Texture Mapping

Texture Mapping is used to enhance the surfaces of three-dimensional surface models with two-dimensional images - so-called “textures” - and surface properties. It was pioneered by Edwin Catmull in 1974 [1]. Textures make computer-generated images appear more detailed and realistic without having to refine the underlying model itself.

Texture coordinates

Each vertex of a 3D object can be assigned a texture coordinate in the uvw space in addition to its position in the xyz space. The texture coordinates (also known as uv- or uvw-coordinates) are used to define how a texture (a bitmap or a mathematical texture) is mapped on a polygon. If a two-dimensional bitmap texture is used, as is the case in computer games, only the u- and v-coordinates are needed to determine which part of the image is to be mapped on the polygon. For mathematical textures, such as 3D noise or volumetric textures, the w-coordinate is also required.

The uv coordinate (0,0) corresponds to the lower left corner of the texture and the uv coordinate (1,1) of the upper right corner. uv-values greater than 1 and smaller than 0 are possible and require edge repetition effect definitions of the texture. Two possibilities are edge repetition or mirroring as shown in Figure 1. A texture can be tiled by defining the texture coordinates using a polygon.

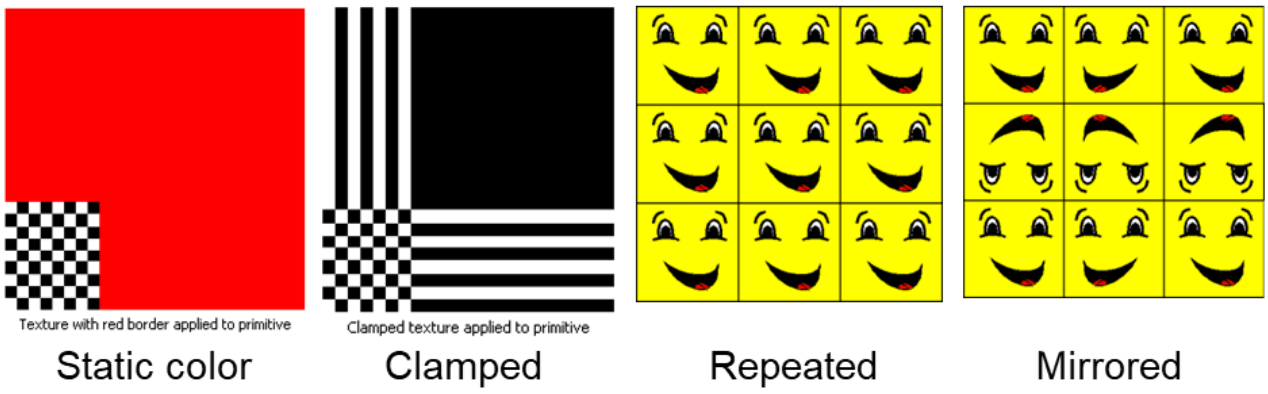

Figure 1: Different ways to handle uv-coordinates > 1 and < 0.

It is also possible to assign several texture coordinates to a vertex. This is referred to as multiple channel mapping. In this way it is possible to display several images or image sections superimposed on a polygon.

In 3D models with many polygons, a single texture is often used for the whole model, so that each point of the model has only one set of texture coordinates (and not different texture coordinates for the different polygons using that point), because this format is particularly convenient for hardware-accelerated 3D graphics and also for the designer of the 3D model.

In the simplest variant of texture mapping, the texture coordinates are interpolated linearly along the polygon’s border lines, which have been transformed from 3D to 2D space. Then they are interpolated linearly along a screen line (or column) from border to border, with each pixel the colour value of the Texel (picture point in the texture) belonging to the interpolated (u, v) coordinates is assigned.

Perspective correction

In the case of polygons that have a greater extent in the direction of view, the above procedure leads to visually unsatisfactory results because the texture coordinates are interpolated after projection and thus do not take into account that a distance in the more distant part of the projected polygon corresponds to a greater distance in

the original polygon in 3D space than a point in the closer part of the polygon. This changes the assignment of texture coordinates to points in three-dimensional space when the perspective changes. The effect is illustrated in Figure 2.

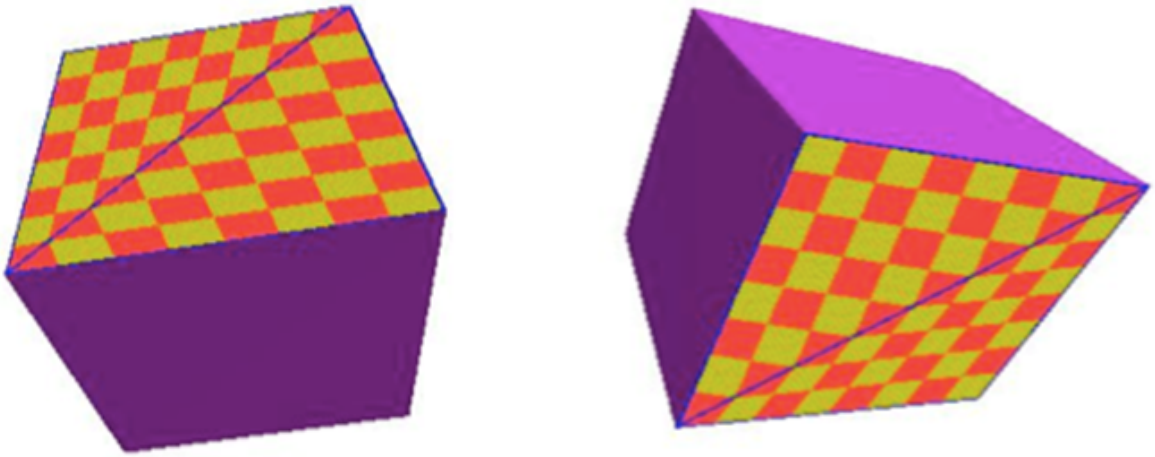

Figure 2: Example for wrong texture mapping caused by perspective distortion.

In order to solve this problem, instead of the texture coordinates u and v, the values of u/z and v/z and also 1/z are interpolated linearly, whereby z is the coordinate in 3D space in the direction of sight (z or 1/z must therefore be stored to each projected point of the polygon). In order to calculate the texture coordinates for a pixel, divisions must now be executed: $u = (\frac{u}{z}/\frac{1}{z}), v = (\frac{v}{z}/\frac{1}{z})$.

Because divisions are relatively slow operations, they are usually not performed on every pixel; instead, u and v are calculated only on a few pixels evenly distributed over the polygon. For all other pixels, the values of u and v are interpolated between those pixels. In this way, the disturbing effects can be greatly reduced without using too much computing power.

Texture interpolation

Naïve methods assume that each pixel can be assigned to exactly one Texel. However, if one looks at both pixels and texels as points without extent, this is generally not the case. Rather, the texture coordinates of a pixel are usually located between several texels. It is therefore necessary to decide how the color value for the pixel is determined from the colour values of the surrounding texels.

The easiest and fastest interpolation method is to simply select the closest Texel. This procedure is called nearest neighbour or point sampling. With the more complex bilinear filtering, the desired colour value from the four surrounding texels is interpolated depending on their distance. Even more complex filters, such as the Gauss filter, include further texels in the calculation or weigh the distance differently. Since inappropriate interpolation methods lead to unwanted aliasing effects - such as Moiré effects - a compromise must be found between speed and artefacts.

Multum in parvo (MIP) Mapping

This mapping technique can be used as long as the raster spacing of the pixels is smaller than that of the texels, i.e. a maximum of one texel is assigned to an arbitrary pixel. However, if the raster spacing of the pixels is larger than that of the texture, one pixel corresponds to a whole area of the texture. Although it is no longer difficult to compute the colour value as an average of all texels, it is very time-consuming and therefore not

practicable.

Instead, multum in parvo (approx.: “a lot in small space”), MIP maps are used. In addition to the original texture, these contain copies of the texture with decreasing size (usually a Gaussian Pyramid representation), so-called “detail levels” (level of detail, LOD). If the largest level of detail is chosen, which restores the usual condition of “pixels smaller than texel” and one can work on it like on the original texture. In addition to the previous interpolation methods, it is also possible to carry out another linear interpolation between two successive levels of detail; in combination with bilinear filtering, this results in trilinear filtering. Furthermore, objects further away in a scene can use smaller LODs, which reduces aliasing further through the properties of Gaussian image sub sampling.

The use of MIP maps in conjunction with point sampling greatly reduces aliasing effects, in combination with more complex filters and interpolation between the detail levels, they can be reduced to a minimum.

Anisotropic filtering

MIP mapping still considers pixels and texels as points, i.e. as one-dimensional objects without extent. Instead, they can also be seen as small squares. Then it has to be taken into account that a pixel projected onto the texture does not form a square, but rather a plane stretched in one direction, when the polygon expands in the direction of view. If this different propagation in different directions (anisotropy) of the pixel in the texture space is taken into account during filtering, this is called anisotropic filtering.

All modern GPUs provide specialised fixed function units called texture samplers to perform MIP and anisotropic filtering texture mapping.

Extensions

There are several methods to make a textured surface appear more tessellated and three-dimensional.

Bump mapping [4] uses an additional texture to define a new normal vector for each vertex. Thus, the lighting calculation is done with a normal vector that varies over the surface.

Displacement mapping [3] uses the information from an additional texture to create additional polygons, i.e. introducing geometric offsets in the texture.

In *environment mapping* [2], an additional texture is used to simulate reflection.

References

[1] Edwin Earl Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces.* PhD thesis, 1974. AAI7504786.

[2] Ned Greene. Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl.*, 6(11):21–29, November 1986.

[3] Johannes Hirche, Alexander Ehlert, Stefan Guthe, and Michael Doggett. Hardware accelerated per-pixel displacement mapping. In *Proceedings of Graphics Interface 2004*, GI ’04, pages 153–158, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[4] Jan Kautz, Wolfgang Heidrich, and Hans-Peter Seidel. Real-time bump map synthesis. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, HWWS ’01, pages 109–114, New York, NY, USA, 2001. ACM.