Radial/Equirectangular environment mapping

What is a Radial/Equirectangular texture?

A standard 360 degree environment map is typically done with a cube map texture that looks like:

A Radial/Equirectangular texture is a 2D texture that looks like:

These sorts of textures are encountered in areas like laser scanning and planet texturing – where the texture has been peeled from around a sphere.

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

http://www.youtube.com/watch?v=3Ic5ZIf74Ls

Google maps street view API uses this projection:

http://notlion.github.com/streetview-stereographic/

Cameras that capture images in this format:

http://www.ptgrey.com/products/spherical.asp

Why use Radial/Equirectangular textures?

Cube maps are widely supported in hardware and with the ability to filter along cube map seams there is even less reason to use anything else. However, if you are targeting hardware that does not support this (or supports it in an inconsistent manner) Radial/equirectangular textures might be an option.

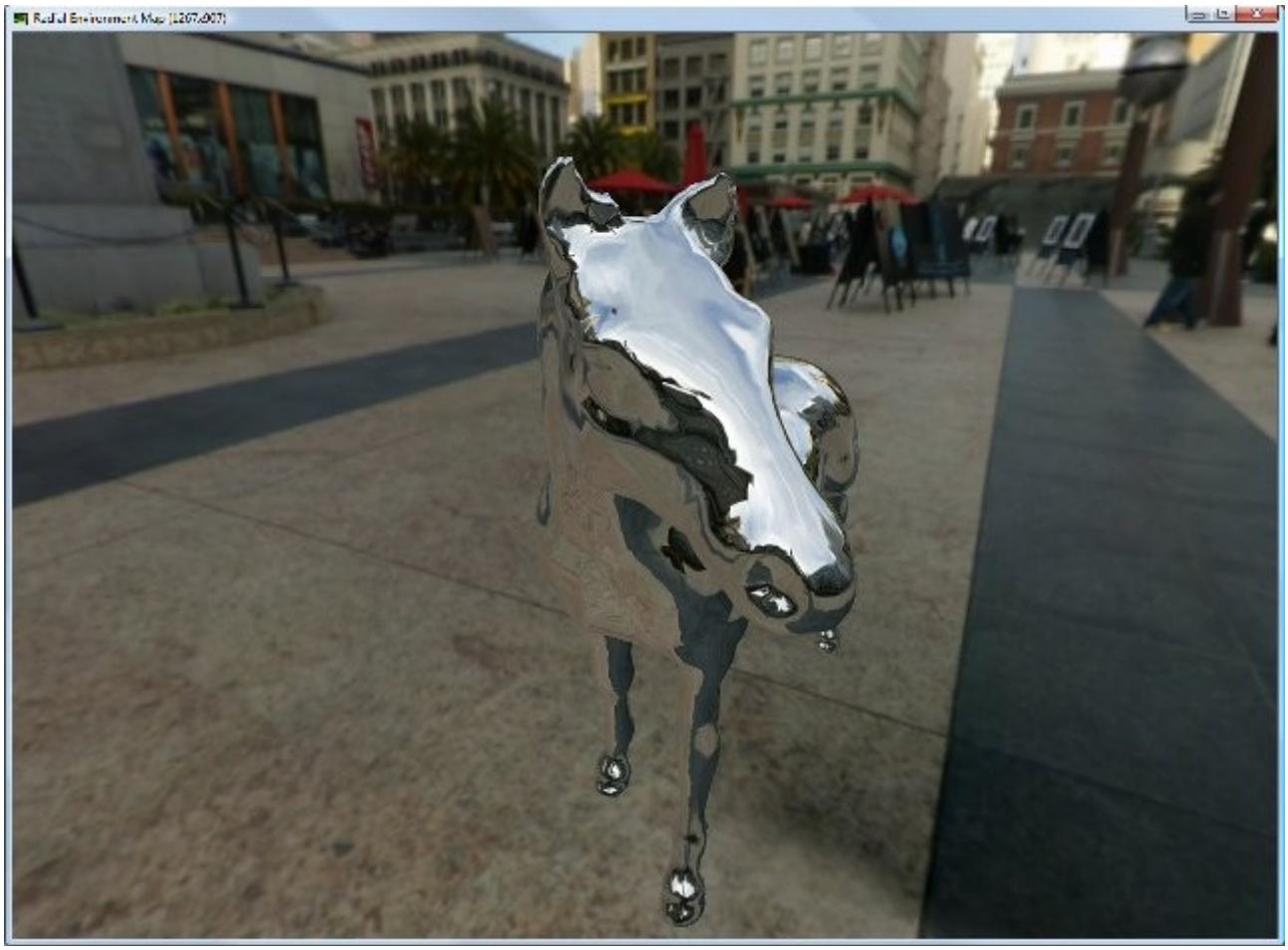
http://altdevblogaday.com/2012/03/03/seamless-cube-map-filtering/

http://forum.beyond3d.com/showthread.php?t=54527

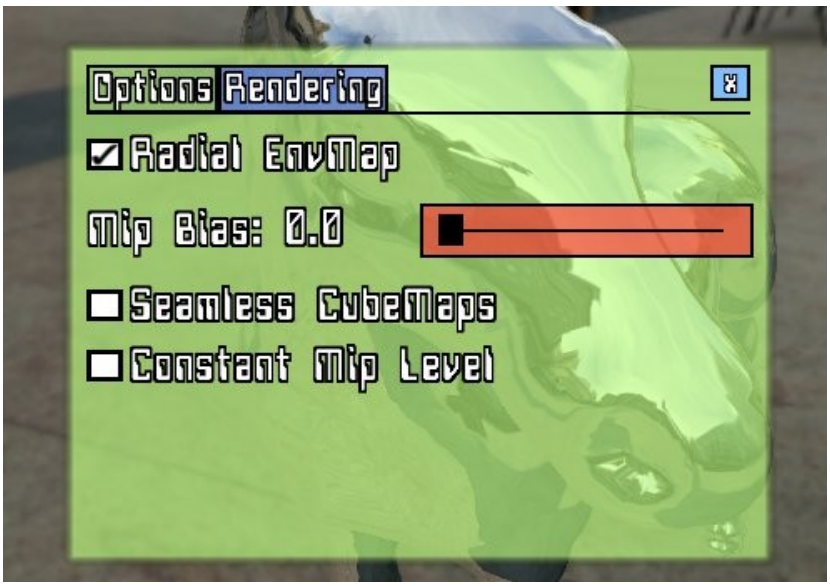
Radial/equirectangular textures also have the advantage of being a single image that might make management/editing easier.

Just like cube maps, care must be taken when generating mip-maps for Radial/equirectangular textures. If standard mip map generation is used, too much details is preserved in the top and bottom areas of the image. (Note: the demo does not take this into account)

Radial/Equirectangular texture demo



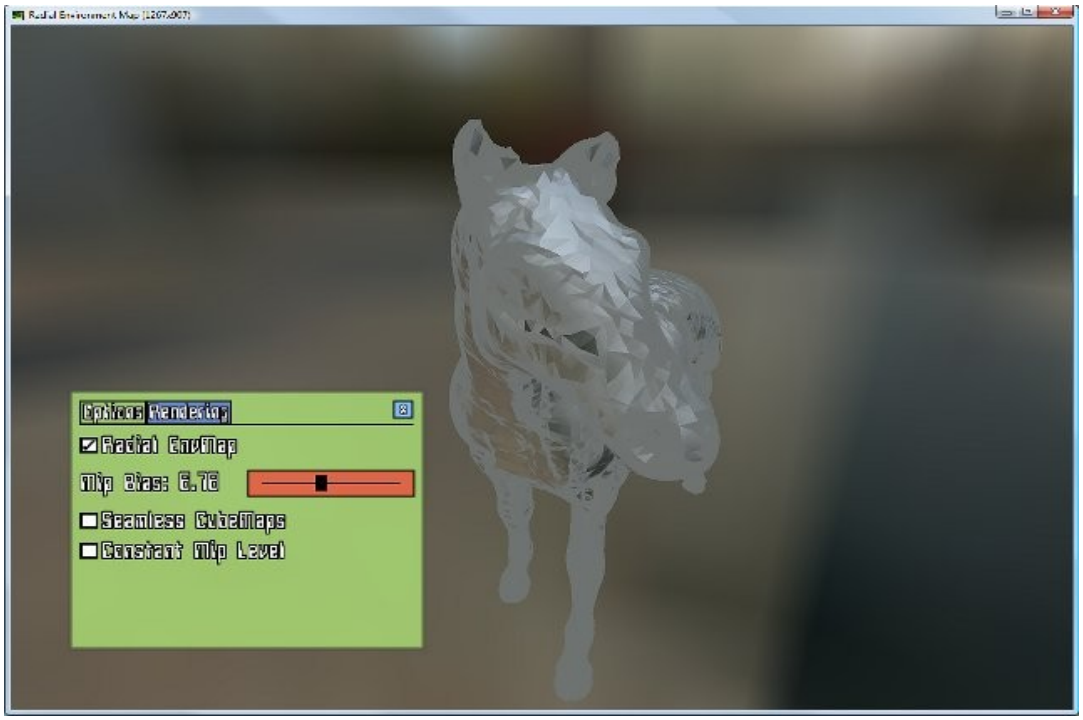
This demo demonstrates experiments in Radial/Equirectangular environment mapping where the horse and the background are rendered using an environment map. On startup the demo converts a cube map to a Radial/Equirectangular texture.



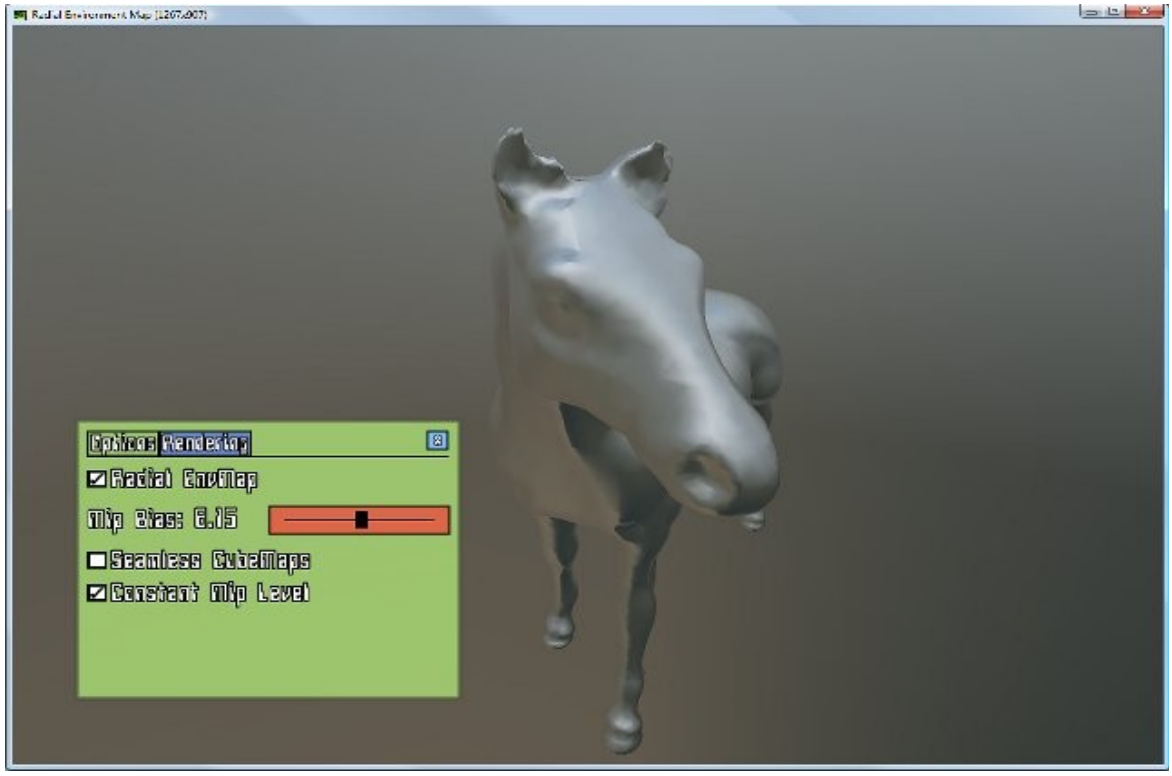
Press F1 to bring up the options menu to change options.(eg. Toggle between the cube map and the Radial/Equirectangular environment map.) Note: the radial map is lower resolution at 2048x1024 vs 6x1024x1024 for the cube map.

Side note: In the book "OpenGL Shading Language" (Orange book) there is some code in section 10.5 on rendering Equirectangular textures. This code does not work correctly as it "smears" the edges of the environment map. This shader code is provided and can be manually turned on by editing the *.shd files in the demo.

Environment map mip bias



Often when using an environment mapping you may want a softer reflection so add a mip bias to the texture lookup. However, this often has the effect of showing the triangle faces of the model as each surface shows a different mip level (even cube maps behave like this)



One way of solving this is to calculate a mip level to use over the entire object.

Shader Source

```
#define PI 3.141592653589793

vec4 RadialLookup(sampler2D a_radialTex, vec3 a_coords, float a_mipLevel)
{
  float r = length(a_coords);
  float lon = atan(a_coords.z, a_coords.x);
  float lat = acos(a_coords.y / r); // Remove divide if a_coords is normalized

  const vec2 rads = vec2(1.0 / (PI * 2.0), 1.0 / PI);
  vec2 sphereCoords = vec2(lon, lat) * rads;

  return textureLod(a_radialTex, sphereCoords, a_mipLevel);
}


float CalcMipLevel(vec3 a_coords)
{
  a_coords = normalize(a_coords); // May not be necessary if already normalized

  vec3 dx = dFdx( a_coords );
  vec3 dy = dFdy( a_coords );
  float d = max( dot( dx, dx ), dot( dy, dy ) );

  // Should technically calculate the arc length on the unit sphere
  // 256 seems to be a good value for the size of the texture we are using
  //float mipLevel = log2( sqrt(d) * 256.0);
  float mipLevel = 0.5 * log2(d) + 8.0;

  return mipLevel;
}
```