Interactive Subdivision of Smooth Surfaces on GPUs

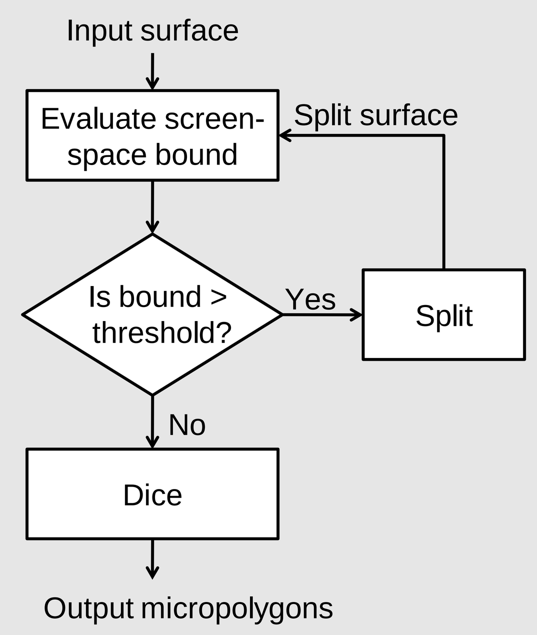
Anjul Patney, Mohamed S. Ebeida*, John D. Owens
University of California, Davis, *Carnegie Mellon University

Parametric surfaces

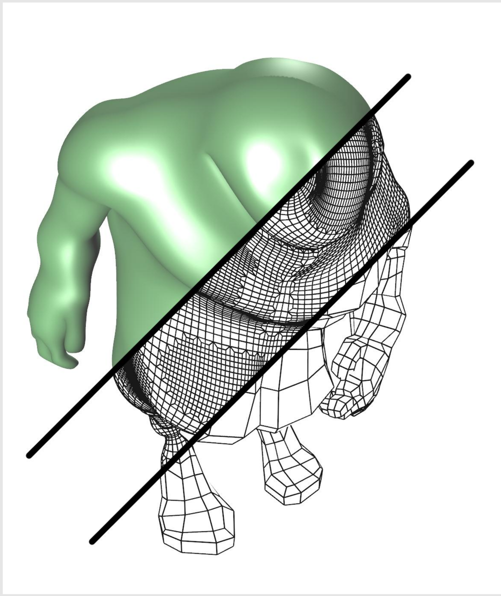
Abstract

We present a GPU based implementation of Reyes-style adaptive surface subdivision, known in Reyes terminology as the Bound/Split and Dice stages. The performance of this task is important for the Reyes pipeline to map efficiently to graphics hardware, but its recursive nature and irregular and unbounded memory requirements present a challenge to an efficient implementation. Our solution begins by characterizing Reyes subdivision as a work queue with irregular computation, targeted to a massively parallel GPU. We propose efficient solutions to these general problems by casting our solution in terms of the fundamental primitives of prefix-sum and reduction, often encountered in parallel and GPGPU environments.

Our results indicate that real-time Reyes subdivision can indeed be obtained on today's GPUs. We are able to subdivide a complex model to subpixel accuracy within 15 ms. Our measured performance is several times better than that of Pixar's RenderMan. Our implementation scales well with the input size and depth of subdivision. We also address concerns of memory size and bandwidth, and analyze the feasibility of conventional ideas on screen-space buckets.



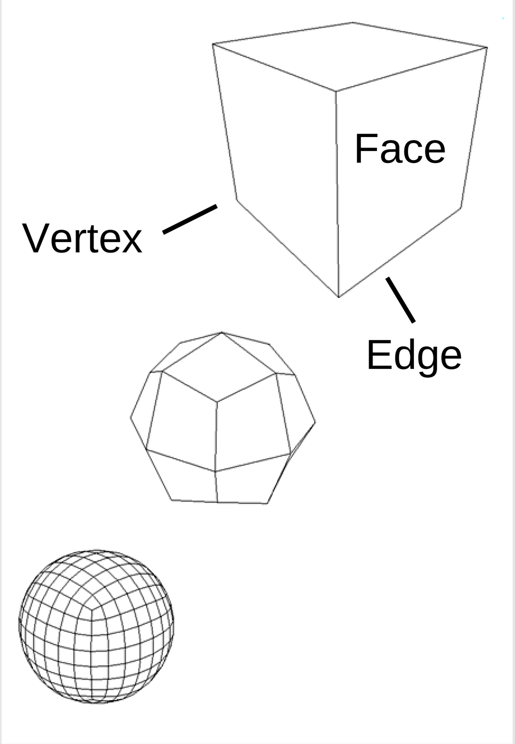
Subdivision Surfaces



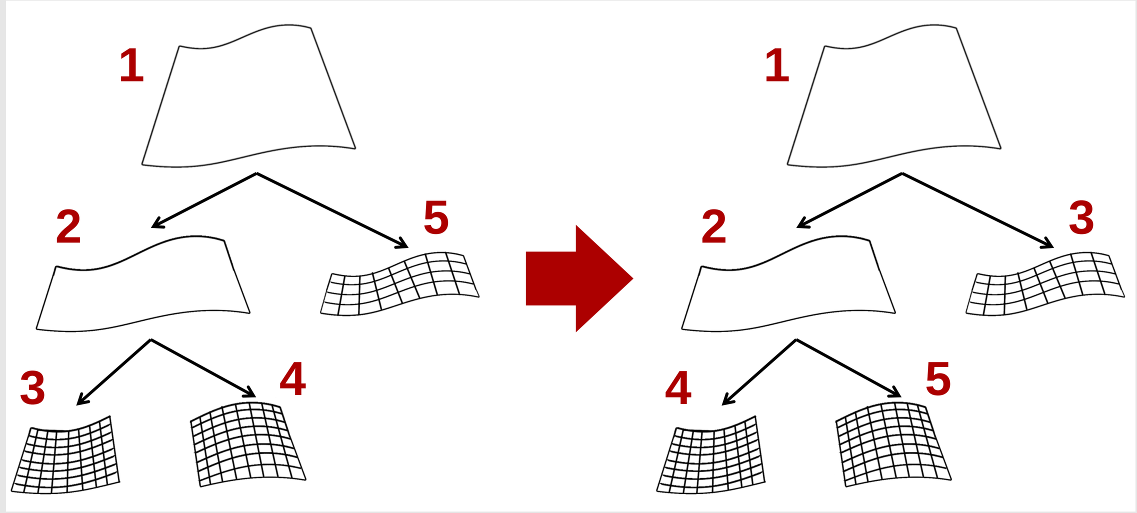
Abstract

We present a strategy for performing view-adaptive, crack-free tessellation of Catmull-Clark subdivision surfaces entirely on programmable graphics hardware. Our scheme extends the concept of breadth-first subdivision, which up to this point has only been applied to parametric patches. While mesh representations designed for a CPU often involve pointer-based structures and irregular per-element storage, neither of these is well-suited to GPU execution. To solve this problem, we use a simple yet effective data structure for representing a subdivision mesh, and design a careful algorithm to update the mesh in a completely parallel manner. We demonstrate that in spite of the complexities of the subdivision procedure, real-time tessellation to pixel-sized primitives can be done.

Our implementation does not rely on any approximation of the limit surface, and avoids both subdivision cracks and T-junctions in the subdivided mesh. Using the approach in this paper, we are able to perform real-time subdivision for several static as well as animated models. Rendering performance is scalable for increasingly complex models.

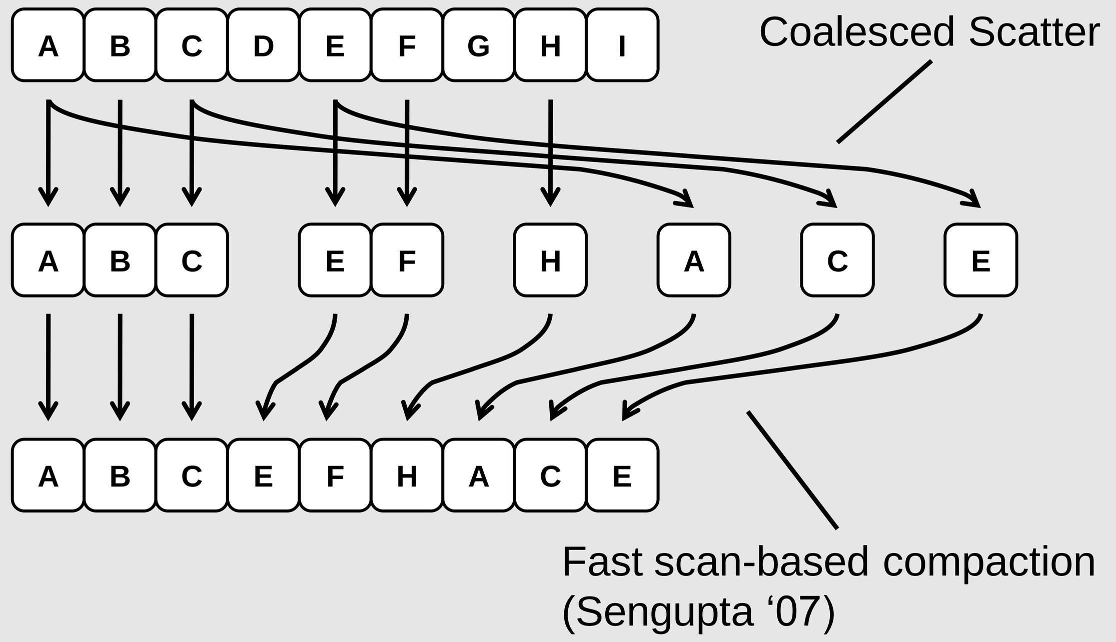


Algorithm



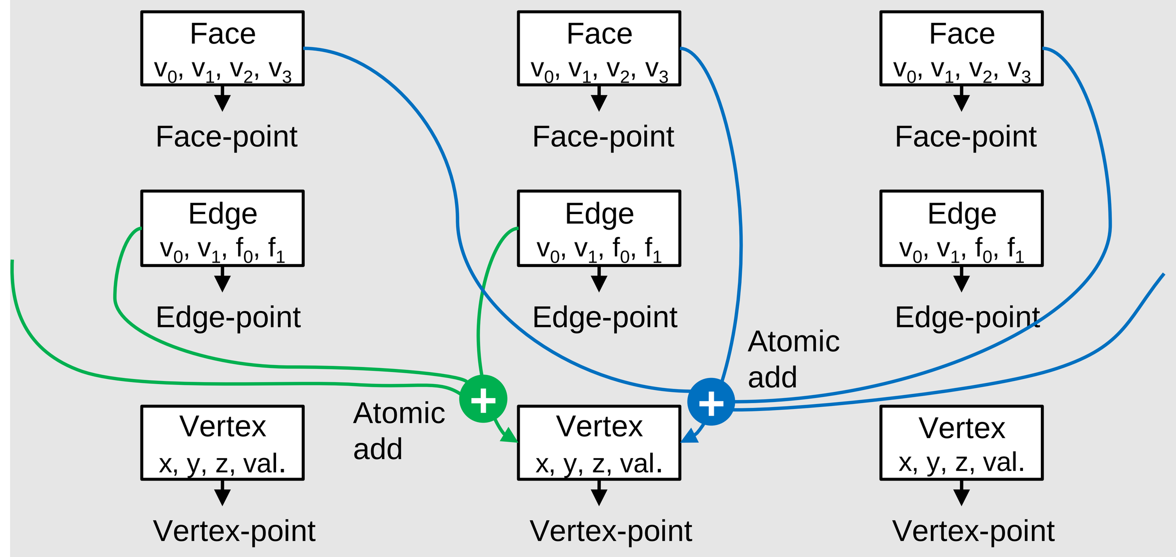
Breadth-first Subdivision

Dynamic Work-Queue

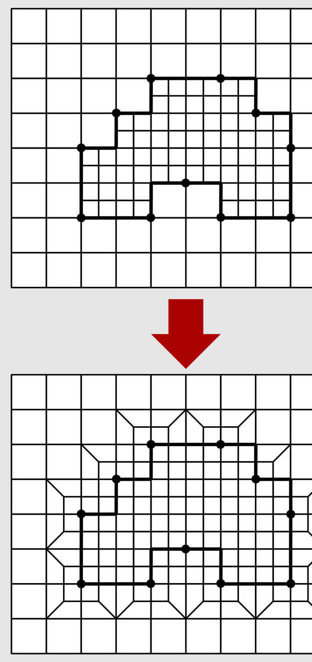


Approach

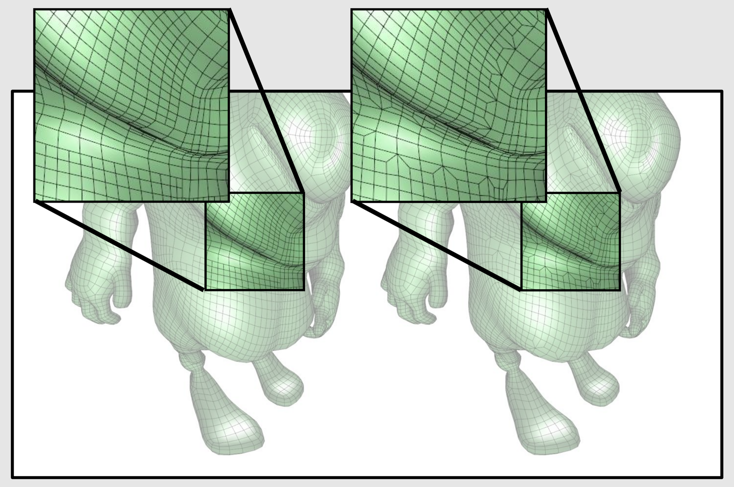
Maintain three arrays: **Vertices**, **Faces**, and **Edges**



Fixing Cracks

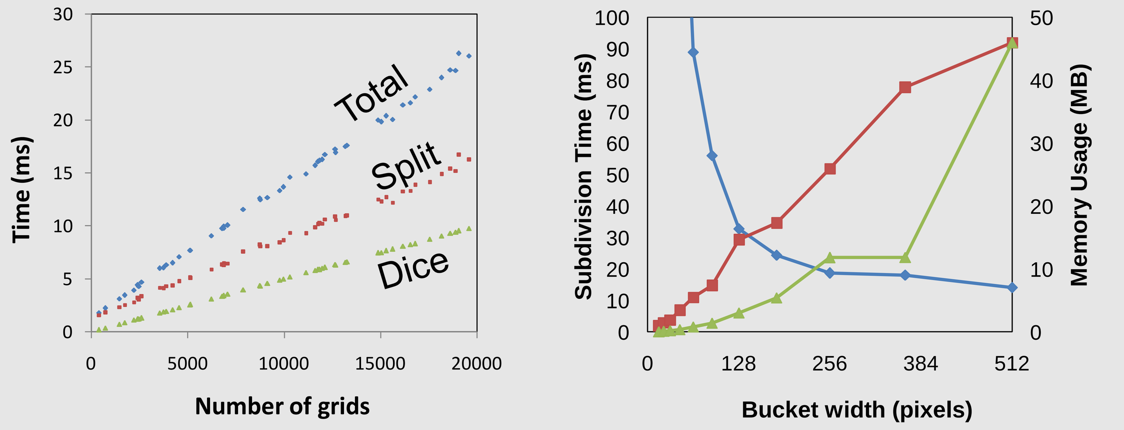


Cracks due to view-adaptive subdivision are fixed entirely in parallel



Results

Model	Input patches	Output grids	Split time	Dice time	Rendering performance
Teapot	32	4,823	2.69 ms	1.27 ms	60.1 fps
Killeroo	11,532	14,426	6.30 ms	3.46 ms	29.7 fps



Summary

Conclusion

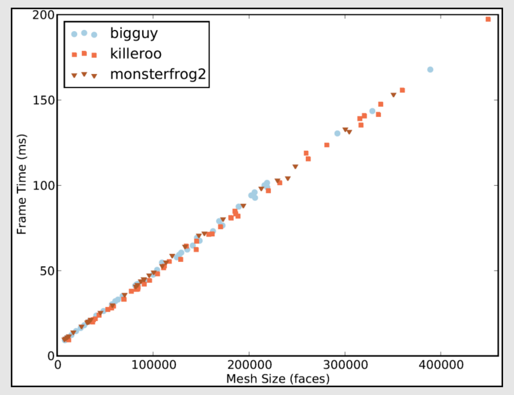
- Recursive Subdivision in real-time
  - Breadth-first formulation
  - Maps well to GPUs
- First step towards a real-time Reyes pipeline

Future Work

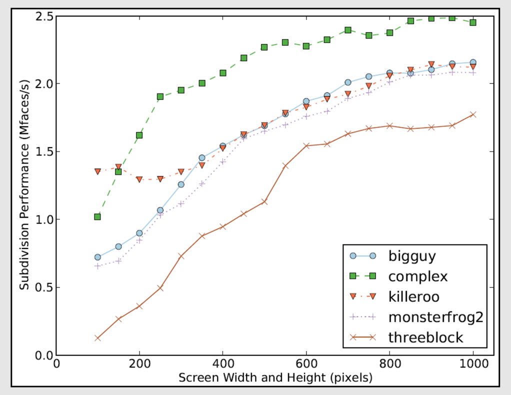
- Resolving subdivision cracks
- Displacement mapping

Results

Model	Input faces	Output faces	Rendering time
Big Guy	1,450	91,992	37.12 ms
Monster Frog	1,292	80,452	34.17 ms
Killeroo	2,894	80,227	31.34 ms



Rendering time grows linearly with scene complexity



Performance grows with complexity, and settles at 2-2.5 Mfaces/sec

Summary

Conclusion

- Parallel GPU tessellation of Catmull-Clark surfaces
- Robust data management for subdivision
- Dynamic view-dependence
- Fixing cracks in parallel

Future Work

- Efficient memory management
- Programmable geometry caching