

Texture Mapping

Christopher Moore

February 21, 2004

Abstract

Texture mapping is an effective method of adding detail to meshes of 3D models. This paper will review the concepts behind the software implementation and then examine several techniques of texture mapping. These techniques include several coordinate generation methods.

1 Introduction

Originally objects were shaded using either solid shading or interpolation between vertices. While these rendering methods can support interesting effects, it becomes challenging to render colorfully elaborate surfaces. The only way to recreate these images using solid and vertex coloring would be to increase the number of polygons to match the number of distinct color points within an object. This costs memory and rendering performance.

One solution that has been discovered is to store the color information in a separate data structure called a texture. Textures represent a mapping from points on an object's surface to various colors. Textures exist in several different forms. Procedural textures are textures described with functions. Otherwise textures typically consist of rectangular arrays. The smallest unit within a texture is called a texel. Each texels represent a color values.

2 Mapping

Points on the surface of an object are mapped to texture space in several ways. The most simple method is found when applying 3D textures. In this case, the color applied to each point on the surface is the color found at the same point within the texture's space. Textures are most commonly mapped to locations on an object surface. Texture coordinates are assigned to points on an object surface. These texture coordinates specify which portions of the texture are to be mapped upon the surface at their associated points. These texture coordinates are used to look up the associated color at that point on the surface.

There are several different methods of mapping texture coordinates onto mesh surfaces. The most common method is for the designer of the mesh to manually specify the texture coordinates at each vertex. However functionally generating the texture coordinates is just as useful and can be more applicable in certain situations. [5]

Three dimensional texture maps provides a direct mapping of object space to texture space. Because of this they are incredibly simple to conceptualize. Wherever a point exists on an object will be where the color is referenced inside a 3D texture. [5]

Two dimensional textures are typically mapped to objects in one of several methods. Linear maps convert points on the surface to points within a texture by taking the surface point vector and transforming it by a linear system. Cylindrical mapping compute 2D coordinates within the texture by calculating the yaw of the surface point about the center of the object, and the height of the surface point. Spherical mapping operates on the same principle but computes its coordinates using the yaw and pitch of each surface point. [5]

There are several uses of one-dimensional textures. One such use is a method of rendering called cartoon shading, or cel shading. Cel shading occurs by mapping the cosine between the surface normal and the light vector into texture space. Cel shading is a relatively new concept for real-time consumer-level graphics-hardware driven applications. Texture mapping is not the only way to render objects using cel shading. Cel shading can also be accomplished using per-pixel color calculations, which is the method that most graphics applications perform the rendering method.

3 Rendering

One method of mapping textures to the screen is called forward mapping. This involves cycling through the texture and mapping every texel to its associated pixel. This method can result in producing holes within the textured image. This also makes it difficult for texels to be mapped to multiple pixels. As a result it is not used very often. [5]

The most common method used is inverse mapping. It involves cycling through the pixels of the polygon projected onto the screen, looking up the correct color based upon the associated texture coordinates, and writing that color to the point on the screen. [5]

Texture coordinates with values outside of these bounds can be treated several different ways. The most common is for the value to be repeated over, useful for tiling textures across a polygon. Another option is for values to be clamped to their valid boundaries, useful for environment and reflection mapping. Several other options exist, such as repeating textures while mirroring every odd instance of the repetition. [3]

Affine texture mapping can be used in projection views that contain no perspective transform. Affine texture mapping involves linear interpolation of texture coordinates across screen space. At each pixel the interpolated texture coordinate is used to look up the texel mapped to that location. Affine texture mapping is the most straightforward method of interpolating texture coordinates. [4]

Affine texture tends to produce distortion when applied to perspective rendering. This occurs when viewing textured polygons sloped towards the viewer. The mapped texture appears to slide across the surface of larger polygons as the view changes. This is due to the nature of perspective views. The portion of the polygon further from the view will appear smaller, while the portion closer to the view will appear larger. As a result, linear interpolation of the texture across the surface will map excessive texels to the closer portion of the polygon while covering the further portion with not enough texels. [4]

A solution to this distortion is through perspective texture mapping. Perspective texture mapping involves interpolating texture coordinates within perspective space rather than screen space. This is accomplished by scaling the texture coordinates by the inverse of the depth at each pixel before interpolating across the polygon. Once in perspective space, the perspective texture coordinates are incremented by the a value proportional to the inverse of the depth at each pixel. The

perspective texture coordinates are then scaled by the depth to retrieve the true texture coordinates. [2]

4 Filters

Once texture coordinates have been calculated for a pixel, there are several methods that can be used to find the color associated with the texture coordinates.

The most simple of all methods is to round each texture coordinate to the next lowest integer and use this value to find the associated pixel. This approach was most popular when texture interpolation was performed in fixed-point precision using integer variables. Once a texture coordinate had been calculated the programmer only needed to perform a bit shift operation to derive the integer value from the texture coordinates. As floating point computations have become faster, and as hardware support has grown, this method is being used less and less. [3]

While using the next lowest integer index to compute the texture color can be computed quickly, it does not always look so good. This is especially true when viewing polygons that have a small number of texels mapped to them. These textures appear to look very blocky and unattractive due to the discontinuities between colors in the texture. [3]

One correction for this is by applying a bilinear filter to the texture. This is accomplished by using the fractional portion of each texture coordinate to interpolate between the mapped texel and the next. The result image appears smoother, though tends to appear blurry with lower resolution textures. [3]

Another issue which textures face is that of aliasing. High detailed textures rendered at far distances tend to appear cluttered due to the large number of pixels mapped to small areas of the screen. As these polygons move relative to the view they appear to flicker. A solution to this problem is called mip-mapping. The process of mip-mapping involves creating a set of smaller, resampled versions of the original texture. As a textured polygon is rendered, the resampled version of the texture is chosen depending upon the depth of each pixel. Texels are then accessed from the chosen resampled texture and rendered to the screen. [1]

5 Conclusion

Texture mapping is a useful means of increasing the visual complexity of an object without increasing the memory usage. Textures can be mapped onto objects through several different methods. When applied correctly the results can appear very appealing.

References

- [1] Andrew Flavell. Run-time mip-map filtering. *Game Developer*, November, 1998.
- [2] Tom Hammerslevs. Perspective corrected texturing, 2004.
- [3] OpenGL Manual Pages, 2004.
- [4] Henry P. Moreton Paul S. Heckbert. Interpolation for polygon texture mapping and shading. 1991.

[5] Peter Shirley. *Fundamentals of Computer Graphics*. A K Peters, Ltd., Maryland, 2002.