Interactive Computer Graphics

CS 418 – Spring 2011

MP3 Teapot Contest

TA: Gong Chen

Email: gchen10 at illinois.edu

Office Hours

Location: 1322 Siebel Center

Time: Monday 4pm-5pm

Today's Topics

- Sphere Maps
- Cube Maps
- Other sample texture mappings
- Fragment Shader Environment Mapping
- https://www.youtube.com/watch?v=H1ogRfMsc64&feature=BFa&list=PL_Q44fZ-Vy_ZMwFRxRHAKey7JwVYlt4DL

Sphere Maps

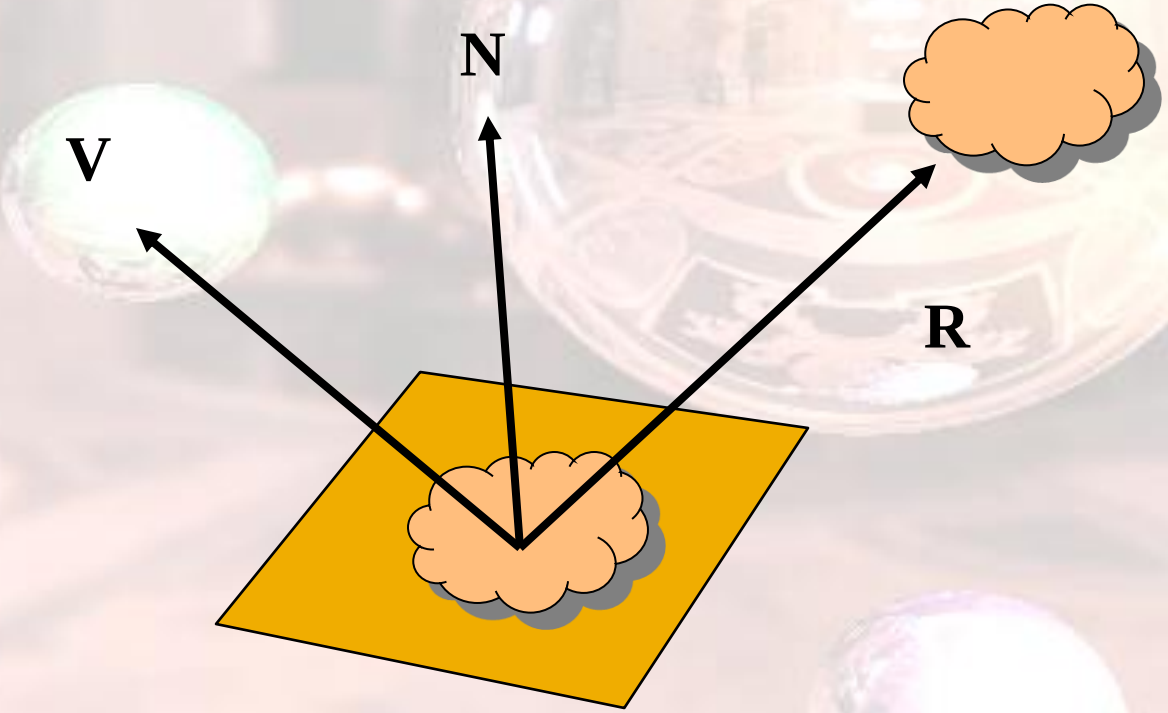
- Not as accurate as real life or as a Cube Map
- But whole lot faster

- Using:
  - sphere map
  - background image

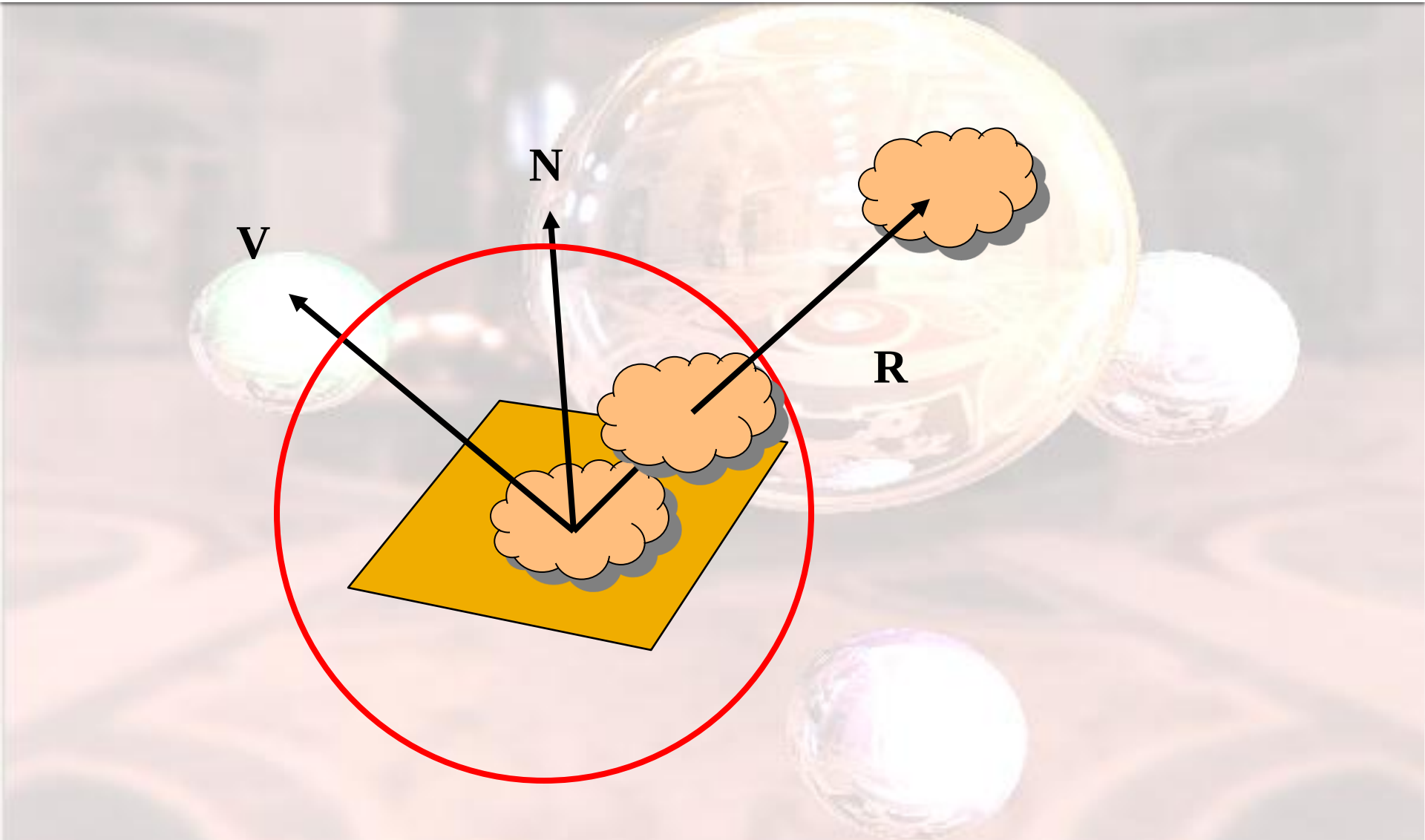
Creating a Sphere Map In Photoshop:

- use a picture of the environment you want to map onto the sphere.
- create a new PSD (Photoshop Format) image
- paste a copy of the environment image into the new image.
- resize the image so that the it's dimensions are a power of 2.
- from the filter menu, select distort and apply a spherize modifier. (in normal sphere maps the outer area will be blackened out, but it doesn't really matter).
- save a copy of the image as a .BMP

Reflecting the Environment



Mapping to a Sphere



Issues with Sphere Maps

- Must assume environment is very far from object (equivalent to the difference between near and distant lights)
- Object cannot be concave (no self reflections possible)
- No reflections between objects
- Need a reflection map for each object
- Need a new map if viewer moves

Creating Sphere Map Texture

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

glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);

glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);

glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);

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

computeNormalsForYourObject();
drawYourObject();
```

Cube Maps

- GL_TEXTURE_CUBE_MAP
  - define 6 textures for a cube GL_TEXTURE_CUBE_MAP_*dir*_*axis*
    - *dir* (direction) is POSITIVE or NEGATIVE
    - *axis* is X, Y or Z
  - Ex. GL_TEXTURE_CUBE_MAP_POSITIVE_Z is the front face of the cube
  - Note : when creating the textures, the target should be GL_TEXTURE_CUBE_MAP_dir_axis instead of GL_TEXTURE_2D.

Advantages of Cube Maps

- Compared to Other Approaches
  - View-independent
    - Unlike sphere mapping
  - Uses a single texture unit
  - Entire texture resolution is sampled
  - Improved environment sampling

Creating The Cube Map

```
glBindTexture(GL_TEXTURE_CUBE_MAP_POSITIVE_X, cubeMap[0]);

glBindTexture(GL_TEXTURE_CUBE_MAP_NEGATIVE_X, cubeMap[1]);

glBindTexture(GL_TEXTURE_CUBE_MAP_POSITIVE_Y, cubeMap[2]);

glBindTexture(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y, cubeMap[3]);

glBindTexture(GL_TEXTURE_CUBE_MAP_POSITIVE_Z, cubeMap[4]);

glBindTexture(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z, cubeMap[5]);

glEnable(GL_TEXTURE_CUBE_MAP);
```

Reflection Map

- We want this environment to be reflected on our shapes/objects.
  - Use GL_REFLECTION_MAP
  - Enable GL_S, GL_T and GL_R with GL_REFLECTION_MAP
  - draw the object

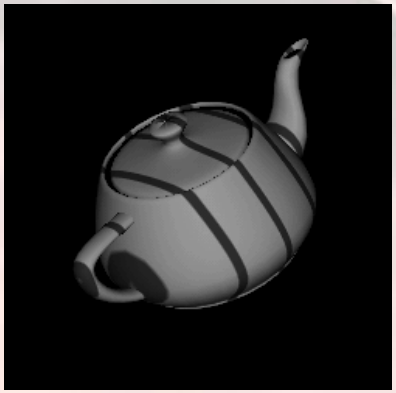
Reflection Mapping

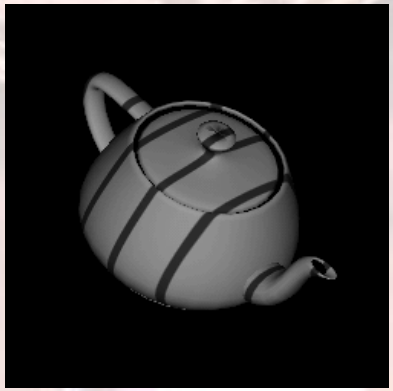
```
//GL_REFLECTION_MAP for s,t,r texture coordinates

glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
glTexGeni(GL_R, GL_TEXTURE_GEN_MODE,GL_REFLECTION_MAP);

glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
glEnable(GL_TEXTURE_GEN_R);

//Draw shape to aply reflection on it

glDisable(GL_TEXTURE_GEN_S);
glDisable(GL_TEXTURE_GEN_T);
glDisable(GL_TEXTURE_GEN_R);
```

Object Linear Mapping

- **Texture is "attached" to object**

```
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
glEnable(GL_TEXTURE_GEN_S);
```

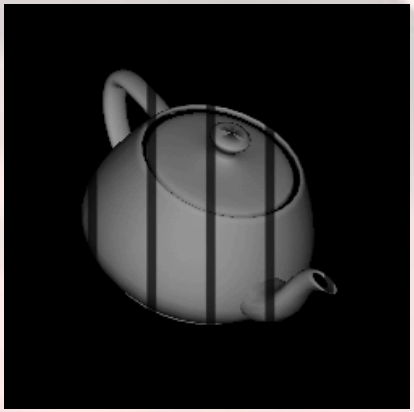


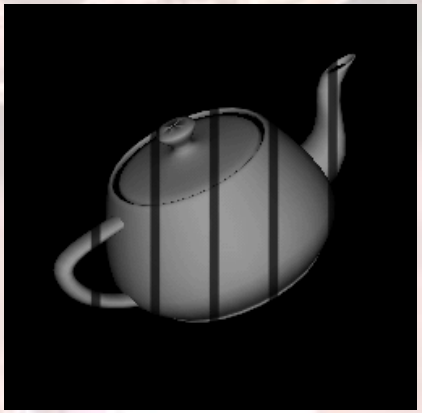


Eye Linear Mapping

- **Texture is "fixed" in eye space**

```
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
glEnable(GL_TEXTURE_GEN_S);
```





Environment Maps with Shaders

- Environment map usually computed in world coordinates which can differ from object coordinates because of modeling matrix
  - May have to keep track of modeling matrix and pass it shader as a uniform variable
- Can also use reflection map or refraction map (for example to simulate water)

Environment Map Vertex Shader

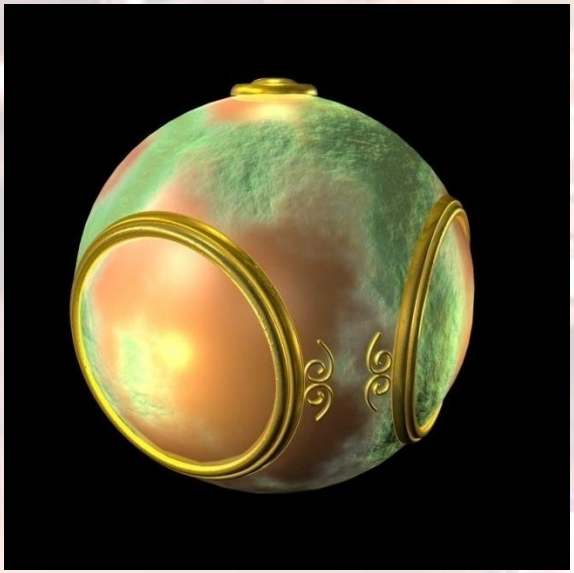
```
Varying vec3 reflectw;
uniform mat4 modelMat;
uniform mat3 invModelMat;
uniform vec4 eyew;
void main(void)
{
   vec4 positionw = modelMat*gl_Vertex;
   vec3 normw = normalize(gl_Normal*invModelMat.xyz);
   vec3 lightw = normalize(eyew.xyz-positionw.xyz);
   reflectw= reflect(normw, eyew);
   gl_Position = gl_ModelViewProjectionMatrix*gl_Vertex;
}
```

Environment Map Fragment Shader

```
/* fragment shader for reflection map */
varying vec3 reflectw;
uniform samplerCube MyMap;
void main(void)
{
    gl_FragColor = textureCube(myMap, reflectw);
}
```

Bump Mapping

- Perturb normal for each fragment
- Store perturbation as textures



Normalization Maps

- Cube maps can be viewed as lookup tables 1-4 dimensional variables
- Vector from origin is pointer into table
- Example: store normalized value of vector in the map
  - Same for all points on that vector
  - Use "normalization map" instead of normalization function
  - Lookup replaces sqrt, mults and adds

Pointers to useful tutorials

- Texture
  - http://nehe.gamedev.net/tutorial/lesson_06_texturing_update/47002/
  - http://nehe.gamedev.net/tutorial/texture_mapping/12038/
- Environment Mapping
  - http://nehe.gamedev.net/tutorial/sphere_mapping_quadrics_in_opengl/15005/
- Bumpmap
  - http://www.paulsprojects.net/tutorials/simplebump/simplebump.html
- GLSL Shader Help
  - Lighthouse3d.com/tutorials