Rasterization and sampling

Rasterization of polygons is based on the rasterization of lines, but requires additional effort for thicker line widths. The rasterization of filled polygons plays a major role in 3D computer graphics, as 3D scenes can be rendered by colour-filling of polygons projected onto the image plane.

When rasterizing thick line segments, it is necessary to decide how they are joined together. Considering thick lines as rectangles produces unsightly results. It is better to draw lines with round ends. Another possibility is mitering, where the line ends are drawn at an angle so that they border each other. At very sharp angles, the line ends protrude too far above the actual polygon corner, which is why they are better cut off.

When rasterizing polygons, it must be decided how their coordinates are interpreted. For example, if a rectangle is rasterized with the endpoint coordinates (1, 1) and (6, 3), 6×3 pixels are normally rendered, although the rectangle is only 5×2 units in size. This unwanted effect is a consequence of finite image resolution. Drawing polygons next to each other causes some pixels to be rendered several times. One way of overcoming this problem, at least in the case of integer coordinates, is to move them to the left and down by half a pixel distance during screening, so that in reality the rectangle is rasterized with the coordinates (0.5, 0.5) and (5.5, 2.5). This avoids that edges of adjacent polygons are painted twice, which would cause undesirable results in certain raster operations such as XOR. An alternative method that avoids floating point numbers is not to rasterize the last pixel of a row or column. It is usually acceptable that the polygon appears slightly displaced.

Even very acute-angled polygon corners can cause pixels to be painted by several segments. Another example of artefacts are slivers, polygon parts that are so thin that they do not include any pixels and where some raster algorithms draw only single or no pixels.

Polygons with more than three corners as part of polygon meshes are often converted into triangles before rasterizing. Many triangles share their edges. If these adjacent triangles are coloured differently and each edge is rasterized as a single-coloured line, the appearance depends on the order in which the triangles are rastered.

In order to avoid this problem, a pixel is painted only if it lies within the triangle to be drawn. Barycentric coordinates can be used for this purpose. Referring to a triangle, each point of the plane can be described by the coordinates (α, β, γ). A point or pixel is strictly within this triangle when each of these coordinates is in the $]0.1[$ interval. This method is also valid for non-integer corner point coordinates.

A special case is represented by pixels that are located exactly on an edge and therefore cannot be assigned to one of the two adjacent triangles. In these cases, a fiducial is selected outside the image and the colour of the triangle whose corner point is not on the edge and which is closer to that point is chosen. This method works whenever the reference point is not on the straight line passing through the edge.

Barycentric Coordinates

In linear algebra and geometry, barycentric coordinates (also homogeneous barycentric coordinates) are used to describe the position of points in relation to a given distance, triangle, tetrahedron or simplex in general. The point is represented by the coefficients of an affine combination (i.e. a linear combination of points, where the sum of the coefficients is 1). They are a special case of homogeneous coordinates.

$x_1, \ldots, x_n$ are the corner points of a Simplex in the vector space A. If the Equation 1 is satisfied for a point p from A,

$$(a_1 + \cdots + a_n) \cdot p = a_1 x_1 + \cdots + a_n x_n. \tag{1}$$

We call the coefficients $(a_1, \ldots, a_n)$ "barycentric coordinates" from p to $x_1, \ldots, x_n$. The vertices have the coordinates $(1, 0, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, 0, 0, 0, \ldots, 1)$.

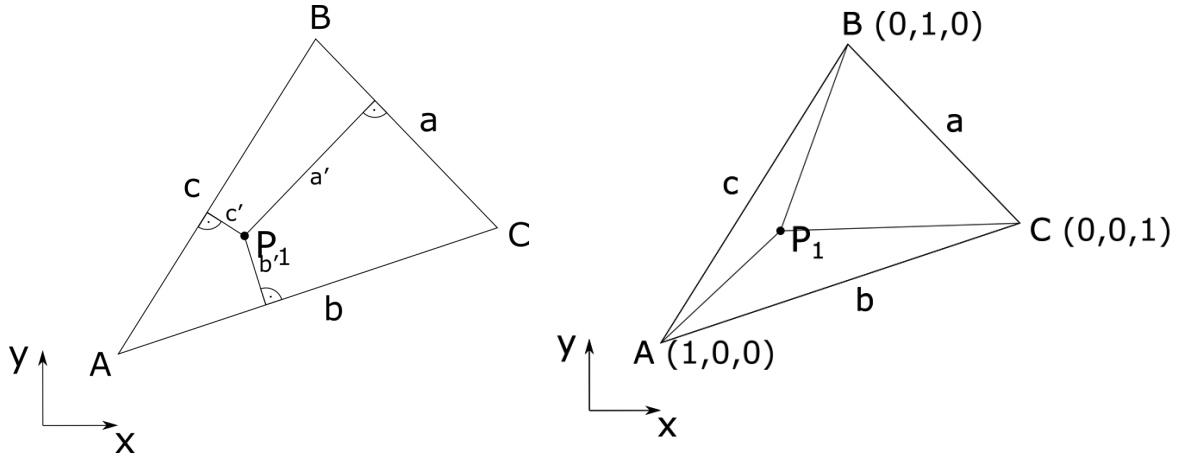
Barycentric coordinates are not unique: For each b are $(ba_1, \ldots, ba_n)$ also barycentric coordinates of p.

If the coordinates are positive, the point p is located in the Convex hull of $x_1, \ldots, x_n$, the simplex with

these vertices. The representation of a point within a convex hull as the sum of vertices of a simplex is called "affine combination" or "barycentric combination".

If we imagine masses in the ratio $a_1, \ldots, a_n$ at the corners of the simplex, then the mass center (the "barycenter") is located in p. This is the origin of the term "barycentric", which was introduced in 1827 by August Ferdinand Möbius.

In addition to trilinear coordinates, barycentric coordinates are often used in triangular geometry to describe the positions of certain points. The difference between trilinear and barycentric coordinates is illustrated in Figure 1.



(a) trilinear coordiantes (b) barycentric coordiantes

Figure 1: Difference between trilinear and barycentric coordinates.

The barycentric coordinates of a point P are given by

$$(A_{BCP},\ A_{CAP},\ A_{ABP}) \tag{2}$$

Here A_{BCP}, A_{CAP} and A_{ABP} mean orientated triangular surfaces, i.e., The surfaces of the triangles receive a positive sign if the direction of rotation is positive (opposite clock orientation), otherwise a negative sign.

It can be easily tested if a point p is inside a triangle by normalising $(A_{BCP} = \alpha,\ A_{CAP} = \beta,\ A_{ABP} = \gamma)$ with $\alpha + \beta + \gamma = 1$, hence defining the barycentric coordinates of p as $p(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$. The test simplifies to p is inside the triangle if

$$0 < \alpha < 1 \tag{3}$$
$$0 < \beta < 1 \tag{4}$$
$$0 < \gamma < 1. \tag{5}$$

If p is on an edge one coefficient will be 0. If p is exactly at the same location as a vertex two coefficients will be 0 and the remaining one will be 1.

Conversion

There is a simple correlation between trilinear and barycentric coordinates: If the given point has the trilinear coordinates (r, s, t), (ar, bs, ct) are barycentric coordinates of this point. a, b and c indicate the side lengths of the triangle.

The conversion into Cartesian coordinates is done by determining the weighted average.

Are (u, v, w) barycentric coordinates of a point $P(x, y)$ and (x_A, y_A), (x_B, y_B), (x_C, y_C) the Cartesian

coordinates of the corners of the given triangle then

$$x = \frac{u \cdot x_A + v \cdot x_B + w \cdot x_C}{u + v + w} \text{ and} \tag{6}$$

$$y = \frac{u \cdot y_A + v \cdot y_B + w \cdot y_C}{u + v + w}. \tag{7}$$

Interpolation

An interpolation method is based on barycentric coordinates, which generalizes the General Linear Interpolation for Functions of several variables.

In the case of a function f of two x and y are $A(x_A, y_A)$, $B(x_B, y_B)$ and $> C(x_C, y_C)$ the function values are given for three points. In this case A, B and C must not lie on a straight line. One must therefore set up a triangle ABC. If an arbitrary point (x, y) is given, then one can definne define

$$f(x, y) = a\, f(x_A, y_A) + b\, f(x_B, y_B) + c\, f(x_C, y_C), \tag{8}$$

where (a, b, c) are the normalised barycentric coordinates of (x, y). This interpolation works also for points outside the triangle.

Polygon filling

The easiest way to rasterize filled triangles and thus polygons is to use the barycentric test for each pixel of the image: Pixels are only painted if they lie within a triangle. A more efficient method only tests those pixels within a rectangle that includes the triangle to be rasterized. In addition to these simple methods, however, faster methods have been developed.

Barycentric coordinates can be used to rasterize triangles and interpolate colour values at the same time as shown in Algorithm 1. More advanced techniques, such as [2] use similar approaches but terminate early if conditions are not met.

Algorithm 1 Polygon filling using barycentric coordinates.

```
1: for all x do do
2:    for all y do do
3:       compute p(α, β, γ) for p(x, y)
4:       if (0 < α < 1 and 0 < β < 1 and 0 < γ < 1 ) then
5:          colour = αc0 + βc1 + γc2
6:          drawpixel(x,y) with colour
7:       end if
8:    end for
9: end for
```

Which triangle to draw? The z-buffer algorithm

Polygons will naturally overlap along viewing directions and in case of opaque objects, only the one that is closest to the viewer needs to be drawn. This can be achieved through the naïve Painter's algorithm or the z-buffer algorithm.

In addition to the visible part of the image memory, which contains the current color values, Wolfgang Strasser defined in his 1974 dissertation [12] another buffer, the z-buffer, which contains the depth of the visible object at each pixel. Alternatively, the pixel values in the frame buffer can be extended by a z-value. At the beginning, the entries in the z-buffer are set to a value representing an infinite distance (backplane distance). The frame buffer is initialized with the background colour. Each polygon is rasterized. Only if the currently

rasterized point of the polygon is closer to the viewer than the point whose distance is entered in the z-buffer, the values in the z-buffer and in the frame buffer are replaced by the distance or colour of the current polygon.

The order in which the polygons are rasterized is basically arbitrary. Not only polygons, but any other graphic primitives can be rendered using the z-buffer algorithm.

The memory precision of the values in the z-buffer has a great influence on the quality of the rendered image. If two objects are very close together, artifacts can occur using 8-bit per z-buffer pixel. 16,24 or 32 bit deep z-buffers produce fewer artefacts but are more memory intensive.

On current graphics cards, the z-buffer consumes a significant amount of available memory and bandwidth. Different methods are used to reduce the influence of the z-buffer on performance. This is made possible, for example, by lossless compression of the z-data, since compression and decompression is more cost-effective than increasing the data transfer rate of a card. Another method saves erasing operations in the z-buffer: the depth information is written to the z-buffer with alternating sign. An image is saved with a positive sign, the next image with a negative sign, then it is erased. Another possibility for optimization is pre-sorting of primitives: If the closest primitives are rendered first, it can be decided later which pixels have to be rendered and which are hidden by foreground objects, thus saving texturing and pixel shader processes. However, pre-sorting (e.g. painter's algorithm) is usually much more computationally expensive than using the z-buffer algorithm.

A simple version of the z-buffer algorithm is outlined in Algorithm 2.

Algorithm 2 A simple version of the z-buffer algorithm.

Let CB be color (frame) buffer, ZB be the z-buffer (equal size)
for each triangle T **do**
  Rasterize T to generate fragments
  for each fragment F with screen position (x, y, z) and color value C **do**
    if $(z < ZB[x, y])$ **then**
      Update color: $CB(x, y) = C$
      Update depth: $ZB(x, y) = z$
    end if
  end for
end for

Anti-aliasing

Sampling and Prealiasing: The generation of a raster graphic from an image description by rasterization or image synthesis comprises of assigning a color to each discrete pixel. This process can be interpreted as a sampling of a signal as defined by signal processing theory. Computer graphics is unique in that many signals are present as abstract image descriptions, which can only be evaluated algorithmically at individual points. An example are images that are calculated by means of ray tracing. Such signals can also be called procedural signals [11].

When rasterizing without antialiasing, the image description is evaluated exclusively on the pixels; other points of the image are not included in the color of the pixels. A problem with this method is that small figures are not captured by the pixel raster and therefore do not appear in the rasterized image. If small image details are arranged regularly, they interfere with the pixel grid, resulting in alias effects. This is illustrated by the adjoining computer-generated picture of an infinitely large chess board.

The period length of the signal is equal to the size of two projected chessboard fields. As soon as the period length falls below two pixel distances, i. e. exceeds the Nyquist frequency, the signal is under-sampled. Certain chessboard fields are no longer captured by the pixel grid; the chessboard pattern is destroyed. According to the Nyquist Shannon sampling theorem, alias effects occur where a high original frequency in the sampling is expressed as a misleading low frequency: Close to the horizon you get the wrong impression that the original signal would contain oddly large fields. The illustrative term "picket fence effect", which is occasionally used

in German, makes this clear: one considers the spatial frequency spectrum of the original image through a frequency resolution fixed by the displaying device. This type of alias effects that are a result of the sampling method is called pre-aliasing [11].

Even if an image is computed with higher image resolutions and thus higher sampling rates, disruptive aliasing effects would occur near the horizon, but only at a higher frequency. This is due to the fact that image gradients towards the horizon are getting smaller and smaller and thus the spatial frequency is unlimited. An example for aliasing can be seen in Figure 2.

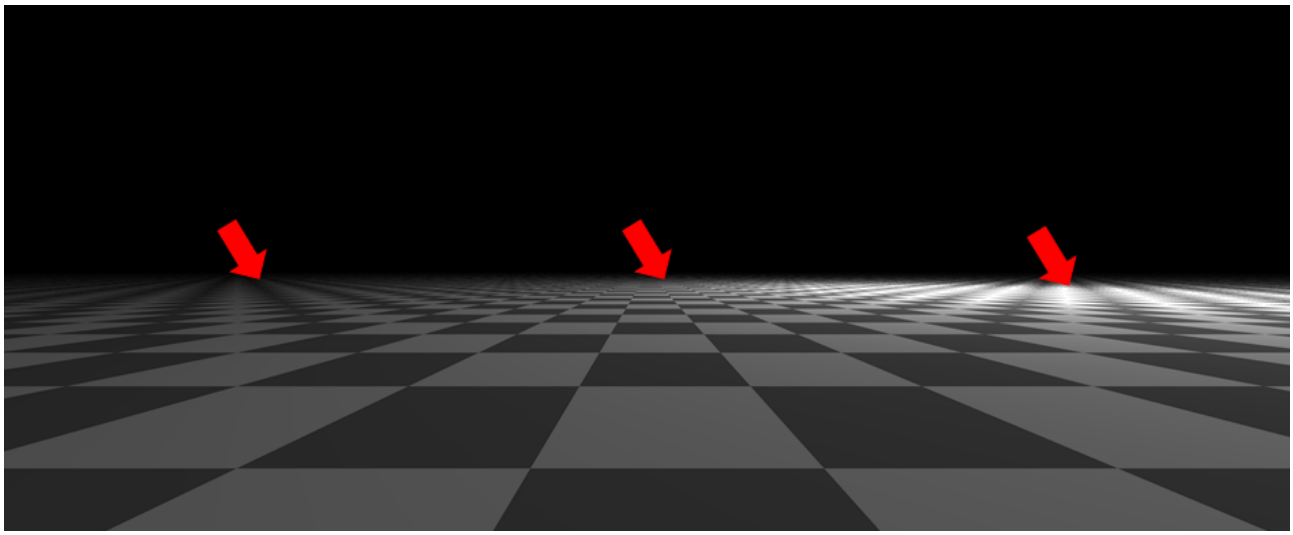


Figure 2: The aliasing effect is strongest close to the horizon where viewing angle and floor elements are almost parallel and objects get very small. Additionally this scene comprises of high-frequency image gradients that cannot be accurately sampled.

Reconstruction and post-aliasing: In signal processing, reconstruction refers to the transformation of a discrete signal into a continuous signal by interpolating between the individual sampled values by means of a reconstruction filter. In computer graphics, the term is interpreted in a slightly different way, since no continuous signal is generated. Rather, reconstruction in this context means calculating the colour of a pixel from the colour values near the pixel, for example to scale a raster graphic [13]. The reconstruction filter used is a two-dimensional function that is centred over the pixel to be calculated and indicates how sample values are weighted. For example, a cone-shaped filter places the highest weighting on a colour value determined directly at the pixel, while farther away values have less influence. Colour values outside the support of the reconstruction filter are ignored. The colour value of the pixel is the sum of the weighted colour values around the center of the filter kernel. This operation corresponds to a convolution with a low pass filter of an order of the selected anti-aliasing level.

Anti-aliasing by means of a box filter, which calculates the mean value of all colour values within a square around the pixel, is also called “weighted area sampling” [7]. Examples of reconstruction filters with weighted area sampling are the Mitchell-Netravali filters (bicubic filters), the Lanczos filter or the Gauss filter.

Without anti-aliasing, each pixel is sampled only once at a position that remains constant relative to the pixel. With anti-aliasing, the image description is evaluated at several and/or different positions relative to the pixel. From the values determined in this way, the colour of the pixel is calculated according to a reconstruction filter. By choosing a suitable sampling method, pre-aliasing can be reduced or avoided by selecting suitable post-aliasing reconstruction filters. The term “anti-aliasing” is misleading in that anti-aliasing is used not only against alias effects, but also against the staircase effect and other undesirable effects, such as small objects falling through the pixel raster. Franklin Crow provided the first description of the aliasing effect as the cause of image artifacts in computer graphics in 1977 [4].

Traditional anti-aliasing techniques of signal processing are not easily transferable to computer graphics. This is due to the fact that computer graphics, in contrast to audio signals for example, have the following

special features when viewed from the signal processing point of view:

- The signals that occur in computer graphics are artificial, abstract descriptions of images that often cannot be captured in their entirety, but can only be evaluated at individual points.
- Computer graphics usually contains hard object edges, which sometimes corresponds to infinite spatial frequencies. Therefore, a perfect reconstruction of the output signal by means of an ideal low pass (sinc filter) is not possible.
- Image descriptions are often not sampled regularly, but at irregularly distributed positions.
- Human visual perception is particularly sensitive to the artefacts caused by less effective anti-aliasing methods.

Prefiltering: In computer graphics, pre-filtering is the process of determining the colour of a pixel without performing individual samples. Instead, the colour is calculated directly from the image description. As described above, the colour value of a pixel corresponds to the total weighted by the reconstruction filter of all colour values of the objects overlapped by the filter kernel. Pre-filtering is only possible for image descriptions whose convolution can be analytically calculated with the reconstruction filter, i.e. can be expressed in the form of known functions. This includes simple geometric objects such as lines. However, pre-filtering cannot be applied to procedural signals, since they can only be sampled at individual points.

One of the first pre-filtering techniques for computer graphics was described by Edwin Catmull in 1978. His algorithm uses unweighted area sampling. The color of a pixel is calculated by clipping the polygons that make up the image against the pixel to determine its area portion. This method is so slow that it could only be used for two-dimensional computer animations with some large polygons [13]. Later methods tried to approximate the area proportions of the object fragments using bitmasks [6] - including Carpenter's A-Buffer, sometimes called multisampling [3] - or lookup tables [1]. In addition, anti-aliasing methods tailored to the rasterization of basic shapes such as lines and circles were developed. High-quality pre-filtering anti-aliasing of any curves with different reconstruction filters is also possibl [5].

Postfiltering: Post-filtering or super-sampling is mainly used if the image description can only be evaluated at individual points. To calculate the colour of each pixel, several sampled values are used, which are weighted by a reconstruction filter. Mathematically, post-filtering is a method for the numerical approximation of the convolutional integral.

Multi-sampling methods differ in the number and distribution of sampling positions per pixel. Real-time rendering usually uses the same pattern for all pixels. Some patterns define different weightings for the sampled values, which is why they can also be considered linear reconstruction filters. If the carrier of a reconstruction filter extends over several pixels and several pixels share some sample values, this is called sample sharing [10].

Another method is Adaptive Supersampling. The number of sampling points can be varied over the image. The decision as to whether to continue sampling is based on local criteria such as contrast. However, such techniques are usually not as common and can lead to artifacts [8].

Instead of distributing the sampling points more or less evenly and weighting the values determined there by a reconstruction filter, importance sampling can be used to calculate the pixel colour. The sampling values are distributed according to the shape of the reconstruction filter (more sampling points close to the pixel) and weighted equally. This method leads to a less noisy result [9].

Commonly used in practice and implemented in hardware is the conceptually simplest method of full-scene anti-aliasing (FSAA): Images are rendered with a higher resolution and then downscaled.

References

[1] Greg Abram and Lee Westover. Efficient alias-free rendering using bit-masks and look-up tables. *SIGGRAPH Comput. Graph.*, 19(3):53–59, July 1985.

[2] Bryan Ackland and Neil Weste. Real time animation playback on a frame store display system. *SIGGRAPH Comput. Graph.*, 14(3):182–188, July 1980.

[3] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008.

[4] Franklin C. Crow. The aliasing problem in computer-generated shaded images. *Commun. ACM*, 20(11):799–805, November 1977.

[5] A. E. Fabris and A. R. Forrest. Antialiasing of curves by discrete pre-filtering. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 317–326, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[6] Eugene Fiume, Alain Fournier, and Larry Rudolph. A parallel scan conversion algorithm with anti-aliasing for a general-purpose ultracomputer. *SIGGRAPH Comput. Graph.*, 17(3):141–150, July 1983.

[7] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice (2Nd Ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[8] David Kirk and James Arvo. Unbiased sampling techniques for image synthesis. *SIGGRAPH Comput. Graph.*, 25(4):153–156, July 1991.

[9] Jaroslav Kivanek and Mark Colbert. Real-time Shading with Filtered Importance Sampling. *Computer Graphics Forum*, 2008.

[10] S. Laine and T. Aila. A Weighted Error Metric and Optimization Method for Antialiasing Patterns. *Computer Graphics Forum*, 2006.

[11] Don P. Mitchell and Arun N. Netravali. Reconstruction filters in computer-graphics. *SIGGRAPH Comput. Graph.*, 22(4):221–228, June 1988.

[12] Wolfgang Strasser. *Schnelle Kurven- und Flächendarstellung auf graphischen Sichtgeräten*. PhD thesis, Technische Universität Berlin, 1974.

[13] Alan Watt. *3D Computer Graphics*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1993.