Finding mold removal directions using graphics hardware

Rahul Khardekar*
University of California, Berkeley

Sara McMains †
University of California, Berkeley

1 Introduction

In molding and casting manufacturing processes, molten raw material is shaped in molds from which the resulting part must be removed after solidification. We consider two part, rigid molds in which the mold halves are separated in opposite directions (the positive and negative casting directions). For a part to be castable in a given direction, it should be free from undercut features which make it impossible to separate the mold halves from the part when translated along the casting directions without colliding with the part. We propose the use of programmable graphics hardware to test the castability and facilitate computer aided design of such parts.

2 Determining castability in a direction

A part is castable in a given direction if and only if, when projected orthographically on a plane normal to that direction, there is no overlap between faces whose normals subtend an angle less than 90° with the positive casting direction. (For all the tests described below, we use orthographic projection.) In our two pass castability algorithm, we place the eye point above the part along the positive casting direction. In the first pass, we render the front faces. In the second pass, we zero the frame buffer but keep the old z-buffer, setting the depth test function so that only the front faces invisible in the first pass are rendered. The part is castable if and only if no pixels are rendered in the second pass, which we check using an occlusion query.

We then use depth textures to highlight the (portions of) faces that are not castable in the given direction so that the designer can make the necessary changes to the part geometry. Our approach is analogous to using shadow mapping to display (portions of) faces in shadow when the object is illuminated by two lights located at infinity in the positive and negative casting directions. We generate two depth textures by rendering the part looking at it from the positive and negative casting directions. We then use a vertex program to transform each polygon by the viewing transform for the two casting directions in turn, followed by a fragment program to check if the transformed depths for each fragment are greater than the depth values stored in the respective depth textures. If both are greater, we highlight that fragment to indicate to the designer that there is an undercut on that section of the surface.

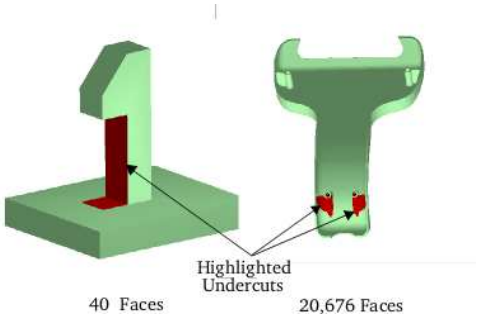

Figure 1: *Sample parts with undercuts highlighted in dark red. The casting direction is vertical for both the parts.*

*rahul@me.berkeley.edu
†mcmains@me.berkeley.edu

3 Determining multiple castable directions

During the detail design stage for a molded or casted part, the casting direction needs to be known a priori (in order to taper vertical facets for mold releasability, add bosses and ribs for rigidity, etc.). Thus the above algorithm is appropriate for interactive inspection of castability during detail design. During conceptual design, however, we would like to be able to generate multiple feasible casting directions for a given part geometry. To do so efficiently, we make use of a theoretical result by [Ahn et al. 2002] which proves that all combinatorially distinct casting directions correspond to cells in an arrangement of great circles on a Gaussian sphere. Every face normal and normal of the triangle formed by every edge-vertex pair of the part generates a great circle in the arrangement. These great circles correspond to the directions where a part face changes from front-facing to back-facing (directions contained in the plane of the face), and directions where a projection of one part face potentially changes from occluding to not occluding (or vice versa) another part face (directions contained in the planes through an edge-vertex pair from each). We reduce the number of great circles by merging adjacent coplanar faces and omitting redundant and non-interacting edge-vertex pairs. We project the remaining great circles on a plane tangent to the sphere to obtain an arrangement of lines. We subdivide the line arrangement in a quad-tree to obtain a small number of lines per quad-tree leaf node. We then construct the arrangement in each leaf node by rendering a half-space for each line with a different color, blending the colors in the frame buffer. We select a random sequence of 1,024 pixel locations in each leaf node and select the points having different colors (corresponding to distinct cells in the arrangement). The directions corresponding to these points, along with face normal and axis directions (which are good heuristic candidates), are tested for castability using the algorithm presented in Section 2.

4 Results

Both algorithms were tested on sample parts, including those pictured in the figure, using a QuadroFX 3000 GPU. The two pass algorithm obtained speeds in the range of 18,200 to 1,040 directions per second for the parts pictured, with 40 to 20,676 faces respectively. This is a speed up of 200 times as compared to running the same algorithm on a GeForce2 GPU where the CPU (Athlon 1.8 GHz) did the rendering, but this is not a fair comparison since a different implementation would probably be more efficient on the CPU. The algorithm for finding multiple feasible directions took 0.42 seconds for a part with 28 faces in which 1,497 different candidate directions were tested. For a part with 328 faces, it took 2 minutes 18 seconds in which 491,010 directions were tested. Please refer to `http://kingkong.me.berkeley.edu/~rahul/gpgpu/index.html` for more details about the results.

References

AHN, H.-K., DE BERG, M., BOSE, P., CHENG, S.-W., HALPERIN, D., MATOUŠEK, J., AND SCHWARZKOPF, O. 2002. Separating an object from its cast. *Computer - Aided Design 34*, 547–559.