4.9

Rendering Distant Scenery with Skyboxes

Jason Shankel, Maxis

shankel@pobox.com

Consider a game that takes place on a Tibetan mountaintop. In the distance, we can see the ridge of the Himalayas, clouds in the sky, and a village down in the valley. Or, consider a game in deep space, with the Eagle and Orion nebulae shining from light-years away. Such imagery can heighten the beauty and sense of immersion of a 3D game.

Rendering distant scenery in 3D can be accomplished with skyboxes. This gem explains the principle of skyboxing and describes alternative means for rendering a skyboxed scene.

Basic Technique

The idea behind a skybox is very simple. Distant scenery is rendered onto six textures, each of which is applied to one side of a cube. The camera is placed at the center of the cube. The camera can rotate freely within, but never move from the center of this cube. When the scene is rendered, the images projected on the walls of the skybox give the impression of distant scenery much in the same way images projected onto the ceiling of a planetarium give the impression of the night sky (Figure 4.9.1).

Skybox Resolution

Ideally, one texel in die skybox object should map to one pixel on the screen. The following formula can be used to determine the ideal skybox resolution for a given screen resolution

$$skyboxRes = \frac{screenRes}{\tan.(fov/z)}$$

where *skyboxRes* is the resolution of one side of the skybox in texels, *screenRes* is the width of the screen in pixels, andyof is the angle of the horizontal field of view. For example, a scene with a 90-degree field of view running at 640x480 would have an ideal skybox resolution of 640 texels.

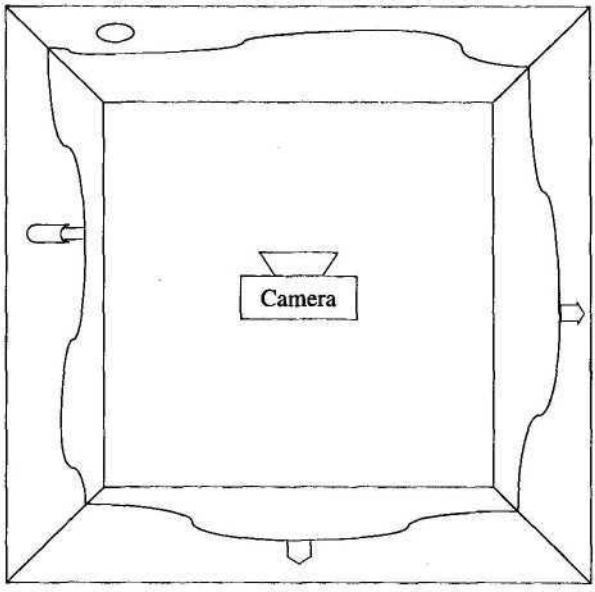

FIGURE 4.9.1 *Skybox as seen from above: distant terrain is rendered on the sides of the box, and the camera is placed at the center.*

Some 3D systems limit texture resolution to 256x256 texels. This means that the skybox textures may be stretched noticeably when rendered to the screen. For some applications, this stretching may be acceptable. Others may wish to subdivide each skybox face to increase texture resolution (Figure 4.9.2).

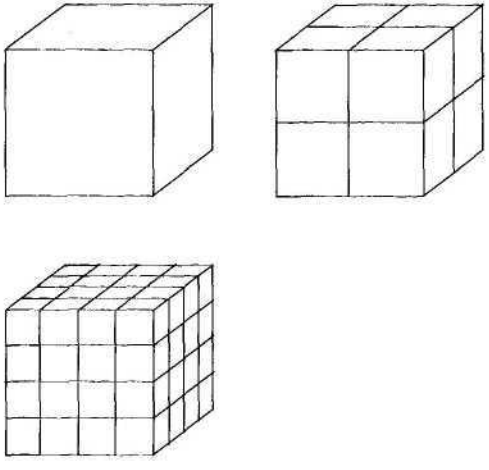

FIGURE 4.9.2 *Subdividing the skybox's faces will improve image quality at the cost of increased texture memory and polygon count. Upper left) One texture per side. Upper right) Four textures per side. Lower left) Sixteen textures per side.*

Skybox Size

Because the camera is always at the center, it does not matter how big we make the skybox as long as it falls within the viewing frustum. As the size of the skybox increases, its walls become proportionally more distant, keeping the same portion in frame.

The skybox dimensions should be chosen so that the farthest points, the corners, are closer to the camera than the far clipping plane. The following formula can be used to calculate the maximum skybox size:

$$\text{skyboxWidth} < \frac{2\sqrt{3}}{3} z_{far}$$

where *skyboxWidth* is the length of one edge of the skybox in world units, and z_{faP}. is the distance to the far clipping plane in world units.

Rendering the Scene

The simplest technique for drawing a skybox is to render an ordinary textured cube, aligned to the world axes and centered on the camera's world position. The skybox should be the first thing rendered. There is no need to clear the color buffer first, as the skybox will always fill the entire screen.

When rendering a skybox, both depth testing and depth writing should be disabled, as the skybox geometry will not intersect properly with the other geometry in the scene. Lighting, fogging, and any other similar effects should also be disabled. The skybox images should be rendered without any atmospheric modification.

As noted earlier, a simple one-texture-per-face cube may produce significant texture stretching. Texture filtering can be used to reduce the effect of this stretching. Texture filtering reduces the jagginess of stretched images by sampling neighboring texels and blending them together.

Texture filtering may cause seams to appear along the edges of the skybox, where two textures meet. These seams are caused by improper texture wrapping. Texture wrapping determines how the texture filter decides which texels "neighbor" the texels at the edge of a texture. If texture wrapping is set to repeat (`GL_REPEAT` in OpenGL), texels on one edge will be combined with texels on the opposite edge, causing seams between textures to stand out.

In OpenGL, texture wrapping should be set to `GL_CLAMP_TO_EDGE_EXT`, if possible. This will clamp filtering to the edge the texture, eliminating interference from border texels. `GL_EXT_texture_edge_clamp` is an OpenGL extension, so it might not be available on all systems. If the `GL_CLAMP_TO_EDGE_EXT` mode is not available, texture filtering should be set to `GL_CLAMP`, which combines edge texels with the texture's constant border color.

Cube Environment Mapping

Cube environment mapping is an alternative to traditional skybox rendering. Cube environment mapping combines all six sides of a skybox into a single texture.

Traditionally, texture coordinates are specified as offsets into a two-dimensional texture map, with (0,0) being one corner of the texture, and (1,1) being the opposite corner. In cube environment mapping, six textures are combined to form a cube, and texture coordinates are specified as three-dimensional vectors, pointing outward from the center of the cube to a point on its surface (Figure 4.9.3).

Cube environment mapping can be used to render distant scenery, and to cast reflections of the skybox on nearby shiny objects.

Cube environment mapping limits skyboxes to one 2D texture per face, so subdividing faces to increase resolution is not possible. Fortunately, most cube environment mapping systems support texture resolutions above 256x256.

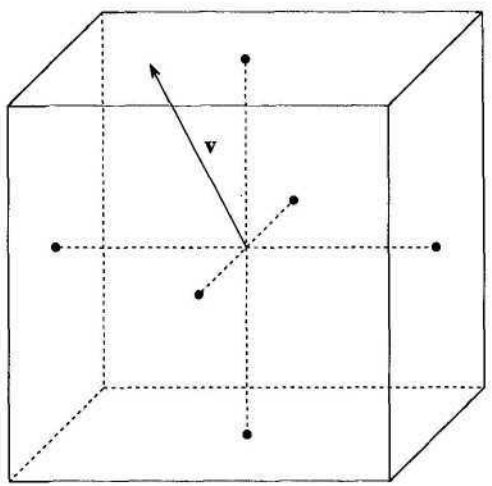

FIGURE 4.9.3 *Texture coordinates for cube environment maps are specified as vectors pointing to the surface of a cube centered on the origin.*

Generating Skybox Textures

Some software packages provide direct support for generating skyboxes. For those that don't, manually generating skybox images is simple.

The size of the output image should be set to texture resolution (e.g., 256x256) and the camera's field of view set to 90 degrees. Six renderings are generated, with the camera looking up and down each of the three cardinal axes (left, right, up, down, forward, and back).

Conclusion

A sense of place is critical to most 3D games. Rendering distant scenery not only increases visual beauty, it also provides players with a means of orienting themselves in the virtual environment. Skyboxes provide an economical and effective way to render distant scenery.

Source Code

Sample source code is provided using OpenGL and GLUT. The sample program demonstrates both 2D and cube environment map rendering.

Cube environment mapping is supported in OpenGL via the GL_ARB_texture_cube_map extension. The sample code checks for the presence of this extension at startup. In cube environment map mode, a shiny object is placed in the camera's view, demonstrating reflectance.

Skybox textures were generated with Terragen (Copyright © 1997—2000 Planetside Software).