ADAPTIVE LEVEL OF DETAIL WITH OCCLUSION CULLING

Hermann Birkholz
University of Rostock
Albert-Einstein-Str. 21, 18059 Rostock, Germany

Stefan Rahn
University of Rostock
Albert-Einstein-Str. 21, 18059 Rostock, Germany

Keywords: Level of Detail, Occlusion Culling.

Abstract: Many techniques have been developed in order to accelerate the visualization of large triangle meshes. Level of Detail techniques can be used to create view-dependent approximations of wide range scenes with low occlusion, such as landscapes. For highly occluded scenes there exist many occlusion culling techniques, which discard occluded parts of the scene before rendering. This can drastically speed up the visualization of such scenes but will not improve rendering of wide non-occluded scenes. In this paper we will combine both acceleration methods. We present a new approach, which combines the benefits of Level of Detail rendering and of Occlusion Culling, in order to minimize their drawbacks. The technique adds occlusion as a view-dependence criterion for Level of Detail rendering and is even able to optimize the refinement of self occluding meshes.

1 INTRODUCTION

Very large triangle meshes can easily be acquired by current 3d-scanning hardware. Meshes consisting of millions of triangles exceed the capabilities of modern graphics hardware in interactive rendering. Interactive frame rates can be achieved only by a reduction of the number of triangles before rendering. Level of Detail (LOD) algorithms create hierarchical descriptions of meshes by iteratively simplifying them locally. All local simplifications are stored together with an error estimation and thus form the hierarchy. This enables view-dependent approximations of the original mesh. The approximation process uses the hierarchy in order to refine the parts of the mesh that are situated within the view frustum and that do not face away from the viewer. The mesh complexity is locally adopted to the viewpoint distance. All these parameters enable interactive frame rates for large triangle meshes but produce poor approximations for highly occluded views. Due to the lack of occlusion information, occluded parts of a mesh are refined in the same way as non-occluded ones. This problem could be solved by an additional occlusion criterion during the approximation process, which forbids further refinement of occluded parts of the mesh. This paper will show recent work regarding this problem and presents an new integration technique for LOD and occlusion culling.

2 RELATED WORK

The technique that is presented in this paper was influenced by miscellaneous papers, which are reviewed in this section.

2.1 Level of Detail Rendering

Many algorithms have been proposed to create LOD hierarchies, which can be used to approximate triangle meshes but do not consider occlusion. Hoppe (Hoppe, 1997) shows how to use his earlier introduced ”progressive meshes”(Hoppe, 1996) for view-dependent rendering. The progressive meshes store a edge-collapse sequence, which is used for the reconstruction of the original mesh. In each simplification step the edge whose simplification causes the least error is collapsed to a new vertex. This operation removes one vertex and two triangles from the surface. View-frustum and backface culling are supported with bounding-spheres and normal-cones. Another approach (Xia and Varshney, 1996) uses ”Merge Trees”, in order to present the LOD hierarchy.

Birkholz H. and Rahn S. (2006).
ADAPTIVE LEVEL OF DETAIL WITH OCCLUSION CULLING.
In *Proceedings of the First International Conference on Computer Graphics Theory and Applications*, pages 279-284
DOI: 10.5220/0001351302790284
Copyright © SciTePress

The ”Merge Tree” is constructed by repeated edge-collapses in the original mesh. Backface and view-frustum culling is supported with normal-cones and bounding spheres, again. Garland and Heckbert (Garland and Heckbert, 1997) extended the edge-collapse hierarchies to vertex-contraction hierarchies, which can also merge vertices without a shared edge. Furthermore they introduced a new quadric error metrics in their paper. Newer techniques enable rendering of very large LOD hierarchies from external memory. Hoppe presented a hierarchical terrain management (Hoppe, 1998) that was extended to arbitrary meshes by Prince (Prince, 2000). An external memory approach based on vertex-clustering was presented by Lindstrom (Lindstrom, 2003). With the ”adaptive TetraPuzzles” (Cignoni et al., 2004) a hierarchy of tetrahedral cells is constructed from the bounding box around the mesh. Each tetrahedral cell is associated with a precomputed simplified part of the original model. The tetrahedra-hierarchy is adaptively refined and replaced with the precomputed geometry of the tetrahedral cells.

2.2 Occlusion Culling

Another method to reduce the number of triangles that has to be processed by the graphics hardware is to detect and to cull occluded regions. In highly occluded scenes most triangles will not be visible and the frame rate would significantly benefit from occlusion culling. For wide and less occluded views however, there will be no improvements.

There exist several different approaches to determine the occluded parts of a scene. Some divide the scene into disjoint cells and compute a set of potentially visible triangles (PVS) for each cell in a preprocess (Chrysanthou et al., 1998; Durand et al., 2000; Schaufler et al., 2000). While rendering, only triangles of the actual PVS are taken into account. Due to the preprocess such techniques are only suited for the visibility of static scenes.

Other algorithms determine occlusion in screen-space and can be accelerated by graphics hardware. The hierarchical z-buffer visibility, presented by Greene (Greene et al., 1993), is meanwhile available in current graphics hardware and enables fast z-tests for bounding objects. The z-buffer values are hierarchically merged 4 by 1 and the deepest value is chosen for each group. This allows to cull small objects with only few z-tests in the upper hierarchy levels. Greene recommends to store the scene in an octree, whose cells are visibility-tested against the hierarchical z-buffer and only processed further when they are visible. The ”hierarchical occlusion maps” used by Zhang (Zhang et al., 1997) allow hardware supported occlusion culling with so-called occlusion maps. He uses alpha-map hierarchies, whose opacity values represent the opacity of selected occluders. Hierarchy levels of the occlusion maps are generated by subsampling the previous hierarchy level 4 by 1 (hardware accelerated). These map hierarchies can then be used in order to determine the visibility of occludees. Screen space techniques have the advantage of build-in occluder fusion. Because all occluders share the screen space, they are merged implicitly.

Finally there exist algorithms that cull occluded objects in the object-space. Hudson (Hudson et al., 1997) constructs shadow frusta from all occluders and tests the occludees against them. To accelerate the selection of proper occluders, a preprocess assigns a set of potentially good occluders to each viewpoint. Coorg (Coorg and Teller, 1997) also determines potentially good occluders for each viewpoint in a preprocess. The occlusion test is then done with the help of supporting and separating planes.

2.3 Integration of LOD and OC

In order to overcome the disadvantages of both Level of Detail and occlusion culling, they were combined in some approaches. Andujar (Andujar et al., 2000) introduces ”Hardly-Visible Sets” in order to differentiate partial occluded objects in the scene. He suggests the use of LOD approximations as occluders to determine occlusion and to use coarser approximations for the rendering of ”Hardly-Visible” objects in the scene. He uses discrete LODs for the approximation and thus can guarantee neither image quality nor the amount of geometry, which ist sent to the graphics hardware.

An integration of approximate visibility and continuous LOD was presented by El-Sana (El-Sana et al., 2001). They divide the space with a uniform grid and store an opacity value for each cell, which depends on the area of the projected geometry in the cell. The split- and merge-criteria for the nodes in the LOD hierarchy is now expanded by occlusion. For each node a visibility value is estimated by tracing the ray between viewer and hierarchy node through the grid. The opacity values of the traversed cells are combined, in order to determine which nodes to split or to collapse. Due to the approximate estimation and the discrete grid, the visibility might appear incorrect. Furthermore, the traverse time for the visibility test could consume much time in hardly occluded scenes.

The approach that is presented in this paper, combines view-dependent LOD and occlusion culling in another way. It uses coarse approximations of the final mesh itself to determine occlusion during the refinement.

3 OCCLUSION CONTROLLED REFINEMENT

In this section we describe our algorithm for the integration of view-dependent LOD and occlusion culling. Multiple LOD objects can be approximated in one scene and be freely moved in the scene. Visibility data is extracted online and does not demand precomputed data.

We use a edge-collapse hierarchy, which stores bounding volumes and normal cones for each node in the hierarchy. The occlusion tests are done in the z-buffer in image space resolution. Occluder fusion is enabled implicitly, due to the use of the z-buffer. Depth values of different occluders always merge to a common occluder in z-buffer. We assume that the LOD extraction process takes much more time of the complete rendering procedure, than the time for the actual rendering of the extracted geometry. This allows us to render different versions of the occluders with increasing detail into the z-buffer without significant effects to the frame rate. Our LOD extraction is controlled by the number of triangles in the approximation. The extraction stops when a given triangle threshold is reached. The refinement always starts with a coarse base mesh that is refined by vertex-split operations. The refinement sequence is controlled by the viewpoint distance and the visibility. The visibility of refinement candidates is determined with view-frustum culling (bounding spheres), backface culling (normal cones) and occlusion culling. Occlusion tests are only initiated when the other culling tests fail. All visible vertices are placed in a refinement priority queue. During the refinement process, always the first vertex is removed from the queue, processed, and its child-vertices are added when they are visible.

3.1 Occlusion Estimation

Occlusion culling is implemented with hardware occlusion queries. These queries test given geometry against the z-buffer of the graphics hardware without modifying it and return the number of pixels that would be visible if the geometry had been really rendered. A coarse approximation of the mesh itself is used as the occluder. It is rendered to the z-buffer and the bounding volumes of vertices, which will potentially be split, are tested against it. As the first occluder, a very coarse approximation of the mesh is chosen, which of course was produced without occlusion culling. For a better adoption this occluder mesh can be updated several times during the refinement process. As bounding volumes for the vertices, we use simple shapes, that conservatively approximate all refinement levels, which are stored in the subtree of the vertex.

3.2 Algorithm

The complete refinement process consists of the following steps:

- Creation of a coarse approximation without occlusion culling.
- Render the coarse occluder to z-buffer.
- While the triangle threshold is not exceeded, test the visibility for the next refinement step and execute it, if it is visible. Otherwise proceed with the next refinement step.
- Render the final mesh to the frame-buffer.

In the refinement loop, the occluder should be updated periodically for a better approximation. On each update, all vertices culled due to occlusion should be tested against the new occluder again.This ensures that the geometry, which was occluded by the coarse approximated occluder can be refined, if it is visible in the final mesh.

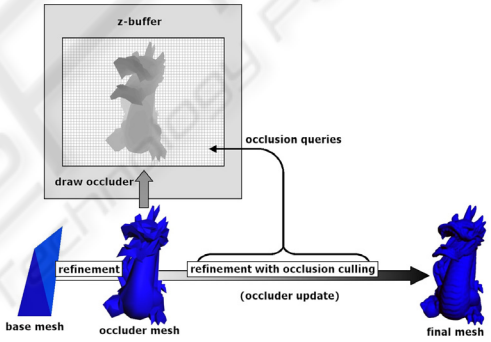

Figure 1: Refinement process.

The whole refinement process is shown in figure 1, where a ”Dragon” mesh is refined to a coarse level. This coarse representation is used as occluder in the further refinement steps. The final mesh will have a much lower triangle resolution in occluded regions, compared to the visible parts, after the refinement is done.

3.3 Reduction of Occlusion Queries Latencies

Due to the time necessary for the occlusion test, the algorithm would make the refinement very slow. Especially large bounding volumes (in screen space) can require long testing time. This latency-time can be reduced by making use of parallel threads on CPU and GPU. We can send multiple queries to the GPU, which are queued there and are processed one by one. We choose the number of occlusion queries that are

queued in the GPU large enough to significantly decrease latency times. When one occlusion result is read from GPU, the further processing of the vertex is determined with it. A visible bounding volume means that the occlusion queries for the bounding volumes of the child vertices are queued in the GPU. When the number of queued queries on the GPU becomes too small, we simply add queries from the global refinement queue.

To improve the frame rate, we do not send all queries to the graphics hardware. To decrease the number of occlusion queries, we only send queries for odd hierarchy levels. This hardly lowers the refinement quality but saves half of the occlusion queries.

4 RESULTS

The LOD refinement with occlusion culling has been tested with different meshes. We determined the ratio between visible and invisible triangles in the scene, in order to measure the efficiency. The scene can be composed of multiple LOD hierarchies, in order to test the occlusion among meshes. All test scenes were approximated with a number of 10,000 triangles. We used two different occluder steps during the refinement process. After 300 refinement steps, a very coarse occluder is rendered to the z-buffer and after 2500 steps, it is updated with a finer version. The usage of further occluders was tested, but it did not improve the distribution of triangles in the visible and invisible areas. The measured frame rates for the LOD rendering without occlusion culling were between 26 and 27 fps, while the use of occlusion culling decreased the frame rate to slightly above 15 fps (due to the occlusion query overhead).

The LOD hierarchies of all meshes were created by half-edge collapses using the Quadric Error Metric (Garland and Heckbert, 1997). The meshes that were used in the test scenes can be found in table 1.

Table 1: Test meshes.

Name	Armadillo	Armadillo Group	Rough Planet
Vertices	172,974	518,922	16,777,218
Triangles	345,944	1,037,832	33,554,432
Disc Size	19,7 MB	59,1 MB	1920 MB

The different bounding volumes were tested with the ”Armadillo” mesh. Due to the queued parallel processing, the complexity of the tested bounding volumes did not affect the frame rate. The box is rendered with 12 triangles while the sphere is approximated with 32 triangles and the cylinder with 64 triangles. The best ratio between visible and invisible triangles was achieved with a cylindrical bounding volume. The cylinder is oriented along the surface normal of the according vertex. All geometry that is associated with the vertex is enclosed by the cylinder. The box and the sphere volume always resulted in more conservative approximations and poorer visibility ratios.

A comparison between view-dependent LOD with and without occlusion culling can be found in table 2. For each view the the number of visible triangles without (VT_{LOD}) and with (VT_{OC}) occlusion culling is shown together with the reached improvement ratio (IR). The total number of triangles which is used to approximate the whole scene is always 10,000. The results show significant improvements

Table 2: Test meshes.

Name	VT_{LOD}	VT_{OC}	IR
Armadillo	2,277	3,642	59,95%
Armadillo group	2,231	3,733	67,32%
Planet full	3,546	4,781	34,83%
Planet near	1,888	4,623	144,86%

in the distribution of triangles. Many more triangles are used to approximate the visible parts of the scene when occlusion culling is integrated. Of course the improvement ratio depends on the amount of occlusion in the view to be rendered. Views with low occlusion, such as the ”Planet full” view (Figure 2) show only little improvement.

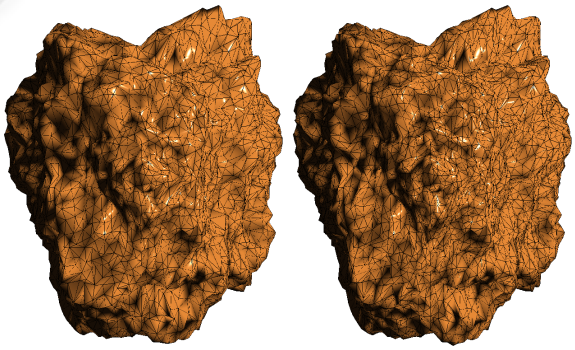

Figure 2: Full view of the ”Rough Planet” mesh, without (left) and with (right) occlusion culling.

The relatively high result in this example, however, can be reasoned with the inefficient backface culling. Due to the rough surface, the backface culling is hardly able to influence the refinement process. The normal cones of the mesh enclose large angles even in the higher levels of the hierarchy and thus will be unable to support visibility culling. Occlusion

culling detects the surface regions on the backside and culls it efficiently. For smoother surfaces this difference would narrow itself. A view of the ”Armadillo” mesh, for instance, shows only improvements below 4% when there is no self-occlusion in view.

Views with heavy occlusion such as in ”Planet near”, show enormously high improvements. If many parts of the mesh are occluded, LOD techniques without occlusion culling distribute the available triangles evenly in both visible and invisible regions. An early detection of occluded regions during the refinement process will influence the arrangement of triangles, and cause higher detail in parts of the view that are visible.

Figure 3 shows a comparison of the view with and without occlusion culling. Because the hill in the front occludes most geometry in the view, standard view-dependent LOD methods distribute many of the triangles to invisible regions.

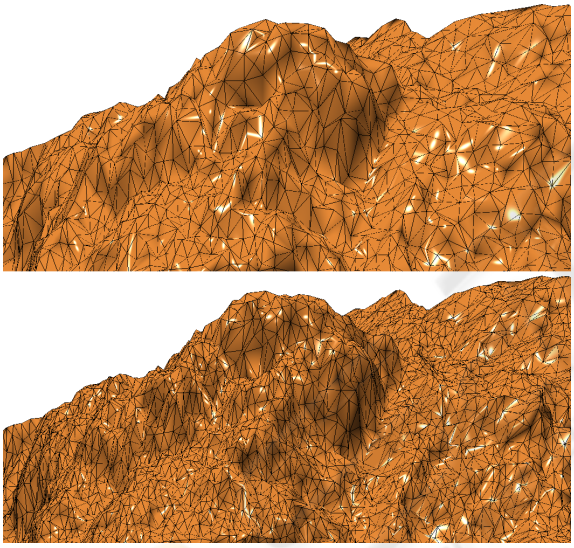

Figure 3: Near view of the ”Rough Planet” mesh, without (top) and with (bottom) occlusion culling.

Our approach however detects the occluded regions and uses only few triangles to approximate them. A birds eye view in figure 4 illustrates the effect of the occlusion culling. The left image corresponds to the upper image of figure 3 and the right image to the bottom image of figure 3.It is noticeable that the regions that are occluded by the hill in the front of the view (yellow lines), are approximated much coarser than the visible parts.

A view on the ”Aramdillo” mesh as listed in table 2 is shown in figure 5. The hand in the front occludes a large region of the mesh and allows us to distribute more triangles in the visible regions of the view.

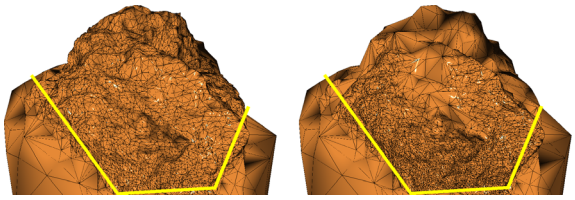

Figure 4: Effect of occlusion culling in the approximation shown in figure 3.

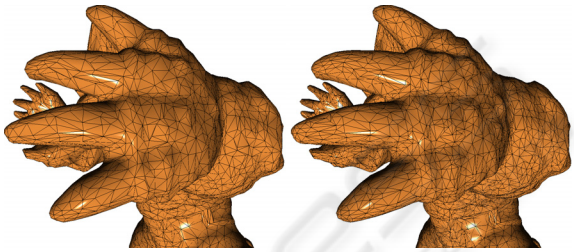

Figure 5: Views of the ”Armadillo” mesh without (left) and with (right) occlusion culling.

Our last example view is shown in figure 6. Three instances of the ”Armadillo” mesh are placed one after another. The instance in the front occludes most geometry of the two other instances such that many triangles would be wasted without occlusion culling.

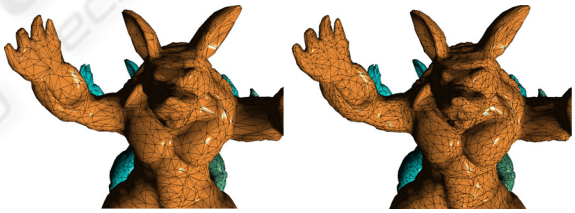

Figure 6: Views of the ”Armadillo Group” without (left) and with (right) occlusion culling.

The effect of the occlusion culling is illustrated in figure 7. The regions that are occluded in figure 6 are coarsely approximated only in the right image. The saved triangles are distributed to the visible parts of the meshes.

5 CONCLUSION

In our paper we presented a new way to integrate occlusion culling and view-dependent LOD. The results show that significant improvements in the distribution of triangles are possible. Views that contain much occlusion can drastically increase the amount of visible

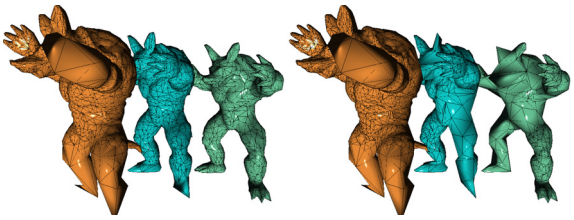

Figure 7: Side-views of the "Armadillo Group" approximations in figure 6.

triangles. Even in situations with less occlusion, our approach can support the culling of faces that are oriented away from the viewer but cannot be culled with backface culling.

The price for the improvement in the rendering-quality, however, is a 42% drop in the frame rate. The approximately 5000 occlusion queries, which have to be done for a reconstruction with 10,000 triangles, consume a large amount of the frame time, although they mainly run in parallel to the refinement process. Our next work will be focussed on the acceleration of the rendering process.

Another way to increase the frame rate would perhaps be the use of time coherency. A frame to frame update of the approximation would likely demand less occlusion queries, when the view moves slowly.

REFERENCES

Andujar, C., Saona-Vazquez, C., Navazo, I., and Brunet, P. (2000). Integrating occlusion culling and levels of details through hardly-visible sets.

Chrysanthou, Y., Cohen-Or, D., and Zadicario, E. (1998). Viewspace partitioning of densely occluded scenes. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry*, pages 413–414, New York, NY, USA. ACM Press.

Cignoni, P., Ganovelli, F., Gobbetti, E., Marton, F., Ponchio, F., and Scopigno, R. (2004). Adaptive tetrapuzzles: efficient out-of-core construction and visualization of gigantic multiresolution polygonal models. *ACM Trans. Graph.*, 23(3):796–803.

Coorg, S. and Teller, S. (1997). Real-time occlusion culling for models with large occluders. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 83–ff., New York, NY, USA. ACM Press.

Durand, F., Drettakis, G., Thollot, J., and Puech, C. (2000). Conservative visibility preprocessing using extended projections. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 239–248, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

El-Sana, J., Sokolovsky, N., and Silva, C. T. (2001). Integrating occlusion culling with view-dependent rendering. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 371–378, Washington, DC, USA. IEEE Computer Society.

Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Greene, N., Kass, M., and Miller, G. (1993). Hierarchical z-buffer visibility. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 231–238, New York, NY, USA. ACM Press.

Hoppe, H. (1996). Progressive meshes. *Computer Graphics*, 30(Annual Conference Series):99–108.

Hoppe, H. (1997). View-dependent refinement of progressive meshes. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 189–198, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Hoppe, H. (1998). Smooth view-dependent level-of-detail control and its application to terrain rendering. In *VIS '98: Proceedings of the conference on Visualization '98*, pages 35–42, Los Alamitos, CA, USA. IEEE Computer Society Press.

Hudson, T., Manocha, D., Cohen, J., Lin, M., Hoff, K., and Zhang, H. (1997). Accelerated occlusion culling using shadow frusta. In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, pages 1–10, New York, NY, USA. ACM Press.

Lindstrom, P. (2003). Out-of-core construction and visualization of multiresolution surfaces. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 93–102, New York, NY, USA. ACM Press.

Prince, C. (2000). Progressive meshes for large models of arbitrary topology.

Schaufler, G., Dorsey, J., Decoret, X., and Sillion, F. X. (2000). Conservative volumetric visibility with occluder fusion. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 229–238, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Xia, J. C. and Varshney, A. (1996). Dynamic view-dependent simplification for polygonal models. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 327–ff., Los Alamitos, CA, USA. IEEE Computer Society Press.

Zhang, H., Manocha, D., Hudson, T., and Kenneth E. Hoff, I. (1997). Visibility culling using hierarchical occlusion maps. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 77–88, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.