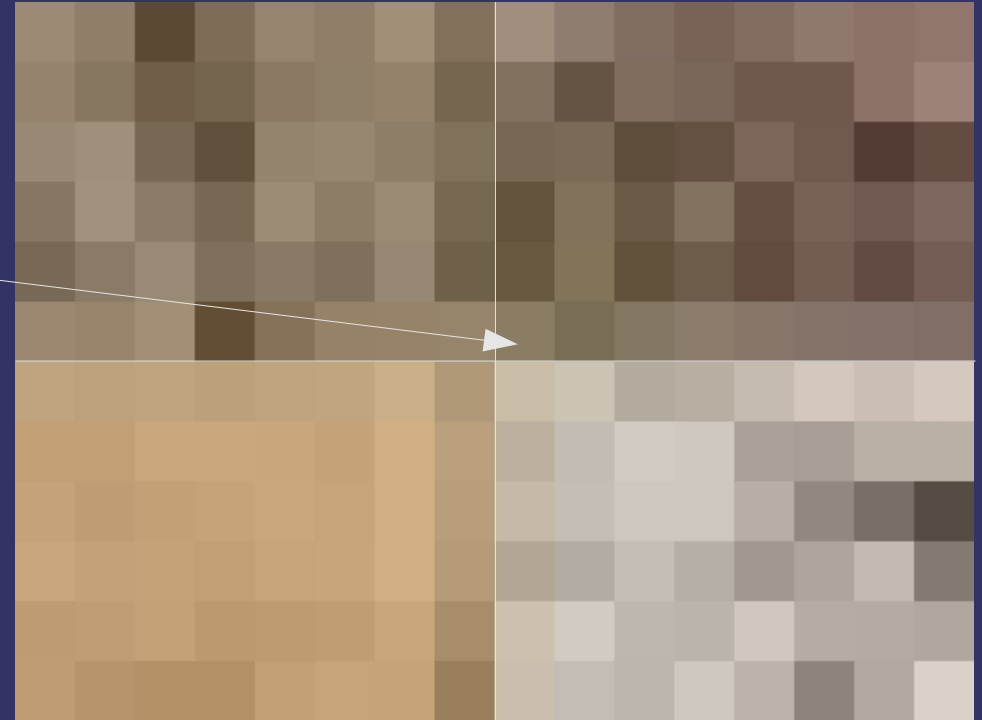
Images Copyright © 2006 Mayang Adnin. See <http://www.mayang.com/textures>

26-February-2009

© Copyright Ian D. Romanick 2009

Texture Atlases

- ⇒ Care must be taken around borders!
 - Sampling this point



Images Copyright © 2006 Mayang Adnin. See <http://www.mayang.com/textures>

26-February-2009

© Copyright Ian D. Romanick 2009

Texture Atlases

- ⇒ Care must be taken around borders!
 - Sampling this point
 - Will use this filter area



Images Copyright © 2006 Mayang Adnin. See <http://www.mayang.com/textures>

26-February-2009

© Copyright Ian D. Romanick 2009

Texture Atlases

- Care must also be taken with mipmapping

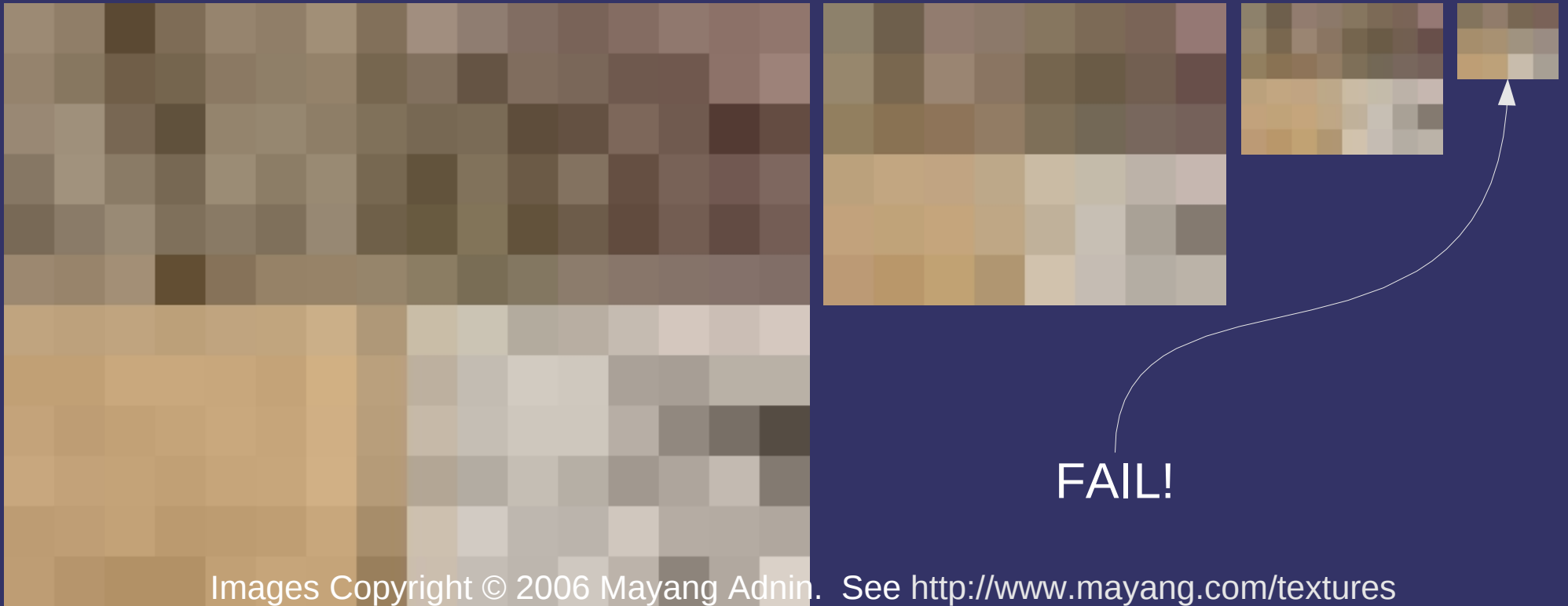


26-February-2009

© Copyright Ian D. Romanick 2009

Texture Atlases

- Care must also be taken with mipmapping
 - Clamping the LOD can fix this



26-February-2009

© Copyright Ian D. Romanick 2009

References

http://www.gamasutra.com/features/20060126/ivanov_01.shtml



26-February-2009

© Copyright Ian D. Romanick 2009

The Balancing Act...

- ⇒ Want to have numerous, highly detailed textures
 - Reduce aliasing
 - Prevent repetitive use of identical textures
- ⇒ Want to have high performance rendering
 - Want to keep all textures in fast, on-card memory
 - Want to minimize bandwidth required to access textures



26-February-2009

© Copyright Ian D. Romanick 2009

Compression

- Two usual ways to reduce storage requirements:
 - Have less data to store
 - Compress data



26-February-2009

© Copyright Ian D. Romanick 2009

Compression

- ⇒ Compression is used all the time!
 - Zip
 - Rar
 - JPEG
 - MPEG
 - MP3
 - etc.



26-February-2009

© Copyright Ian D. Romanick 2009

Compression

- General data compression techniques have rely on a common principle:

Reduce data size by storing redundancies in a compact manner

- Each data set has a different amount of redundancy:

```
-rw-rw-r-- 1 idr idr 20005 2009-02-25 18:41 crazy_paving_4142298.JPG  
-rw-rw-r-- 1 idr idr 23246 2009-02-25 18:42 diagonal_pattern_brick_flooring_9181152.JPG  
-rw-rw-r-- 1 idr idr 22886 2009-02-25 18:42 tiles_golden_feathers_motif_4142310.JPG  
-rw-rw-r-- 1 idr idr 29135 2009-02-25 18:42 wet_lichen_brick_5132630.JPG
```

- All four images are the same resolution and color depth



26-February-2009

© Copyright Ian D. Romanick 2009

Compression

- Variable compression is unsuitable for texture storage
 - Variably compressed data must be *serially* accessed to find a particular data element
 - Textures are accessed randomly
 - Texture-fetch hardware must quickly convert a texture coordinate to a texel address



26-February-2009

© Copyright Ian D. Romanick 2009

Texture Compression

- Several fixed-ratio compression techniques exist specifically for textures:
 - S3TC
 - FXT1
 - PVR-TC
 - ETC
- All techniques compress a rectangular block of texels into a fixed size block
 - Blocks are usually either 2×2 or 2×4



26-February-2009

© Copyright Ian D. Romanick 2009

Texture Compression

⇒ What's the trade-off?



26-February-2009

© Copyright Ian D. Romanick 2009

Texture Compression

⇒ What's the trade-off?

- Access speed improves
- Compression ratio decreases
 - JPEG regularly achieves 10:1 or 20:1 where as most texture compression algorithms only achieve 4:1
- Quality decreases
 - Each block is compressed the same amount (ratio) regardless of how much redundancy is actually available
 - Hand-wavy description: if there isn't 4:1 worth of redundancy, actual data is thrown away



26-February-2009

© Copyright Ian D. Romanick 2009

Texture Compression

- Specify that OpenGL compress textures for you
 - Use one of the *generic* compressed formats for the `internalFormat` specified to `glTexImage2D`
 - `GL_COMPRESSED_ALPHA`
 - `GL_COMPRESSED_LUMINANCE`
 - `GL_COMPRESSED_LUMINANCE_ALPHA`
 - `GL_COMPRESSED_INTENSITY`
 - `GL_COMPRESSED_RGB`
 - `GL_COMPRESSED_RGBA`



26-February-2009

© Copyright Ian D. Romanick 2009

Texture Compression

- Specify that OpenGL compress textures for you
 - Use one of the *specific* compressed formats for the `internalFormat` specified to `glTexImage2D`
 - `GL_COMPRESSED_RGB_S3TC_DXT1_EXT`
 - `GL_COMPRESSED_RGBA_S3TC_DXT1_EXT`
 - `GL_COMPRESSED_RGBA_S3TC_DXT3_EXT`
 - `GL_COMPRESSED_RGBA_S3TC_DXT5_EXT`
 - etc.



26-February-2009

© Copyright Ian D. Romanick 2009

Texture Compression

- Determine which compressed formats are available:
 - Find out how many formats by querying `GL_NUM_COMPRESSED_TEXTURE_FORMATS`
 - Get the list of formats by querying `GL_COMPRESSED_TEXTURE_FORMATS`



26-February-2009

© Copyright Ian D. Romanick 2009

Texture Compression

⇒ Specify pre-compressed data:

```
void glCompressedTexImage1D(GLenum target,  
    GLint level, GLenum internalformat,  
    GLsizei width, GLint border,  
    GLsizei imageSize, const GLvoid *data);
```

```
void glCompressedTexSubImage1D(GLenum target,  
    GLint level, GLint xoffset, GLsizei width,  
    GLenum format, GLsizei imageSize,  
    const GLvoid *data);
```



26-February-2009

© Copyright Ian D. Romanick 2009

Texture Compression

⇒ Determine the actual format of a texture:

```
void glGetTexLevelParameteriv(GLenum target,  
                               GLint level, GLenum pname, GLint *params);
```

- Pass `GL_TEXTURE_INTERNAL_FORMAT` for `pname`

⇒ Read back a compressed texture:

```
void glGetCompressedTexImage(GLenum target,  
                             GLint level, GLvoid *img);
```

- Will fail if the internal format is not a compressed format



26-February-2009

© Copyright Ian D. Romanick 2009

References

http://www.gamasutra.com/features/20051228/sherrod_01.shtml



26-February-2009

© Copyright Ian D. Romanick 2009

Next week...

- ⇒ Introduction to procedural shaders
 - Fundamental techniques
 - Noise
 - Antialiasing procedural shaders



26-February-2009

© Copyright Ian D. Romanick 2009

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of Intel or the Art Institute of Portland.

OpenGL is a trademark of Silicon Graphics, Inc. in the United States, other countries, or both.

Khronos and OpenGL ES are trademarks of the Khronos Group.

Other company, product, and service names may be trademarks or service marks of others.



26-February-2009

© Copyright Ian D. Romanick 2009