VGP351 – Week 9

- Agenda:
  - Quiz #4
  - Last day of texture mapping
    - Reflection mapping
    - Projective texturing
    - Texture atlases
    - Texture compression

Reflection Mapping

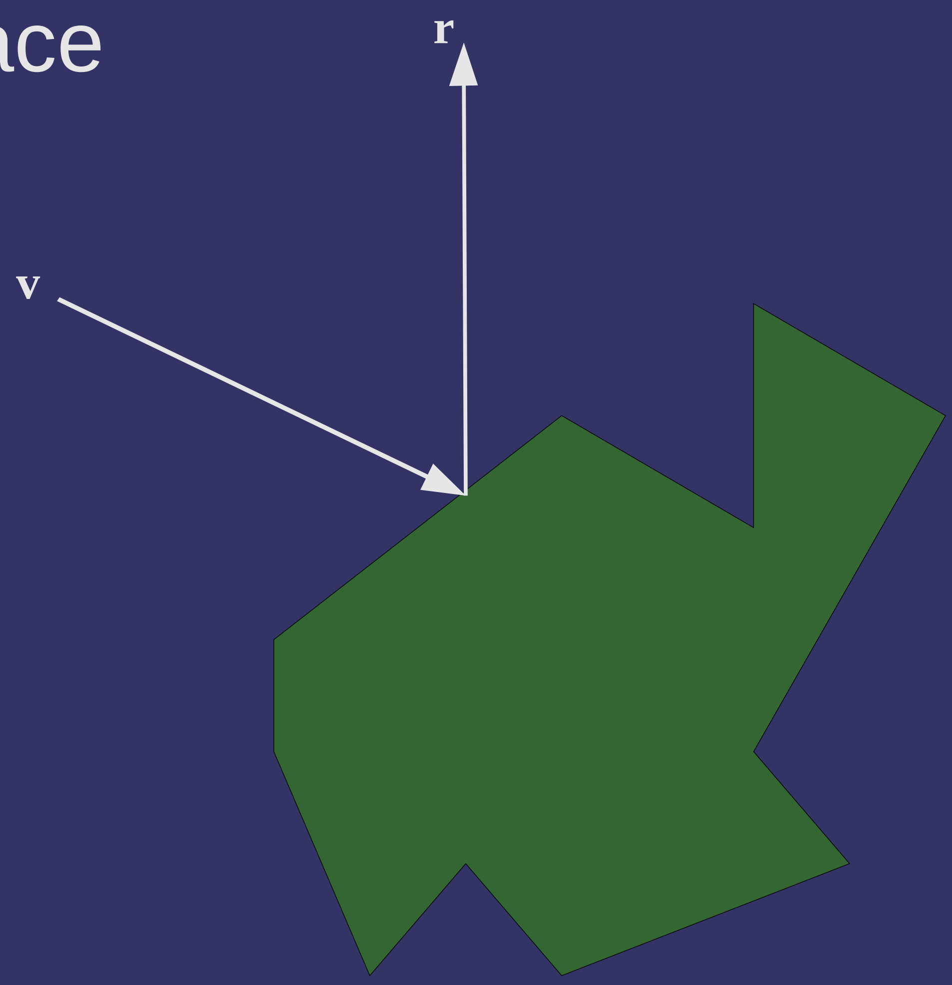
- Simulate reflections of the environment using a texture and texture coordinate calculations
  - Can either be called “environment mapping” or “reflection mapping”

Reflection Mapping

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

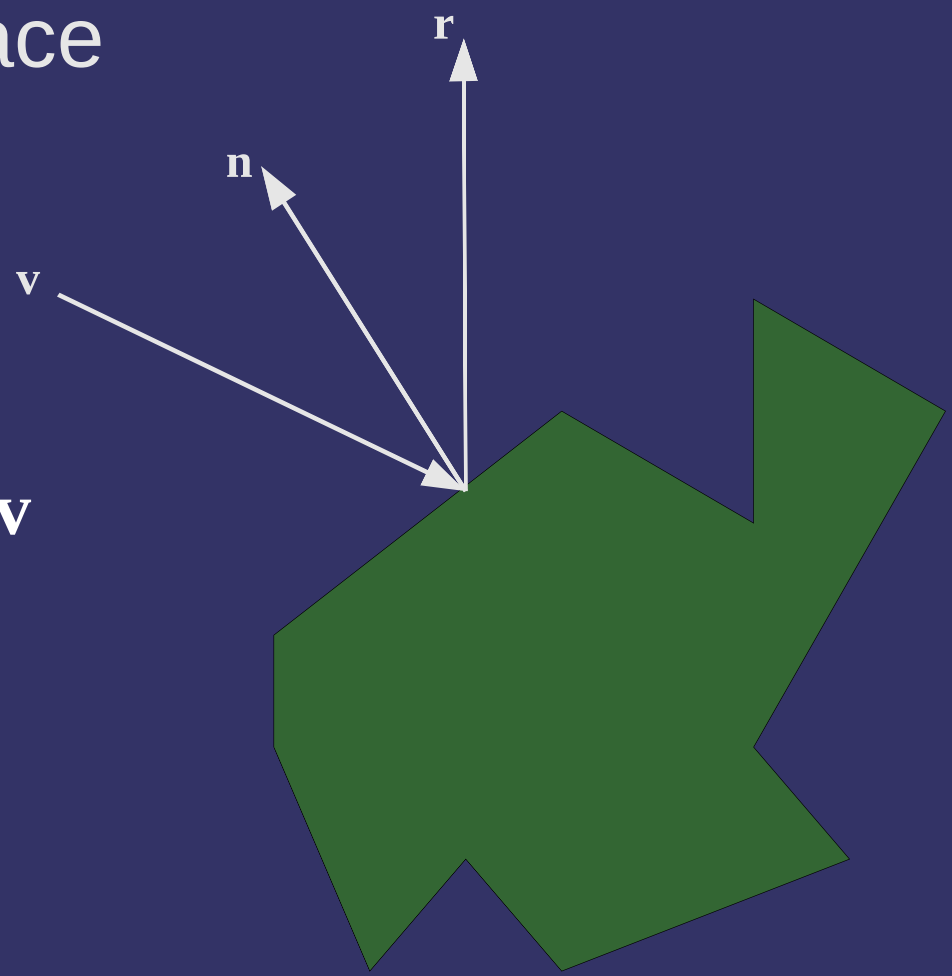
Reflection Mapping

- Calculate the reflection vector and map to texture space



Reflection Mapping

- Calculate the reflection vector and map to texture space



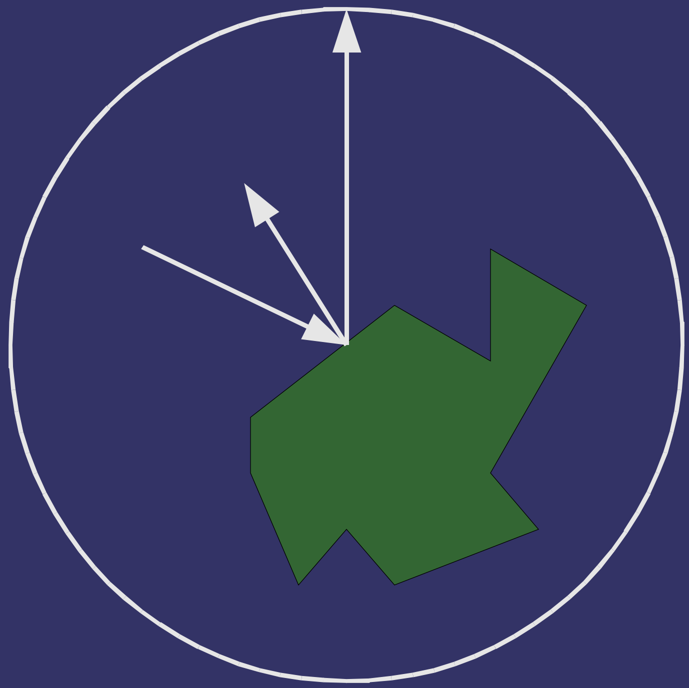
$$\mathbf{r} = \frac{2(\mathbf{n} \cdot -\mathbf{v})}{|\mathbf{n}||\mathbf{v}|}\mathbf{n} + \mathbf{v}$$

Reflection Mapping – Cylindrical

- *Exactly* like cylindrical projection
  - Use the reflection vector instead of the position

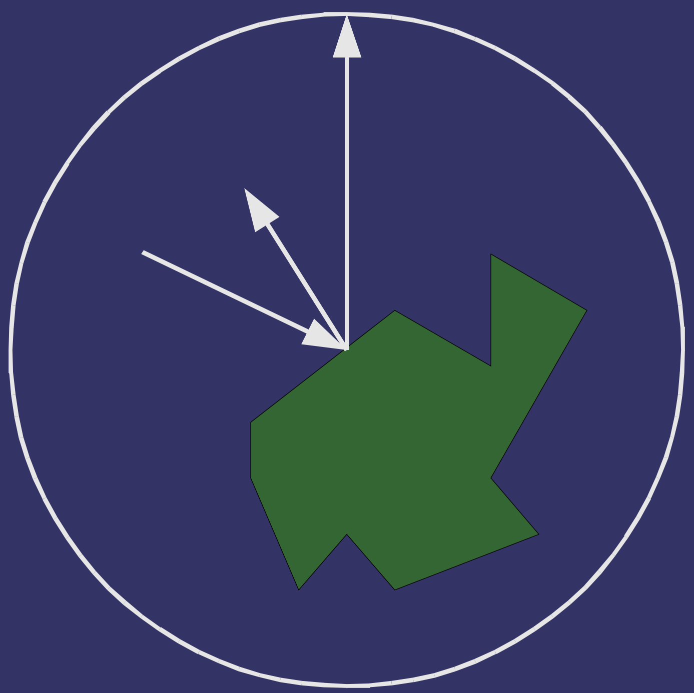
$$s = \frac{\operatorname{atan}(\mathbf{r}_x / \mathbf{r}_z)}{2\pi}$$

$$t = \mathbf{r}_y$$



Reflection Mapping – Cylindrical

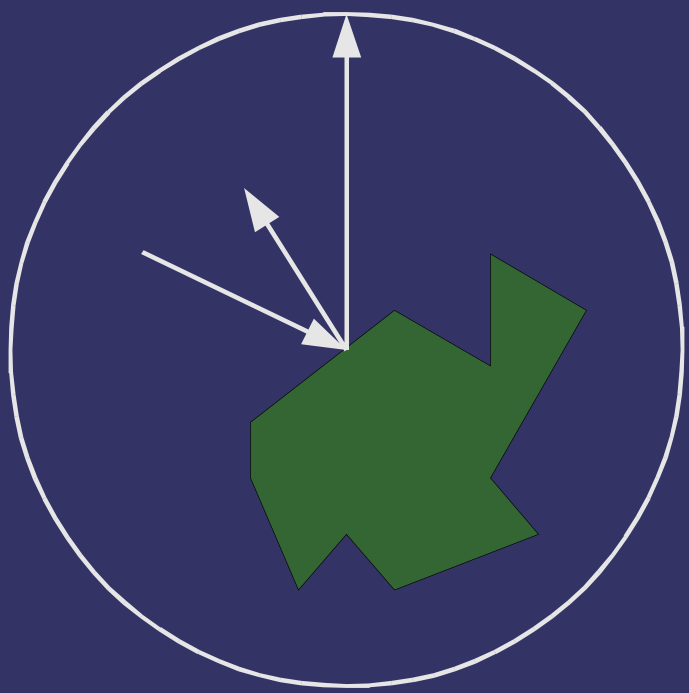
- Pros:
  - Easy to implement
  - Easy to get source images
  - Only one texture image
- Cons:
  - Distortion increases away from horizon
  - Can't reflect sky or ground (i.e., $\mathbf{r} = (0, \pm 1, 0)$)



30-November-2011

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., $\mathbf{r} = (0, \pm 1, 0)$)



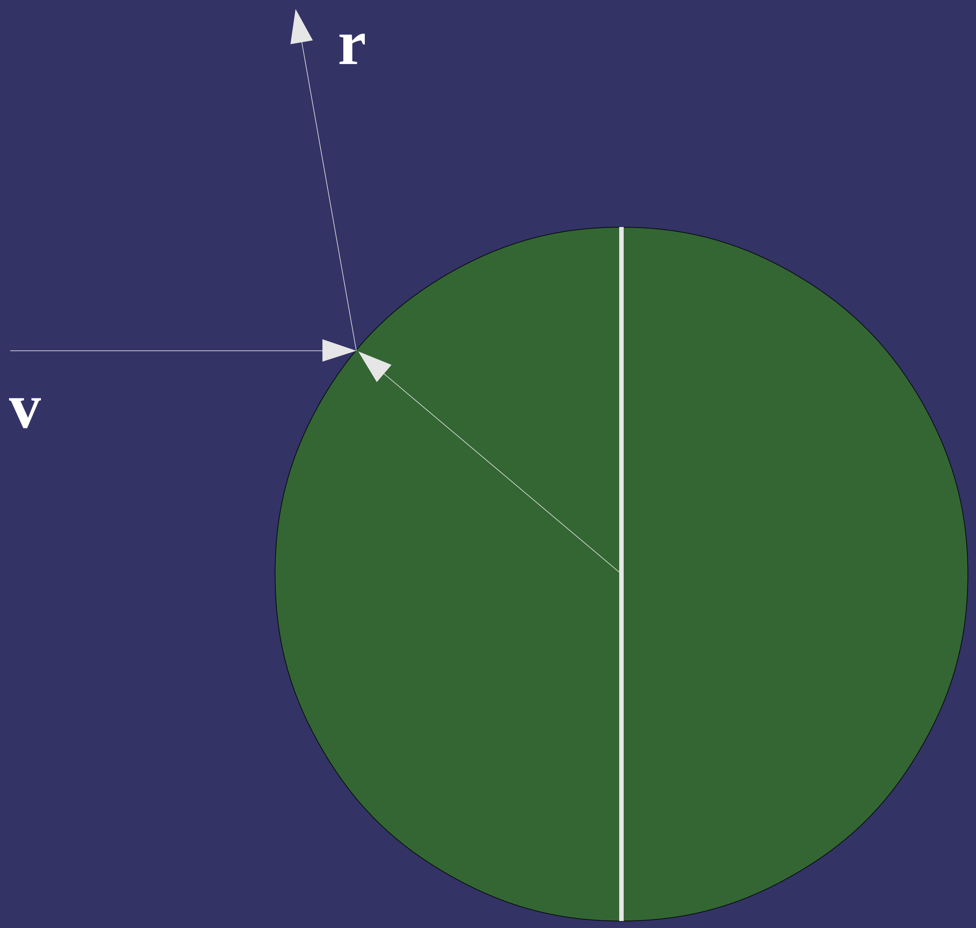
Caused by arctangent!

$$\lim_{z \to 0} \arctan(x/z) = \pm\infty$$

and has a discontinuity when $z = 0$.

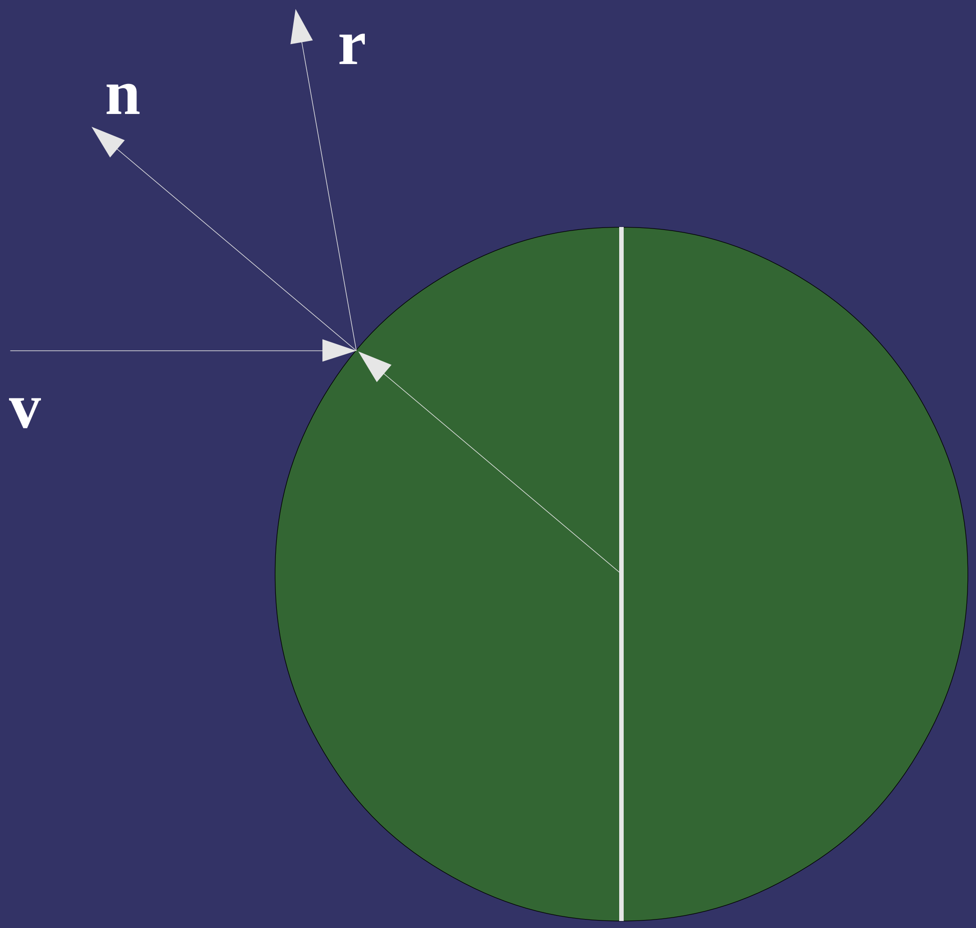
30-November-2011

Reflection Mapping – Hemispherical



- Find location with matching infinite view vector and **r**

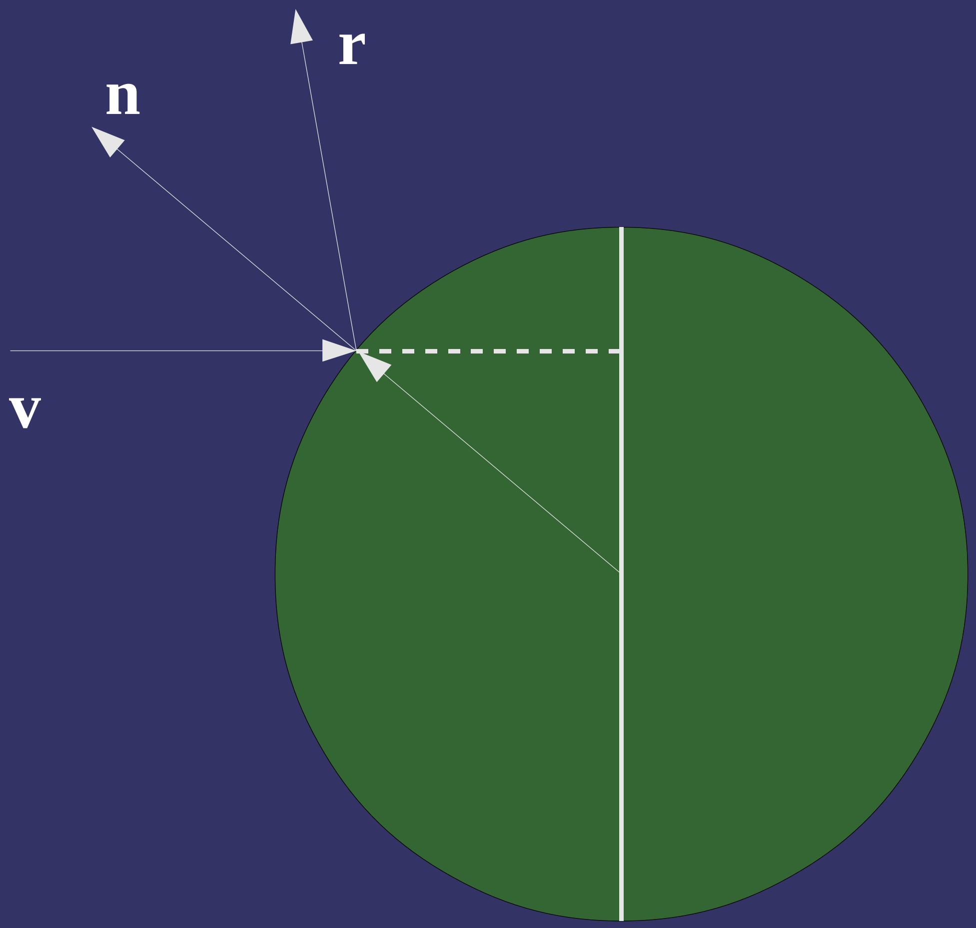
Reflection Mapping – Hemispherical



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

30-November-2011

Reflection Mapping – Hemispherical

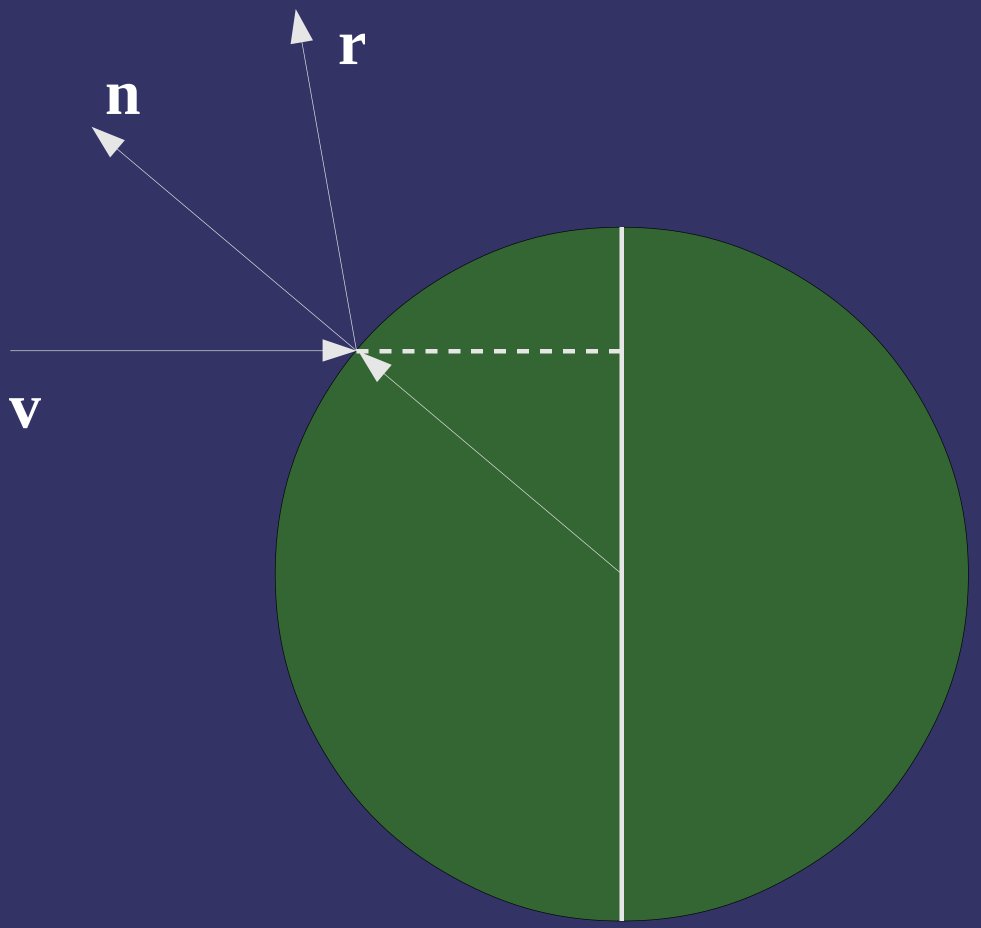


- Find location with matching infinite view vector and $\mathbf{r}$
  - This is the normal of the sphere at that location
  - Texture coordinate is the projection of this vector onto the image

$$s = \frac{\mathbf{r}_x}{\sqrt{2(\mathbf{r}_z + 1)}}$$

$$t = \frac{\mathbf{r}_y}{\sqrt{2(\mathbf{r}_z + 1)}}$$

Reflection Mapping – Hemispherical



- Find location with matching infinite view vector and $\mathbf{r}$
  - This is the normal of the sphere at that location
  - Texture coordinate is the projection of this vector onto the image

Range [-1, 1]

$$s = \frac{\mathbf{r}_x}{\sqrt{2(\mathbf{r}_z + 1)}}$$

$$t = \frac{\mathbf{r}_y}{\sqrt{2(\mathbf{r}_z + 1)}}$$

Reflection Mapping – Hemispherical

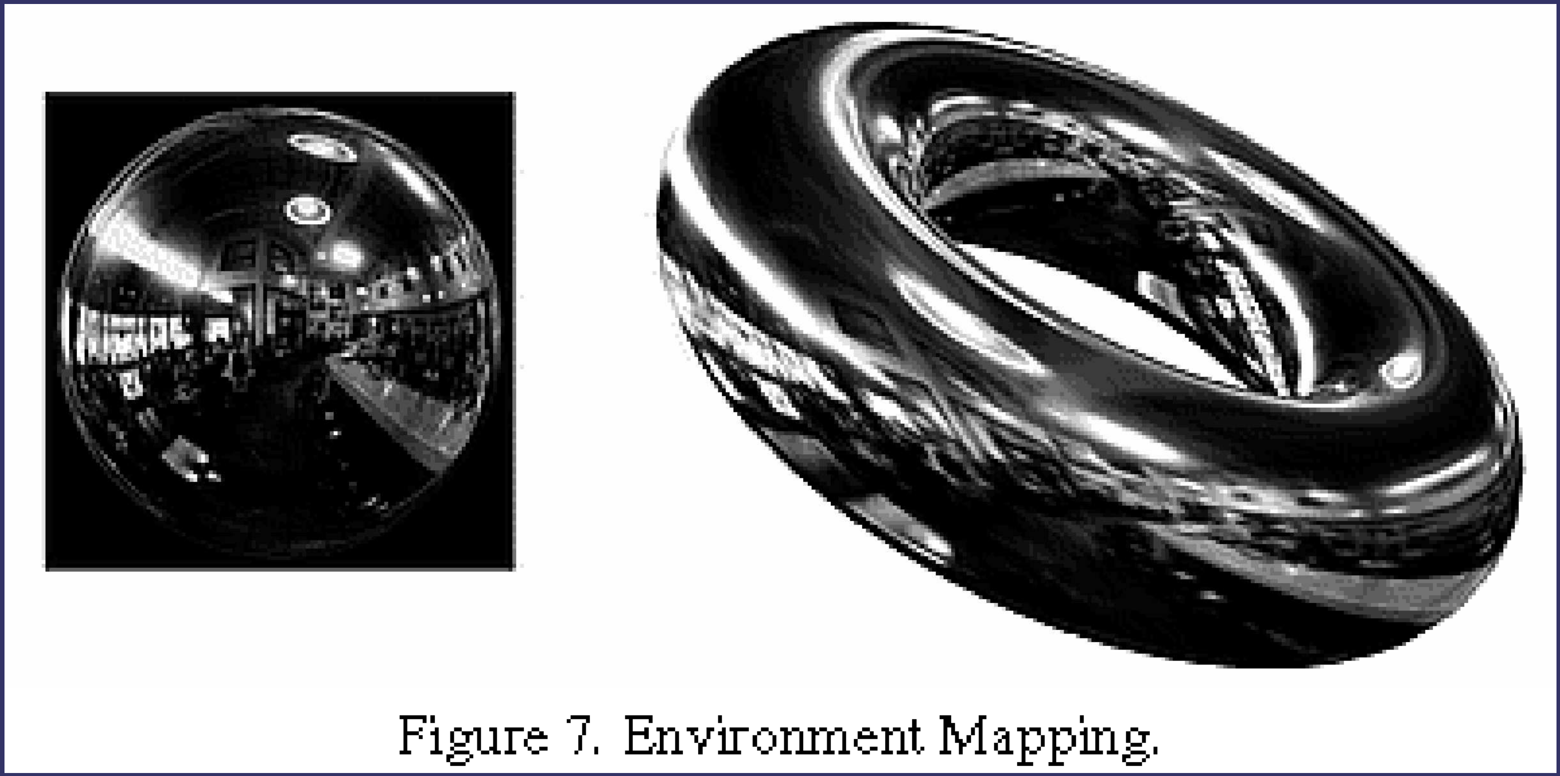
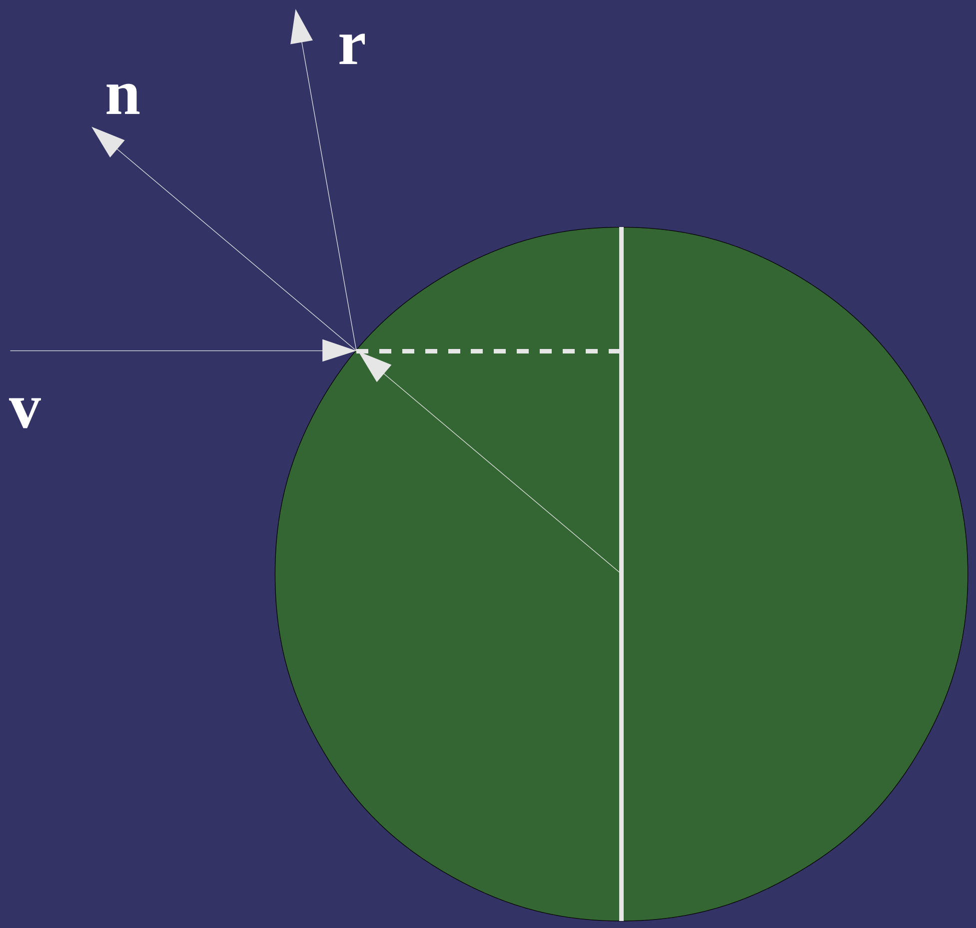


Image from http://www.graficaobscura.com/texmap/index.html

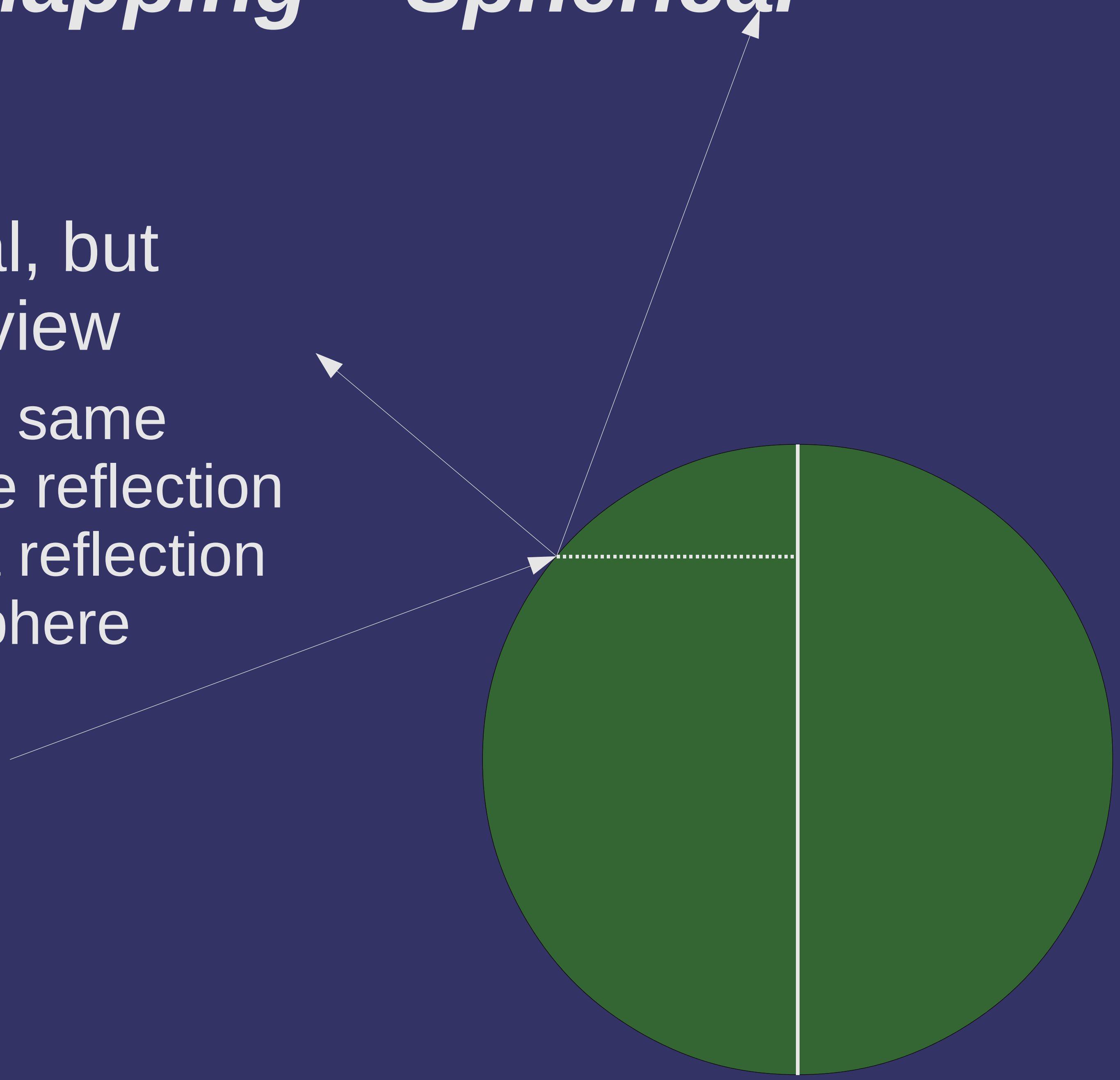
Reflection Mapping – Hemispherical



- 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

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



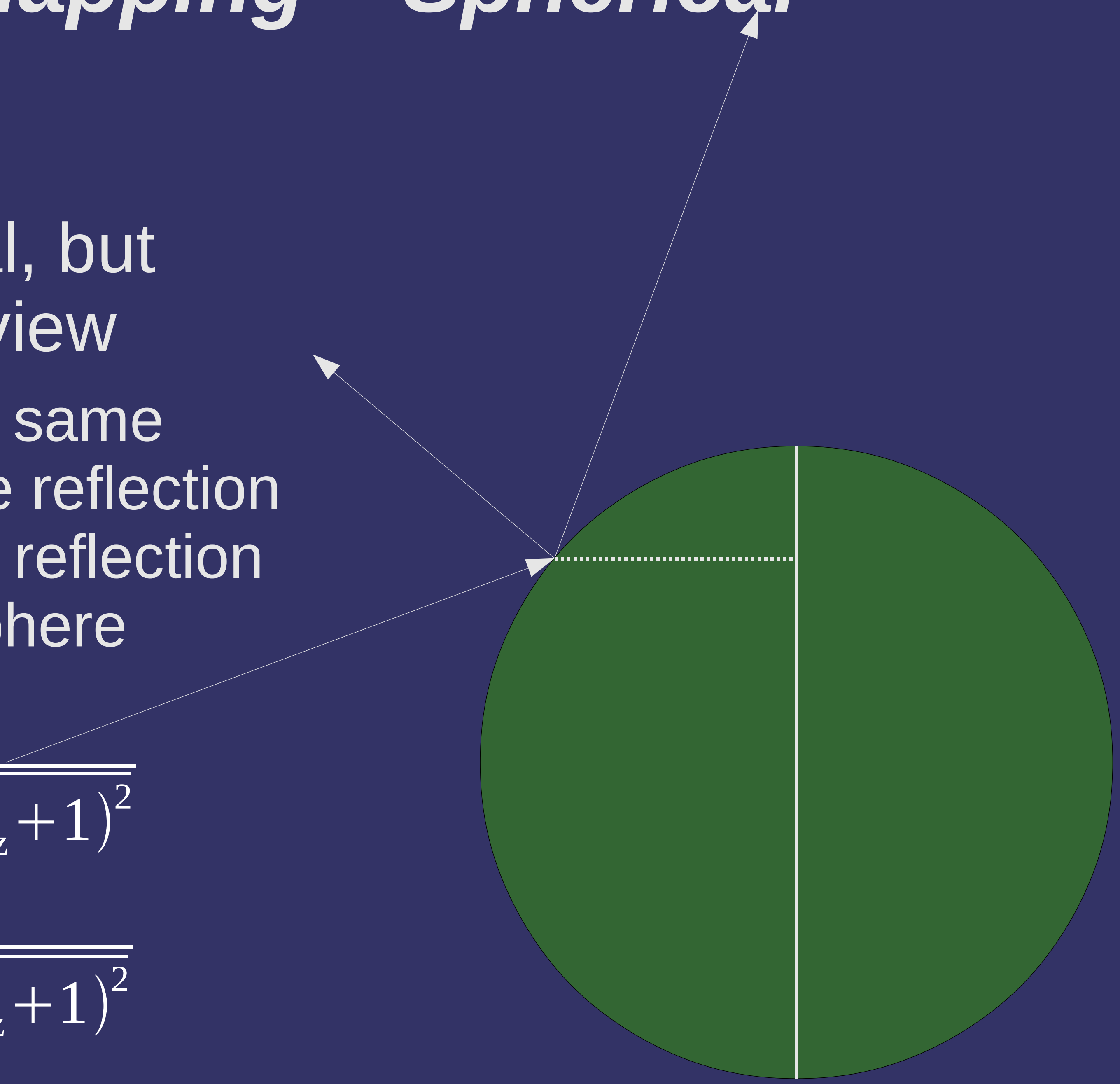
30-November-2011

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

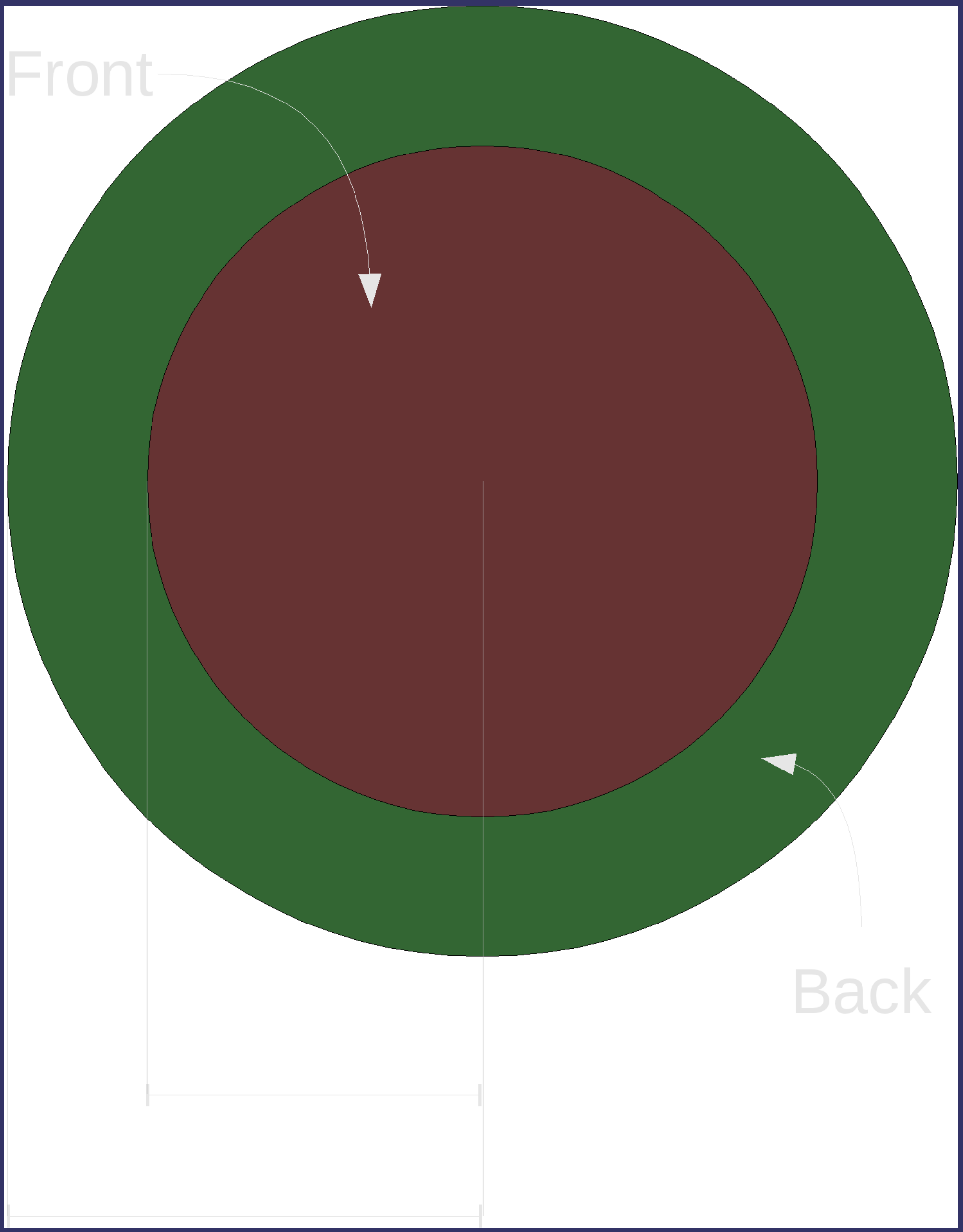
$$s = \frac{\mathbf{r}_x}{\sqrt{\mathbf{r}_x^2 + \mathbf{r}_y^2 + (\mathbf{r}_z + 1)^2}}$$

$$t = \frac{\mathbf{r}_y}{\sqrt{\mathbf{r}_x^2 + \mathbf{r}_y^2 + (\mathbf{r}_z + 1)^2}}$$



Reflection Mapping – Spherical

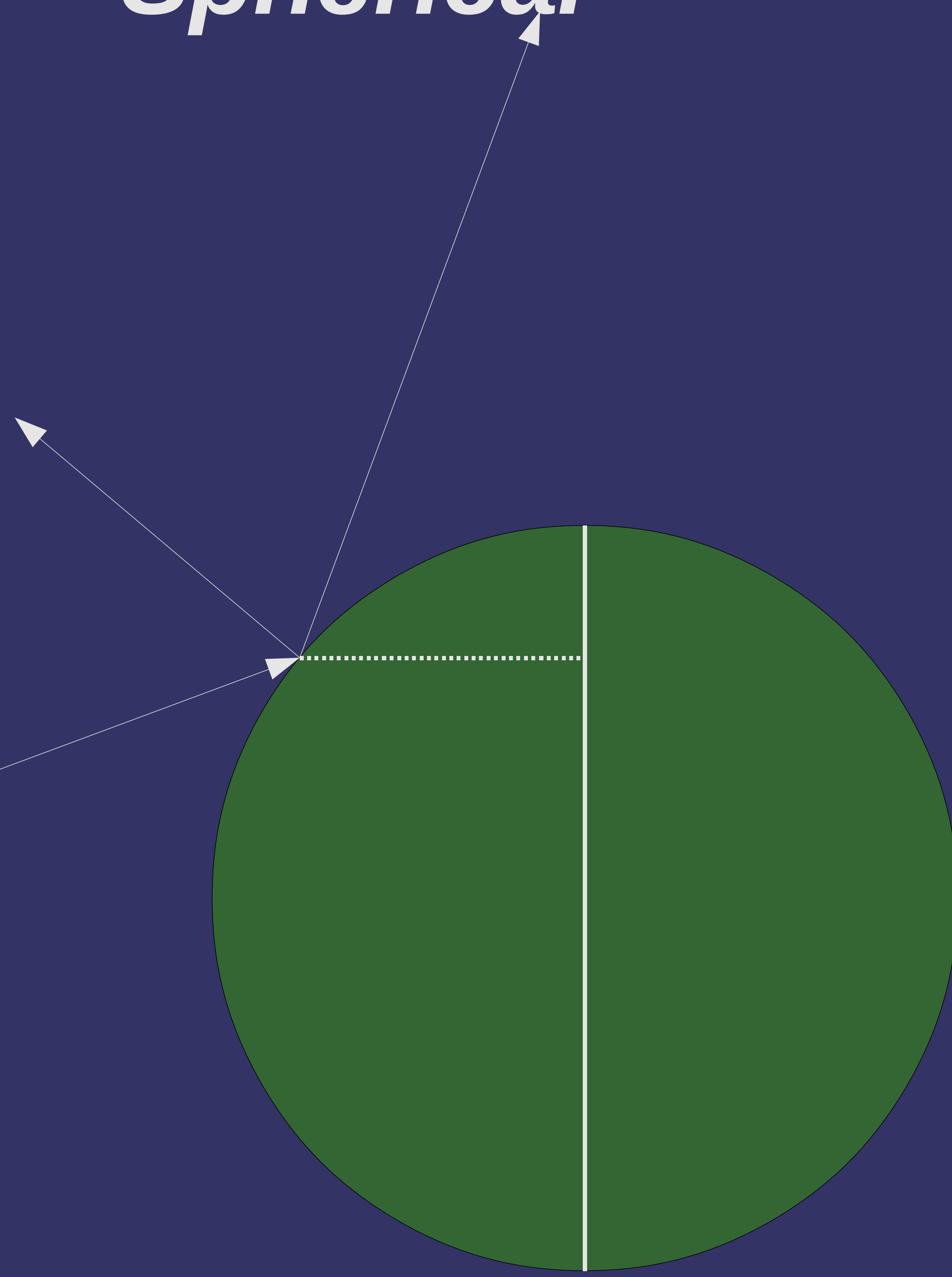
- Single image for full 360° view



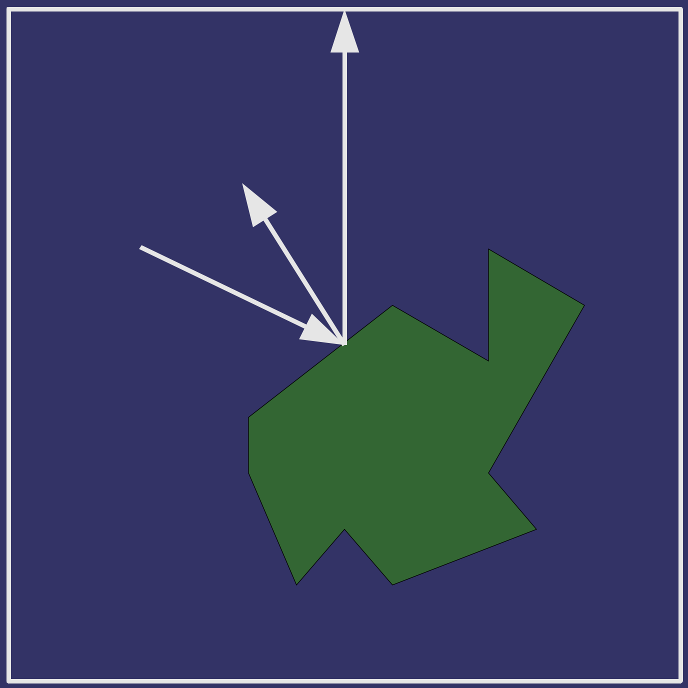
30-November-2011

Reflection Mapping – Spherical

- Pros:
  - Easy to implement
  - Only one texture
  - Local viewer and view independent
- Cons:
  - Distortion increases as $\mathbf{r}$ diverges from $\mathbf{v}$
  - Difficult to get source images
  - Difficult to render to reflection map

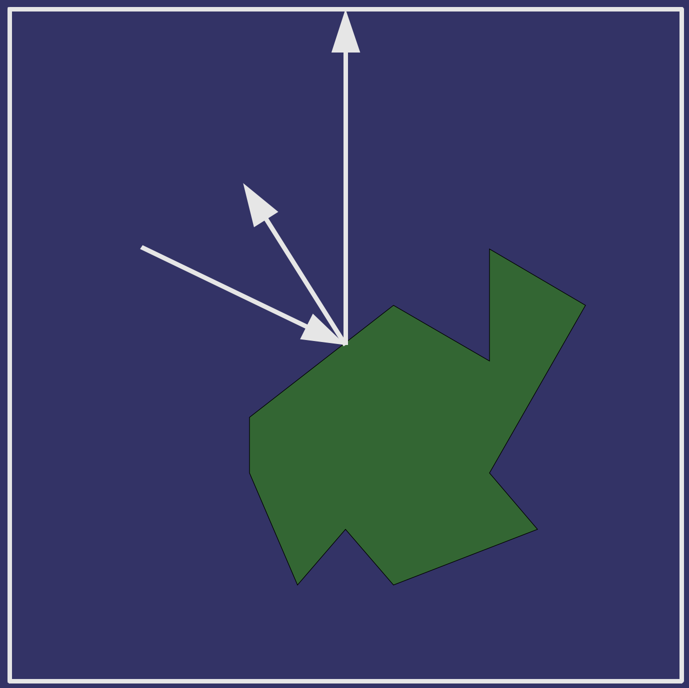


Reflection Mapping – Cube



- Extend $\mathbf{r}$ to intersect unit cube surrounding point

Reflection Mapping – Cube



- Pros:
  - Trivial to implement
  - Easy to render to reflection map
- Cons:
  - More difficult to get source images
  - Discontinuities at cube-face boundaries
  - Have to load multiple images

Reflection Mapping – Cube

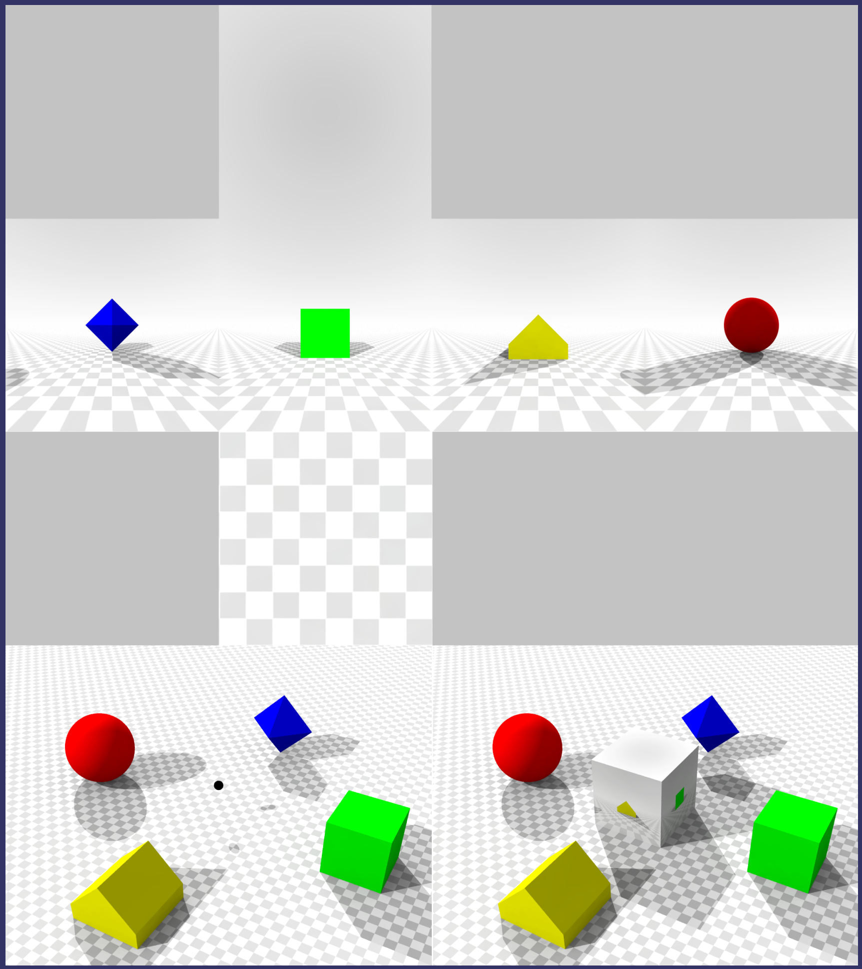


Image from http://en.wikipedia.org/wiki/File:Panorama_cube_map.png

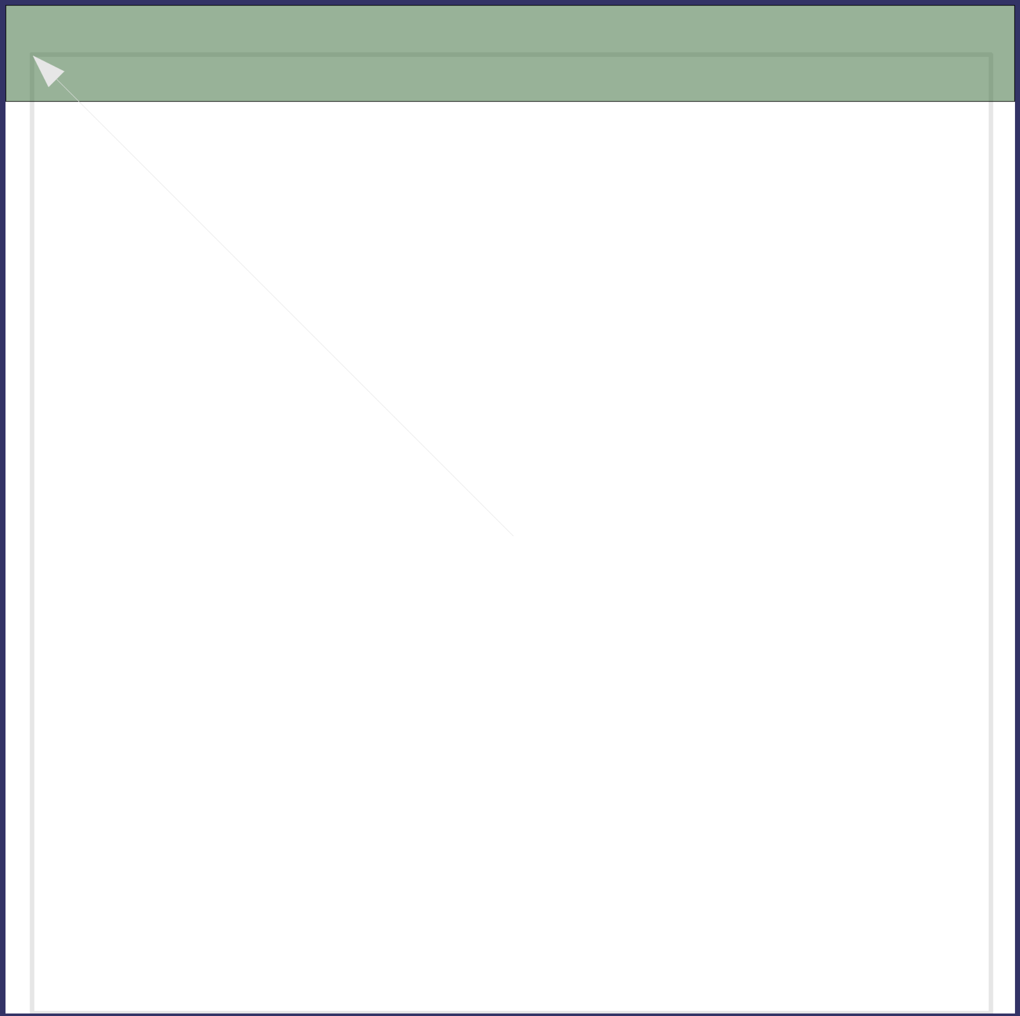
30-November-2011

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

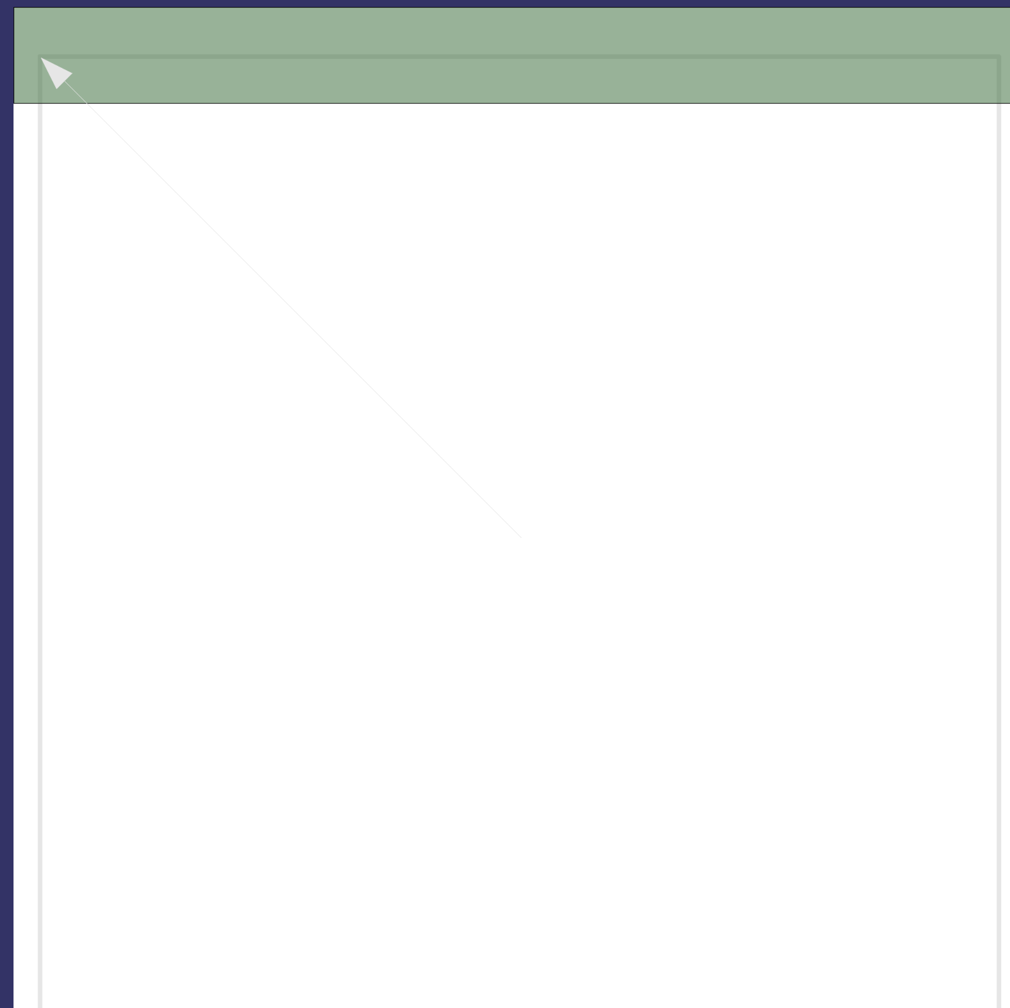
Cube Maps

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



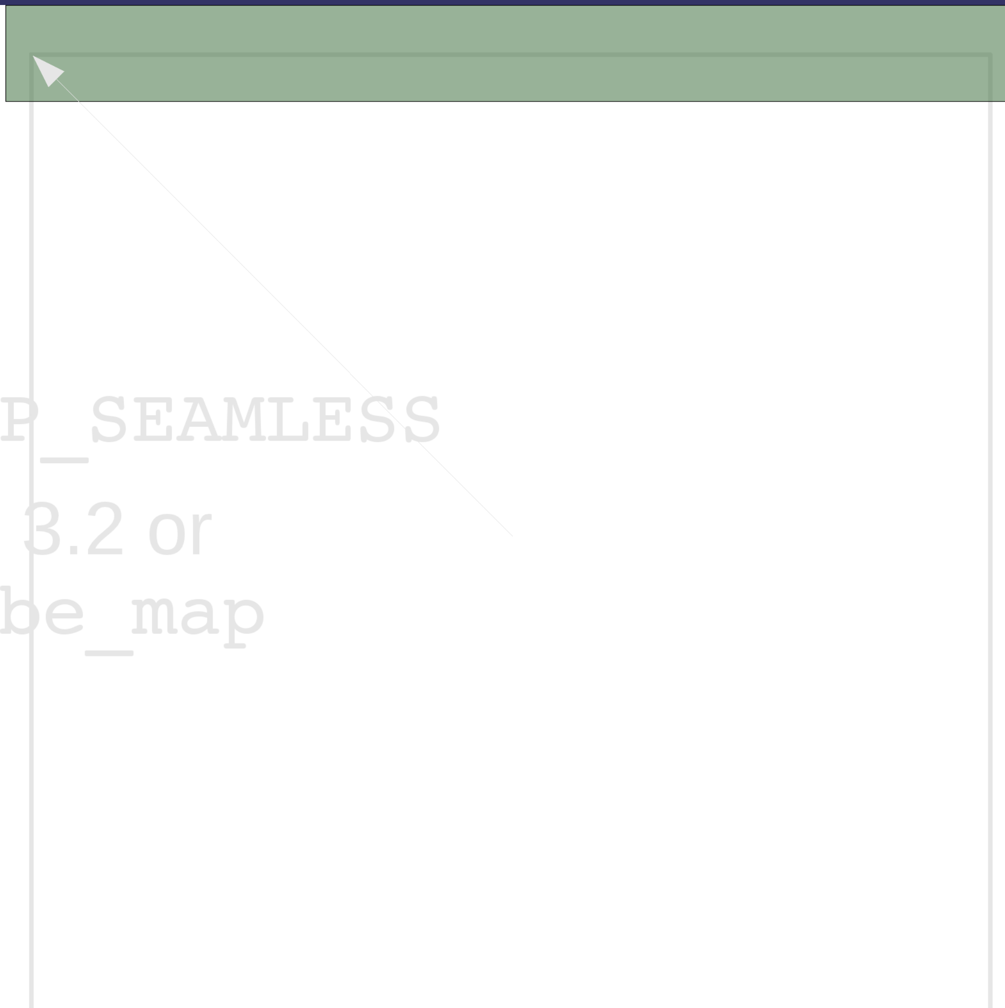
Cube Maps

- 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!



Cube Maps

- Newer hardware can sample from two cube map faces
  - Global enable `GL_TEXTURE_CUBE_MAP_SEAMLESS`
  - Requires either OpenGL 3.2 or `GL_ARB_seamless_cube_map`



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`

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

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.

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 several shadow techniques
    - You'll have to wait until VGP353

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?

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

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

Projective Texturing

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

Projective Texturing

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

$$s = (x/2.0) + 0.5$$
$$t = (y/2.0) + 0.5$$

Projective Texturing

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

$$s = (x / 2.0) + 0.5$$
$$t = (y / 2.0) + 0.5$$

Scale (2.0) — Translation (0.5)

Projective Texturing

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

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

Projective Texturing

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

Projective Texturing

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

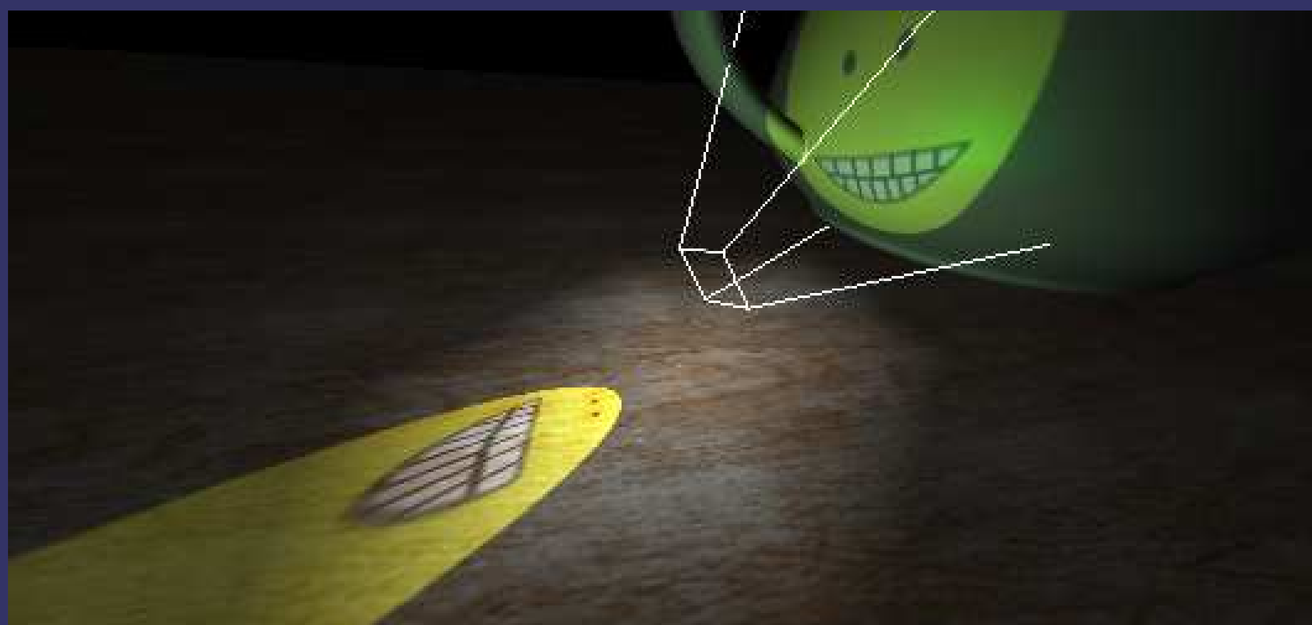


Figure 4. Projective texture mapping produces a reverse projection as well.

Image from [Everitt 01]

Projective Texturing

- How can anti-projections be eliminated?

Projective Texturing

- How can anti-projections be eliminated?
  - Detect the -z case and don't use the texture

```
color = (tc.z < 0.0)
    ? vec4(0.0) : texture2DProj(tex, tc);
```

  - Clamp z at 0 or ϵ

```
tc.z = max(tc.z, 0.0);
color = texture2DProj(tex, tc);
```

References

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

Everitt, Cass. 2001. "Projective Texture Mapping." NVIDIA Corporation. http://developer.nvidia.com/object/Projective_Texture_Mapping.html

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

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

Cost of State Changes

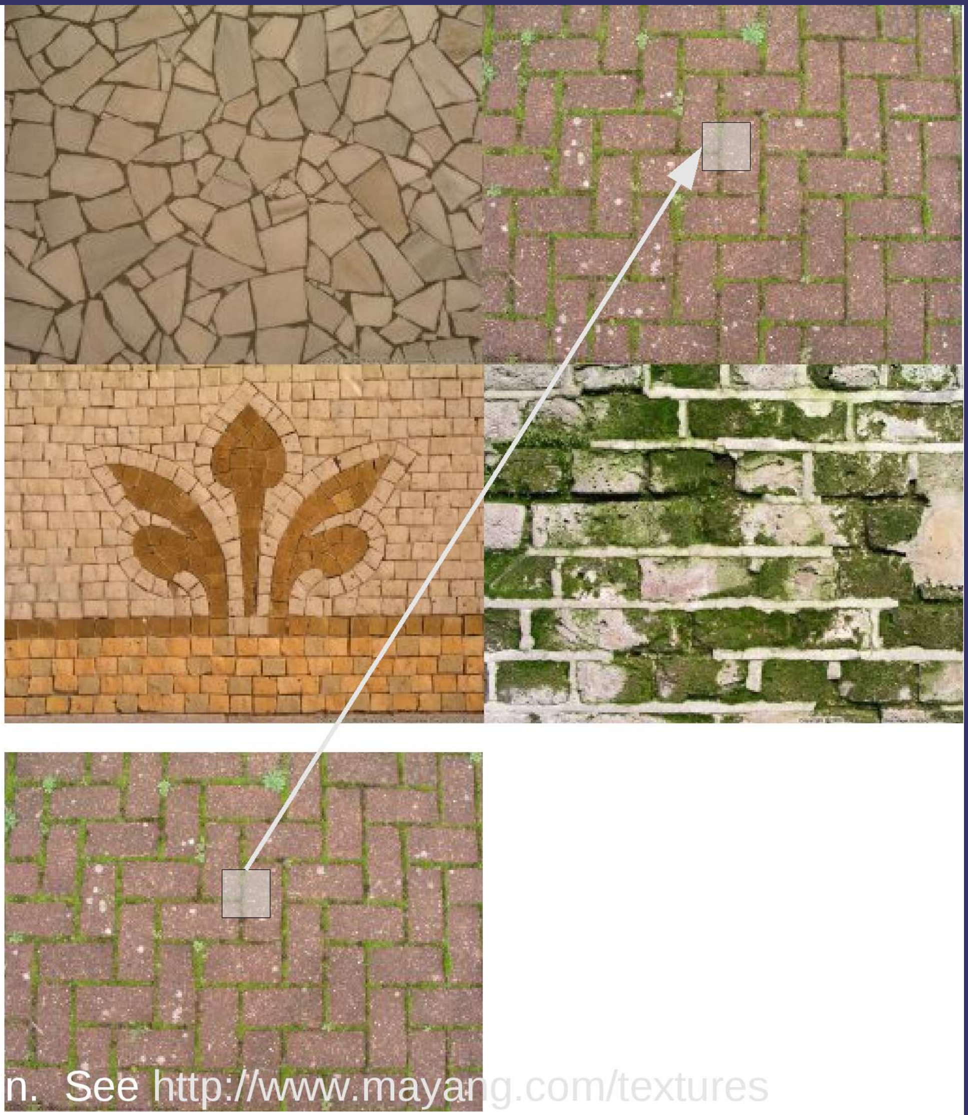
- Most common strategy to reduce state changes is sorting
  - Objects are sorted by common state and rendered in batches
  - This is a hassle and may not always be possible

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*

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



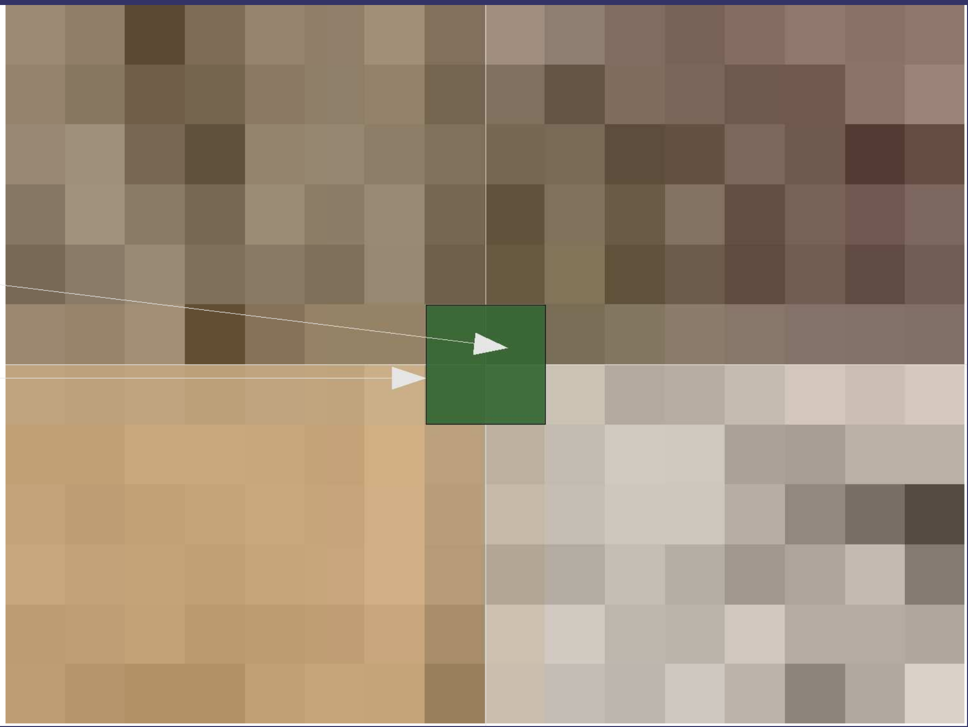
n. See http://www.mayang.com/textures

Texture Atlases

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

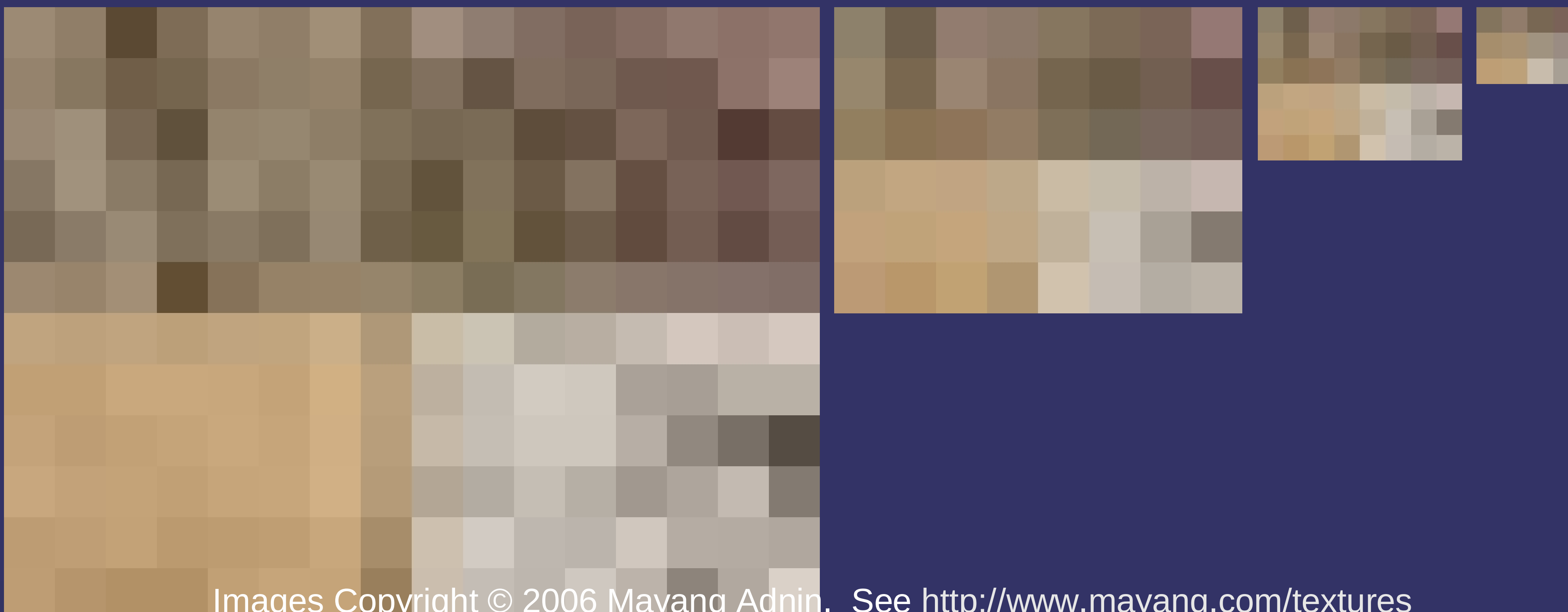
Texture Atlases

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



Texture Atlases

- Care must also be taken with mipmapping



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

Texture Atlases

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

References

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

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

Compression

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

Compression

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

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

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

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

Texture Compression

- What's the trade-off?

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

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`

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.

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`

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);
```

Texture Compression

- Read back a compressed texture:

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

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

References

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

Next week...

- Framebuffer blending
  - Alpha blending
  - Multipass rendering
  - Compositing
- Stencil buffer
- Quiz #4

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.