Graphics Hardware (2006)
M. Olano, P. Slusallek (Editors)

Minimum Triangle Separation for Correct Z-Buffer Occlusion

Kurt Akeley and Jonathan Su

Microsoft Research Asia

Abstract

We show that, and how, window coordinate precision (the representations of x_{win} and y_{win}), field of view, and error accumulated by single-precision mapping arithmetic contribute to, and sometimes dominate, effective z-buffer resolution. Our results are developed analytically, then verified through simulation. Using our approach system designers can allocate numeric precision more efficiently, and programmers can more confidently predict the minimum triangle-to-triangle separation required to ensure correct z-buffer occlusion.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Graphics Processors

1. Introduction

Because z-buffer values (z_{buf}) are represented with finite precision, occlusion fails for triangles that are too close together, resulting in visual artifacts (punch-through). Fixed-point z-buffers using traditional projection arithmetic require separation that increases as the square of the distance from the viewer [Ake90]. For large ratios of distances to the far and near eye-coordinate clipping planes (large $\frac{f}{n}$) and low-precision z_{buf} representations (e.g., 16-bit fixed-point), the result is extremely poor z-buffer resolution throughout much of the eye-coordinate frustum. Graphics programmers have been instructed to move n as far from the viewer as possible, and to utilize atmospheric scattering effects (fog) to roll-off visibility prior to a pulled-in f [Ake90].

More fundamental solutions include changing the projection arithmetic (e.g., w-buffering) and using floating-point z_{buf} representations with the depth mapping reversed ($z_{buf} = 1.0$ near, 0.0 far) [AF98, LJ99], which we and Lapidous et al. refer to as complementary z-buffering. Analyses of these improved approaches have concentrated on error due to finite-precision z-buffer representation, ignoring error due to the finite resolutions of other values, especially of the spatial window coordinates x_{win} and y_{win}. As a result, these analyses approximate the required separation only for fronto-parallel triangles.

We asked the more general question of the minimum object-space separation required between triangles of *arbitrary* orientation to ensure correct occlusion (S_{min}). We first considered systems with ideal precision for all calculations, errors due only to finite-precision representations of x_{win}, y_{win}, and z_{buf}. We then considered the additional effects of finite precision projection, viewport, and rasterization arithmetic, hereafter referred to as mapping.

2. Window Coordinate Representations

Graphics processors typically represent x_{win} and y_{win} in fixed-point, with four to twelve fractional bits (e.g., 10.8, 10 integer and 8 fraction bits). Triangle vertexes are forced to fixed-point representation prior to rasterization, causing their positions to be perturbed. The plane containing a triangle changes, and rasterized z_{buf} values are affected. While S_{min} could be analyzed in terms of the direct effects of x_{win} and y_{win} precision on z_{buf} values, we find it instructive to consider spatial and depth-related precision separately. We ask not when z_{buf} values along a sample ray can intersect, but rather under what circumstances parallel triangles of arbitrary location and orientation can exchange their ordering.

To answer this, we form an uncertainty cuboid for each precise 3-D location in window coordinates (Figure 1a). The depth of an uncertainty cuboid is the numeric distance between the representable z_{buf} values nearest its location. If z_{buf} is fixed-point all uncertainty cuboids have the same depth. If z_{buf} is floating-point, the uncertainty depth is vanishingly small for uncertainty cuboids near $z_{buf} = 0.0$, and

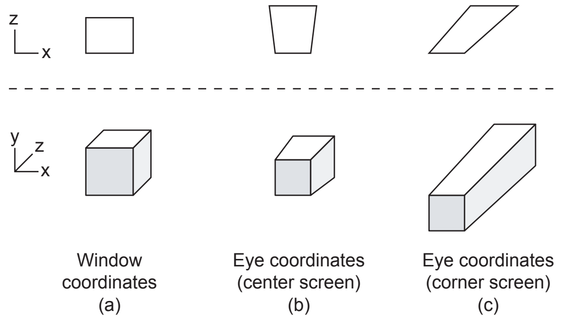
z
x
y
z
x
Window coordinates (a)
Eye coordinates (center screen) (b)
Eye coordinates (corner screen) (c)

Figure 1: *An uncertainty cuboid in (center screen) window coordinates (a) reverse maps to a frustum in eye coordinates (b). Reverse mapping a dimensionally equivalent cuboid located at the upper-right corner of the screen yields a highly sheared frustum (c). Perspective is exaggerated.*

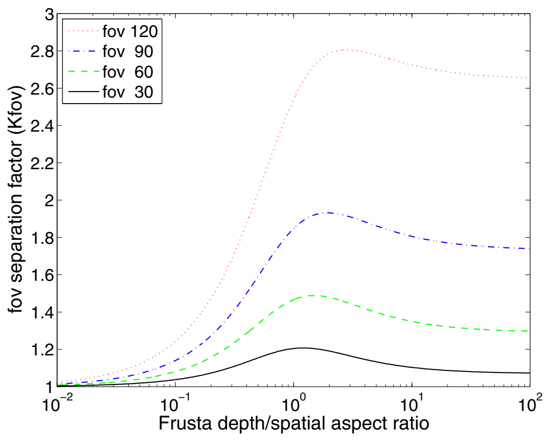

Figure 2: *fov separation factor (K_{fov}), which scales the required triangle separation, relates the longest diagonal of a sheared eye-coordinate uncertainty frustum (Figure 1c) to the same diagonal of the unsheared, center-screen frustum (Figure 1b). This factor is a strong function of the aspect ratio of the unsheared uncertainty volume, computed as the ratio of its depth to its x,y diagonal, and of the (side-to-side) field of view angle (assuming a square viewport).*

equivalent to fixed-point for cuboids near $z_{buf} = 1.0$. (In this work we conservatively treat floating-point uncertainty as a piecewise-continuous function, so that similar values have similar uncertainty.) The widths and heights of all uncertainty cuboids are the same, determined by b, the fractional precision of x_{win} and y_{win}.

To evaluate S_{min} in eye coordinates, we invert the projection and viewport transformations and reverse map the uncertainty cuboids. Under this mapping each cuboid becomes a frustum, albeit one with nearly parallel sides due to its small size. Parallel triangles may swap their order if and only if any of their uncertainty frusta overlap. Thus S_{min} for a location in eye coordinates is the length of the longest diagonal of that location's uncertainty frustum. And the worst-case triangle orientation at that location is perpendicular to the longest diagonal.

Assuming (without limitation) a symmetric viewing frustum, center-screen uncertainty frusta are symmetric (Figure 1b), but frusta in screen corners are highly sheared (Figure 1c). Shear increases the length of the longest diagonal, so, for a given z_{eye} value, S_{min} is greatest for extreme values of x_{eye} and y_{eye}. We define the magnitude of this field of view (fov) separation factor, K_{fov}, as the ratio of corner-screen to center-screen diagonal length for uncertainty frusta on a given z_{eye} plane. K_{fov} increases super-linearly as fov is increased, corresponding to the larger maximum shear. The shape of the uncertainty frusta also affects K_{fov}. Diagonals of shallow frusta are almost unaffected (left side of Figure 2). K_{fov} is maximized for slightly over-deep frusta, and remains substantial as the depth/spatial ratio further increases.

We confirmed our analysis with a simulation implemented using double-precision arithmetic, forcing only x_{win}, y_{win}, and z_{buf} to their nearest representable values. Triangle separation at the average of the z_{eye} values was reported for each pixel punch-through error. When large numbers of triangle pairs are rasterized at multiple depths and orientations, the model is confirmed if punch-through errors occur at separations up to but not exceeding the predicted S_{min}.

We ran the simulation for many combinations of parameters and visually confirmed correspondence with predicted behavior. Figure 3 illustrates our visualization approach. Four line plots indicate required separation for z_{buf} error alone (as though the uncertainty frusta had no width or height), for x_{win} and y_{win} error alone (as though the frusta had no depth), for center-screen uncertainty frusta, and for frusta projecting to screen corners. Punch-through errors, resulting from the rasterization of many randomized triangles clustered at ten discrete z_{eye} values, are plotted with points. These points cluster beneath, but tightly fitted to, the predicted corner-screen separation plot.

The significance of both window-coordinate precision and fov in determining required triangle separations is clearly demonstrated. For the traditional z-buffer modeled in Figure 3 the 90-deg fov nearly doubled S_{min} in the far field (where the uncertainty frusta become over-deep). Likewise, any analysis that ignored the effects of x_{win} and y_{win} representation would substantially underestimate S_{min} in the near field. The observation by Lapidous et al. that applications have not fully utilized traditional z-buffer precision in the near field [LJZW01], which was based on a depth-only analysis, is therefore explained: this precision was almost certainly not available to be utilized.

Assuming a square viewport, the separations in Figure 3

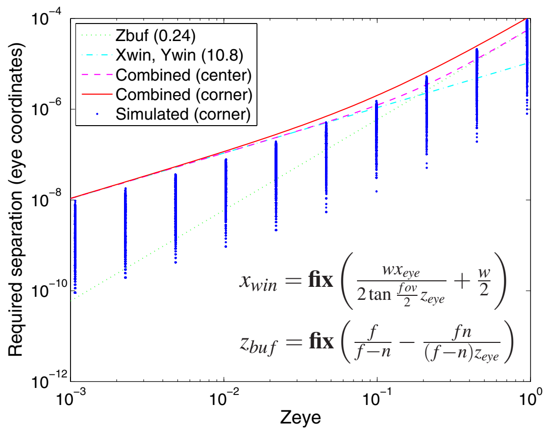

Figure 3: *Required triangle separation, log-log plotted as a function of z_{eye}, for a traditional z-buffer with $f = 1$, $n = 0.001$, $fov = 90$, and a square viewport ($w = h = 1024$). Plots indicate the separations required for uncertainty due to spatial representation (10.8 fixed-point), depth representation (0.24 fixed-point), and the combination of the spatial and depth uncertainties, at both the center and the corner of the screen. Corner-screen simulation results at ten different depths (each error plotted as a dot) show punch-through at separations up to, but not exceeding, the computed value.*

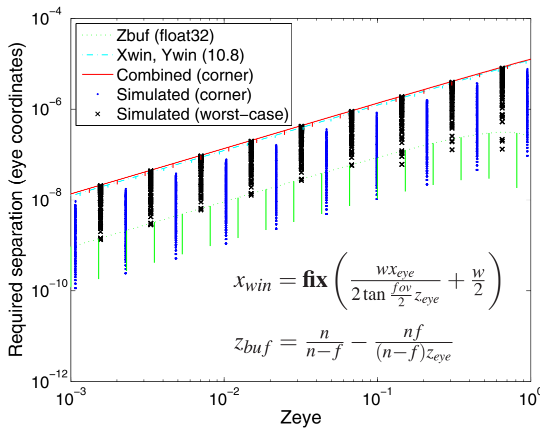

Figure 4: *Required triangle separation, log-log plotted as a function of z_{eye}, for a complementary z-buffer with $f = 1$, $n = 0.001$, $fov = 90$, and a square viewport ($w = h = 1024$). Plots indicate the separations required for uncertainty due to spatial representation (10.8 fixed-point, with mapping error), depth representation (32-bit floating-point, with mapping error), and the combination of the spatial and depth uncertainties (corner screen only). Vertical bars illustrate the increase in required separation due to mapping error. Corner-screen simulation results at ten different depths (each error plotted with a dot) show punch-through errors at separations approaching, but well below, the computed value. Corner-screen simulation results at nine additional depths (plotted with x's), computed with all numeric errors forced to their worst-case values, closely match the computed separation value.*

can be adjusted for different values of f, $\frac{f}{n}$, fov, w, and b as follows:

1. Scale the spatial-only plot by $1024/w$, by 2^{8-b} and by $\tan\frac{fov}{2}$.
2. Scale the depth-only plot by $0.001\frac{f}{n}$.
3. Compute diagonal lengths for center-screen frusta at different depths.
4. Scale these lengths by K_{fov} (estimated using Figure 2).
5. Scale both axes of the plot by f to account for the size of the viewing frustum.

3. Mapping Error

Mapping error further increases x_{win}, y_{win}, and z_{buf} uncertainty, and thus required triangle separation. The interval arithmetic needed to compute this error accumulation is well understood [Hig96], but we were unable to locate a software package that suited our needs. We wrote a simple equation analyzer that uses double-precision arithmetic to approximate ideal computation, modeling accumulated error as a continuous, double-precision value.

Figure 4 illustrates the results of this analysis and simulation for a hypothetical system using a complementary, 32-bit floating-point z-buffer. All mapping arithmetic is modeled as 32-bit floating point, except that barycentric triangle areas are computed exactly, because they could easily be implemented with exact fixed-point arithmetic. The analysis was tested to be invariant to the projected dimensions of the triangles, although extreme aspect ratios were not considered.

Vertical bars extending below the three line plots indicate the portions of these separations due to mapping error. The z_{buf} mapping error bars are long because the precision of the floating-point z_{buf} values matches the precision of the mapping arithmetic. Yet the contribution of mapping error to S_{min} is minor, because spatial-related error dominates depth-related error, and the 10.8 fixed-point spatial precision is far below that of floating-point.

The ten clusters of simulation punch-through errors are all very safely below predicted S_{min}. To ensure that this loose fit was the result of conservative interval evaluation (longer sequences of arithmetic operations are increasingly unlikely to simultaneously introduce worst-case error) and limited simulation, not of analysis error, we modified the simulation to force all arithmetic errors to their worst-case extremes, in the directions indicated by our interval evaluation. The nine clusters of simulation punch-through errors recomputed in this manner, plotted with x's rather than points, are a tight fit to predicted S_{min}.

4. Application Developer Guidance

The parameter adjustments described at the conclusion of Section 2 provide useful intuition for application programmers, but they are impractical for computing accurate S_{min} plots for traditional z-buffer systems, if only due to the added complication of mapping error. Here we suggest an alternate approach for systems with complementary z-buffers, which, we agree with Lapidous et al. [LJ99], are currently the best practice for z-buffering.

Because S_{min} scales directly with f ($\frac{f}{n}$ held constant) the relationship of S_{min} to z_{eye} of the system depicted in Figure 4 can be modeled as a linear one, valid for all f:

$$S'_{min} = K_{sep} z_{eye}, \quad K_{sep} = \max \frac{S_{min}}{z_{eye}} = 1.37 \times 10^{-5}$$

The conservative fit is close, at worst roughly ten percent too large in the far field. Importantly, this model may be of practical use in applications, because programmers already manage level of detail as a linear function of z_{eye}.

Both spatial and complementary-depth separation are well approximated by a linear function of z_{eye}, so this approach will provide a good fit regardless of which separation (if either) dominates. System constant factors, such as b and the details of mapping arithmetic, can easily be incorporated in K_{sep}. Representative values of the variable factors w and fov can be expanded into a small 2-D table of K_{sep} values. There is no need to incorporate n, because complementary z-buffering, unlike traditional z-buffering, is invariant to n.

5. System Designer Guidance

The parameters of Figure 4 correspond to those specified by Direct3D® 10 [Bly06]. For the chosen application-specified parameters ($fov = 90$ and $w = 1024$) it is clear that S_{min} is dominated by spatial precision. This remains true even for reasonable variations of fov and w. The designers of Direct3D 10 could have reduced S_{min} by increasing b (to 10 or perhaps even 12 bits); or they could have reduced z_{buf} precision without increasing S_{min}.

More broadly, the fact that $\log_2 w$ and b contribute equally to spatial separation suggests that graphics system designers consider linking these two values (e.g., $b = 20 - \lfloor \log_2 w \rfloor$). This would eliminate the dependence of S_{min} on viewport dimensions, at the (potentially low) cost of allowing the binary point within existing x_{win} and y_{win} hardware registers to be moved.

6. Conclusion

We have provided a concise and intuitive analysis of minimum triangle separation for correct occlusion, showing that, and how, it is affected by depth and spatial representations, application-specified parameters, and mapping arithmetic. With this understanding graphics system designers can allocate numeric precision with a fuller understanding of the consequences, and application programmers can more confidently predict the triangle separation that is required to ensure correct z-buffer occlusion. Should industry go forward with per-system specification of S_{min}, our approach provides the required infrastructure, and strongly suggests that agreement be built around a model of linearly increasing S_{min} as a function of z_{eye}. Uncertainty volume analysis might also provide insight into other important problems, such as the evaluation and optimization of shadow mapping algorithms.

Our analysis does not include precision loss due to clipping, which could be significant for extremely large triangle projections. We also have not considered arithmetic error due to vertex-shader and geometry-shader arithmetic, in part because such analysis is heavily application dependent. On the other hand, our analysis of mapping error yields worst-case intervals that are approached but never reached in practice. Determining if, and how, these omissions and conservative mapping error bounds compensate each other remains as future work.

Finally, our approach ensures correct occlusion if no triangle separation within a model is smaller than the maximum S_{min} throughout the eye-coordinate volume in which the model is rendered. But per-vertex optimization of the separation between large adjacent triangles is not specified, due to the non-linear relationship of error interpolation in window coordinates (during rasterization) to the corresponding errors in eye coordinates.

7. Acknowledgements

We thank Amar Patel, David Blythe, Jim Kajiya, and Marcel Gavriliu for their support during this work.

References

[AF98] AKELEY K., FORAN J.: Apparatus and method for selectively storing depth information of a 3-d image. US Patent, 1998.

[Ake90] AKELEY K.: The hidden charms of the z-buffer. *IRIS Universe 11* (1990), 31–37.

[Bly06] BLYTHE D.: The direct3d 10 system. *ACM Trans. Graph. 25*, 3 (2006).

[Hig96] HIGHAM N. J.: *Accuracy and Stability of Numerical Algorithms*. SIAM, 1996.

[LJ99] LAPIDOUS E., JIAO G.: Optimal depth buffer for low-cost graphics hardware. In *HWWS '99: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (1999), pp. 67–73.

[LJZW01] LAPIDOUS E., JIAO G., ZHANG J., WILSON T.: Quasi-linear depth buffers with variable resolution. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (2001), pp. 81–86.