1 Introduction

Modern graphics hardware employs z-culling, or early-z culling, as a conservative z-test before the actual per-fragment z-test [Kilgariff and Fernando 2005]. Z-culling is used to avoid executing expensive fragment shaders for invisible fragments, and minimize reading, testing, and updating z-buffer entries. However, these optimizations need to be disabled for shaders that modify the z value of a fragment (depth replace). Such shaders are rapidly gaining in importance, however, especially in the context of per-pixel displacement mapping such as relief mapping [Policarpo et al. 2005]. We propose a slight modification to graphics APIs and hardware drivers that would allow to retain many of the z-cull optimizations for a large class of depth-modifying shaders, thus including displacement shaders. The basic idea is for shaders to specify a non-negative depth offset that is added to the z value of the fragment. While the exact z value is not known before the shader is executed, the lower bound is sufficient to perform all z-culling tests. Updates to the z-cull buffer are limited, but not worse than with texkill shaders.

Z-culling usually works in conjunction with a tiled z-buffer, and maintains z-max and/or z-min values for each tile of pixels [Hasselsgern and Akenine-Möller 2006]. In the following, we assume the z-test function to be set to GL_LESS, but adaptation to other z-test functions is straightforward. Initialization is done as \( z_{min} = z_{max} = z_{far} \). During rendering, these values are updated and both decrease monotonously. Z-max culling allows to reject (z-fail) an entire incoming tile when \( z_{src-min} > z_{max} \), where \( z_{src-min} \) is the minimum \( z \) of the entire incoming tile, and \( z_{max} \) is the maximum \( z \) of the tile in the tiled z-buffer. In the z-fail case, the fragment shader is not executed, the z-buffer need not be read or updated, and the per-pixel z-test can be skipped. Although it is not strictly required, any update of \( z_{max} \) improves performance, because its value always decreases and thus subsequent tiles are more likely to be rejected. Complementarily, Z-min culling allows to accept (z-pass) an entire incoming tile when \( z_{src-max} < z_{min} \). In the z-pass case, the z-buffer need not be read and can be updated without performing the per-fragment z-test. In principle, updates of \( z_{min} \) decrease performance because subsequent tiles are less likely to be accepted. However, keeping \( z_{min} \) up to date is mandatory for rendering a correct image. Killing fragments (texkill operations) can be considered identical to the case of \( z_{src-min} = z_{src-max} = z_{far} \). The basic idea is for shaders to specify a non-negative depth offset that is added to the z value of the fragment. While the exact z value is not known before the shader is executed, the lower bound is sufficient to perform all z-culling tests. Updates to the z-cull buffer are limited, but not worse than with texkill shaders.

2 Preserving the Validity of Z-Culling

Per-pixel displacement mapping usually uses a ray-casting algorithm in the fragment shader to find the intersection of the view ray with the displaced surface, which is given as a height field. Depending on the algorithm, this surface can be rendered as being either always above the base polygon, or always below it. In the latter case, the surface is always behind the actual geometry, which implies that the fragment shader only increases the z value of the fragment.

The most important idea of our proposal is to observe which z-cull optimizations can be preserved if the fragment shader adjusts the z value of the polygon in one direction only, either closer to the far plane or to the near plane. Both z-max and z-min culling consist of two parts: testing the current tile, and updating the z-cull buffer for future tests. Both tests and z-max buffer updates can be temporarily suspended, but z-min culling has to be disabled for the rest of the frame if a z-min buffer update is missed. Given that z-max culling only uses \( z_{src-max} \) for comparison, the optimization can be preserved when the shader only increases \( z \). In this case, only \( z_{src-max} \) may change during shader execution, whereas \( z_{src-min} \) stays untouched. This allows the z-max test and the z-min update to stay enabled. Note that this can be considered to be the same case as shaders with texkill operations, which should allow actual implementation with very minor modifications to graphics APIs and drivers. It is important to note that z-min needs to be updated in order for the test to stay correct, but since the shader only increases \( z \) this can be done from the base polygon \( z \). Contrarily, z-max updates are not mandatory, although they improve culling efficiency. Unfortunately, if the shader only decreases \( z \), both z-max culling and z-min culling must be disabled. Because the z-min buffer would need to be updated with \( z_{min} = z_{near} \) in order to stay valid, z-min culling even stays disabled for the rest of the frame.

Another possible option for further optimization would be to also specify an upper bound for the possible increase of the z value of the fragment. This upper bound is known in most displacement mapping techniques and could be specified via a mechanism similar to the specification of a polygon offset (z bias), or per vertex. However, we assume that implementing this additional optimization in current graphics hardware would be relatively complicated. Furthermore, z-max should be updated regularly in order to increase culling efficiency and obtaining a tighter bound earlier, as determined by a maximum z offset, would overall only yield minimal performance improvements.

In summary, the major proposed modification to let the shader specify whether it guarantees depth replace to be non-negative with respect to the depth of the base polygon would add very little complexity to the rendering pipeline and could be performed at the driver or firmware level. It would, however, significantly increase performance of a large class of fragment shaders, specifically including per-pixel displacement mapping.

References

