Spacetime Catmull Recursive Subdivision Facilitated with Occlusion Culling

Andreas Genz
FB3, Grafische Datenverarbeitung, Universität Bremen
Linzerstr. 9a, 28359 Bremen, Germany
genz@acm.org

ABSTRACT

We describe an extension and a generalization of the Catmull recursive subdivision algorithm: first, an image-based occlusion culling stage is added; second, all rendering stages, that is, view-frustum and occlusion culling, subdivision of geometric primitives into micropolygons, and rasterization, are performed in spacetime.
Operating in spacetime allows to exploit temporal coherence in animated scenes.

Keywords

image generation, rendering, spacetime rendering, occlusion culling, hidden surface removal, performance

1. INTRODUCTION

The following rendering algorithm is optimized for very complex dynamic scenes preferring a procedural description of geometry. Main objective is to produce and process geometry only if it affects the synthesized animation. We choose the Catmull recursive subdivision algorithm as a foundation, because it was designed with procedurally described geometry in mind and it is the ancestor and back-bone of the REYES architecture, which is in use for most movie productions these days. We insert an occlusion culling stage with a summed-area table as its occlusion representation. Also, we generalize the complete rendering pipeline to operate in spacetime. The occlusion representation becomes a summed-volume table then.

2. PREVIOUS WORK

In 1988, Andrew Glassner published implementation results of accomplishing ray tracing in spacetime [Gla88]. In 1999, Damez and Sillion have shown how to perform radiosity calculations in spacetime [DS99]. And in 2003 Havran et al. picked up the task of ray tracing in spacetime again [HDMS03]. But to our knowledge, no one ever tried to realize a spacetime z-buffer rendering pipeline.

SHORT papers proceedings, ISBN 80-903100-9-5
WSCG'2005, January 31-February 4, 2005
Plzen, Czech Republic.

An extensive overview on general occlusion culling techniques is given by Cohen-Or et al. [COCSD03]. Image-based occlusion culling was pioneered by Ned Greene [GKM93]. Variants of his hierarchical z-buffer are used in today's hardware [AMN03] and in the prman renderer—Pixar's implementation of the REYES architecture—to reduce the number of shading calculations [AGB99].
Another candidate for an image occlusion representation is the summed-area table, which is in common use for texture mapping today [Cro84]. It was used for volume rendering by Huang et. al. [HCSM00], for occlusion culling by Ho and Wang [HW99], and the author [Gen01].
Catmull introduced in his dissertation a rendering algorithm [Cat74] fetched up by Lucasfilm's rendering group to realize the REYES architecture [CCC87, AGB99]. This architecture may be a good candidate for future realtime rendering pipelines implemented in hardware [OKTD02].

3. THE CATMULL RECURSIVE SUBDIVISION ALGORITHM

Z-buffer algorithms hold the advantage to address surfaces independently of each other.
Subdivision splits an object into more than one smaller objects, the sum of the smaller objects represents the original one, and the smaller objects can be handled independently. Subdivision of the scene space (for example by octrees), of the raster image space into buckets or a hierarchy, of the camera space into layers along

the z-axis as will be shown later, and of geometric primitives in different spaces are plausible.
The REYES architecture takes this two concepts—z-buffer and subdivision—devised by Catmull [Cat74] and extends them by incremental methods to dice each surface into micropolygons, by shading micropolygons with texture filtering, and by subpixel rasterization with jittered samples for antialiasing [CCC87].

Algorithm C (*Catmull's recursive subdivision algorithm*). Given: camera, surfaces. Find a raster image of the scene

For each surface:
C1. Compute camera-space axis-aligned bounding box.
C2. If bounding box is entirely outside viewing volume:
drop surface.
C3. Else: project bounding box into image space.
C4. If area of bounding rect larger than one pixel:
C5. subdivide surface.
C6. Else: rasterize surface, z-buffer visibility test.

Catmull's recursive subdivision algorithm can be seen as the first stage (splitting) of the REYES pipeline plus a z-buffer. Surfaces are subdivided in parameter space (**C5**) until the resulting face becomes small enough, such that the area of its bounding rect is smaller than one pixel (**C4**). Then, one point inside the surface is tested against the z-buffer for visibility, and its color is written to the color buffer if it passes the test (**C6**). During subdivision, if an surface is outside the viewing volume, it is dropped. For purpose of efficiency and code minimalism, tests are not done on the surface directly, but on its bounding box. Therefore, every face must have the following properties to be used: it must have a well defined bounding box and subdivision procedure.
The Catmull recursive subdivision algorithm accomplishes hierarchical view-frustum culling and level-of-detail implicitly. Ideally, the whole scene can be described as one surface that is subdivided during the rendering process, so that unnecessary geometry is not stored and processed at all.
Here we make use of this algorithm, because it is well suited for a simple implementation to measure the number of required recursive subdivisions. Our goal is to reduce this number by occlusion culling and by exploiting temporal coherence, so that the following pipeline stages of a fictional REYES architecture would not be fed with too many geometric primitives.

4. SUMMED-AREA TABLE OCCLUSION CULLING

For scenes with high depth-complexity processed by traditional z-buffer visibility algorithms, a huge amount of scene geometry entirely hidden by other geometry is rendered. Although, hidden geometry does not contribute to the requested image if we restrain light models to be local.
To overcome this, we add a simple occlusion culling method to the Catmull recursive subdivision algorithm. The algorithm is slightly variated by exchanging **C3** with **C2**; the bounding rect given by the projected bounding box is used for view-frustum culling, occlusion region estimation, and area measurement (subdivision criteria).

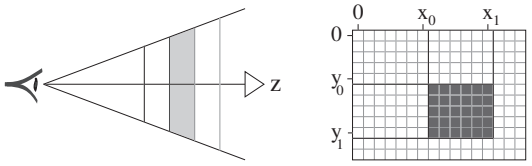

Figure 1: Depth layers and summed-area table in raster space

Our image based occlusion culling algorithm using a summed-area table is defined as follows:
a. Split scene into layers along the z-axis, see the left side of figure 1. Necessary depth sorting for occlusion culling is done implicitly via hierarchical view frustum culling.
b. Render the front most layer with the Catmull recursive subdivision algorithm.
c. Build a summed-area table $T(x,y)$ of the function $c(x,y)$, where $c(x,y) = 1$, if pixel (x,y) is covered by some surfaces, otherwise $c(x,y) = 0$. The summed-area table is the occlusion representation.
d. Render the next layer with a modified Catmull recursive subdivision algorithm:

Algorithm C´ (*Catmull's recursive subdivision algorithm facilitated with occlusion culling*). Given: camera including layerdepth, surfaces. Find a raster image of the scene

For each surface:
C´1. Compute camera-space axis-aligned bounding box.
C´2. Project bounding box into image space.
C´3. If bounding rect is entirely outside image borders:
drop surface.
C´4. Else: if area of bounding rect is larger than one pixel:
C´O. If bounding rect is totally occluded:
drop surface.
C´5. Else: subdivide surface.
C´6. Else: rasterize surface, z-buffer visibility test.

e. Return to step **c**, unless the image has been totally covered or the last layer is reached.
In step **C´O**, let us assume the bounding rect has two corners (x_0,y_0) and (x_1,y_1), see also the right side of figure 1. Denote the number of pixels covered by the bounding rect by B and compute $B = (x_1 - x_0)(y_1 - y_0)$. Denote the number of pixels covered by some surfaces in this bounding rect by A and compute

$A = T(x_1,y_1) - T(x_1,y_0) - T(x_0,y_1) + T(x_0,y_0)$. Also, denote an error tolerance $\varepsilon \geq 0$.
It can be seen that the surface is totally occluded and can be dropped, if $A \geq B - \varepsilon$.

Figure 2: Scene of trees.

Figure 2 shows a case study: the rendering of a scene containing approximately one hundred thousand trees with geometrically different features. The scene has been described procedurally, we make use of data amplification [EWM+98]. Our software implementation of the outlined algorithm is three times faster than the native Catmull subdivision algorithm for the rendition of this image. But runtime is highly dependent on the geometric model and the view-point.

Figure 3: Depth-of-field and tollerance culling.

Properties of using a summed-area table as the occlusion representation are as follows: (1) An easy to understand and straight forward to implement method. (Think about the z-buffer compared to more elaborate hidden surface algorithms.) (2) A relatively long update time compared to the z-pyramid used for the hierarchical z-buffer. However, computing the summed-area table can be accomplished in parallel to rendering the next layer. (3) A fast occlusion test and efficient culling of small and thin surfaces. (4) A simple way to achieve occlusion culling with tolerances, for that surfaces do not have to be totally occluded to be dropped, see figure 3, left for an excessive appliance for illustrative purposes. Furthermore, using layers gives the potential to generate computationally cheap depth-of-field effects, as shown in figure 3, right. Different layers can be rendered even in different resolutions resulting in reduced rendering time.

5. SUBDIVISION IN SPACETIME

Traditionally, rendering is handled by a mapping from a 3D scene description to a 2D raster image: $R^3 \rightarrow N^2$. However, rendering an animation is a 4-dimensional problem. We generalize this mapping process from a 4D scene description including motion descriptions to a 3D raster image block—a sequence of 2D images: $R^4 \rightarrow N^3$.

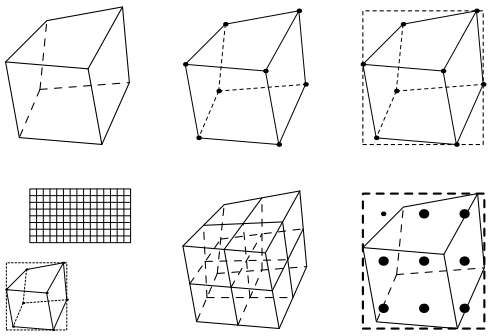

Figure 4: Illustrations for spacetime subdivision denoted in left-to-right order: trilinear body; steps **S1**, **S2**, **S3**, **S4**, and **S6** of algorithm S.

In computer graphics bodies are usually reduced to their surfaces, the things that are directly visible. In a minimalistic case study, we address linear motions of the control points of a bilinear patch: 3D control points are linearly interpolated in time. We define a trilinear body in spacetime, given by 8 control points: $P_0, P_1, \ldots, P_7; P_i \in R^4$, see figure 4.

Algorithm S (*Recursive subdivision algorithm operating in spacetime*). Given: camera, bodies. Find a raster image block.

For each body:
S1. Project control points to image spacetime.
S2. Compute bounding block in image spacetime.
S3. If bounding block is entirely outside view spacetime:
drop body.
S4. Else: if area of bounding rect in image space larger
than area of one pixel:
S5. Subdivide body.
S6. Else: rasterize body, z-buffer visibility test.

We add an occlusion culling stage analogous to that of the last chapter: Corners of the bounding block in raster image spacetime become then (x_0,y_0,t_0) and (x_1,y_1,t_1). The Number of pixels covered by this block is $\#B_S = (x_1-x_0)(y_1-y_0)(t_1-t_0)$, and the number of pixels covered by some body in this block computed with a summed-volume table T_S becomes:

$$\begin{aligned} A_S \quad = \quad & T_S(x_1,y_1,t_1) - T_S(x_0,y_1,t_1) - T_S(x_1,y_0,t_1) - \\ & T_S(x_1,y_1,t_0) + T_S(x_1,y_0,t_0) + T_S(x_0,y_1,t_0) + \\ & T_S(x_0,y_0,t_1) - T_S(x_0,y_0,t_0) \end{aligned}$$

An body is totally occluded for all frames in $[t_0, t_1]$ and can be dropped, if $A_S \geq B_S - \varepsilon$.

Figure 5: Trilinear bodies moving in spacetime.

In figure 5, the light patch is not moving and so the number of its subdivisions is independent of the number of frames. On the right side, it is culled by only one occlusion test for the last two frames (micropolygons are randomly graded to show their structure).

6. DISCUSSION AND FUTURE WORK

We made some observations that may influence future work for spacetime recursive subdivision:

Bounding boxes are too large in higher dimensions. Our implementation shows an explosion in the number of subdivisions for complex bodies. One way to address these problems is to perform anisotropic subdivisions: to subdivide not in all parameter spaces at once. This is also beneficial for texture filtering.

The layer depth should be changed adaptively in dependance of the number of newly covered raster elements of the last rendered layer. To adapt our algorithm to interactive real-time rendering, a prediction function would be required measuring the probability of future events.

An A-buffer rasterization [Car84] with volume samples represented by bit masks would be beneficial.

Geometric primitives and motion descriptions that are well suited for subdivision in space time have to be found. Furthermore, a known scene description language has to be adapted or a new one has to be developed.

Our approach consumes a huge amount of memory for z-buffer, raster image, and summed-volume table. Subdivision of the image into buckets [CCC87] can help, although some spacial coherence at the border of buckets will be lost.

7. ACKNOWLEDGEMENT

Warm thanks go to Daehyun Kim, Frieder Nake, Caroline von Totth, Eric Allen Engle, anonymous WSCG reviewers, and Eva.

8. REFERENCES

[AGB99] Apodaca A. A., Gritz L., and Barsky B. A. *Advanced RenderMan: Creating CGI for Motion Picture.* Morgan Kaufmann Publishers Inc., 1999.

[AMN03] Aila T., Miettinen V., and Nordlund P. Delay streams for graphics hardware. *ACM Trans. Graph.*, 22(3):792–800, 2003.

[Car84] Carpenter L. The a-buffer, an antialiased hidden surface method. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 103–108. ACM Press, 1984.

[Cat74] Catmull E. E. *A subdivision algorithm for computer display of curved surfaces.* PhD thesis, University of Utah, 1974.

[CCC87] Cook R. L., Carpenter L., and Catmull E. The reyes image rendering architecture. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 95–102. ACM Press, 1987.

[COCSD03] Cohen-Or D., Chrysanthou Y. L., Silva C. T., and Durand F. A survey of visibility for walkthrough applications. In *Transactions on visualization and computer gaphics, vol. 9, no. 3*, pages 412–431. IEEE Computer Society, 2003.

[Cro84] Crow F. C. Summed-area tables for texture mapping. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 207–212. ACM Press, 1984.

[DS99] Damez C. and Sillion F. Space-time hierarchical radiosity. In *Rendering Techniques '99*, pages 235–246, New York, NY, 1999. Springer Wien.

[EWM+98] Ebert D. S., Worley S., Musgrave F. K., Peachey D., Perlin K., and Musgrave K. F. *Texturing and Modeling*. Academic Press, Inc., 1998.

[Gen01] Genz A. Occlusion culling and summed-area tables. In *SIGGRAPH'2001: Conference Abstracts and Applications*, page 238. ACM SIGGRAPH, August 2001.

[GKM93] Greene N., Kass M., and Miller G. Hierarchical z-buffer visibility. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 231–238. ACM Press, 1993.

[Gla88] Glassner A. S. Spacetime ray tracing for animation. *IEEE Computer Graphics and Applications*, 8:60–70, March 1988.

[HCSM00] Huang J., Crawfis R., Shareef N., and Mueller K. Fastsplats: optimized splatting on rectilinear grids. In *Proceedings of the conference on Visualization '00*, pages 219–226. IEEE Computer Society Press, 2000.

[HDMS03] Havran V., Damez C., Myszkowski K., and Seidel H.-P. An efficient spatio-temporal architecture for animation rendering. In *Proceedings of the Eurographics Symposium on Rendering, vol. 14*, pages 106–117, June 2003.

[HW99] Ho P. C. and Wang W. Occlusion culling using minimum occluder set and opacity map. In *IEEE International Conference on Information Visualization*, pages 292–300, 1999.

[OKTD02] Owens J. D., Khailany B., Towles B., and Dally W. J. Comparing reyes and opengl on a stream architecture. In *Proceedings of the conference on Graphics hardware 2002*, pages 47–56. Eurographics Association, 2002.