

Parallel View-Dependent Tessellation of Catmull-Clark Subdivision Surfaces

Anjul Patney¹ Mohamed S. Ebeida² John D. Owens¹

University of California, Davis¹
Carnegie Mellon University²

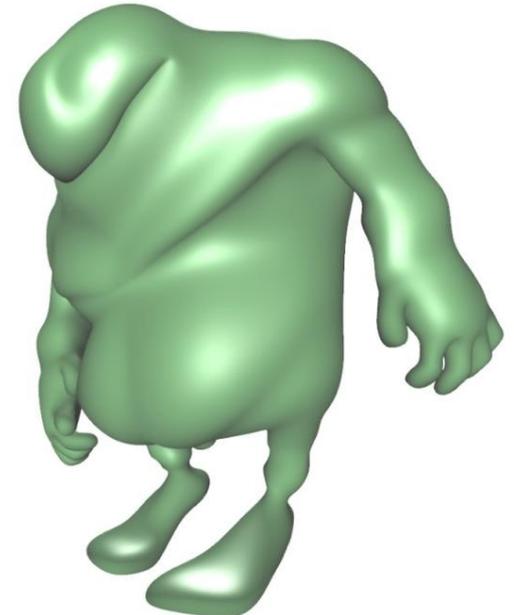
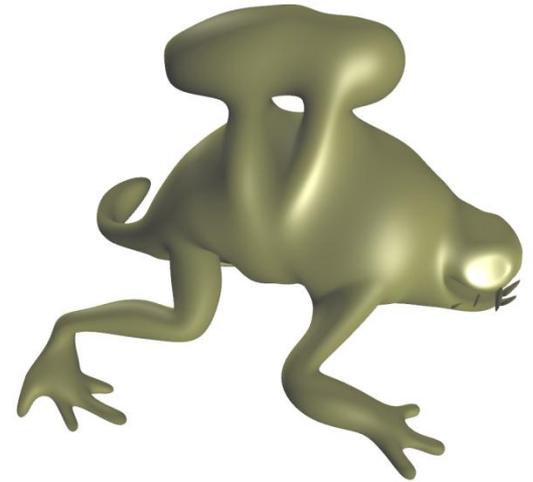
Smooth Surfaces in Interactive Graphics

Motivation

Resolution independent higher-order surfaces in real-time graphics

Challenges

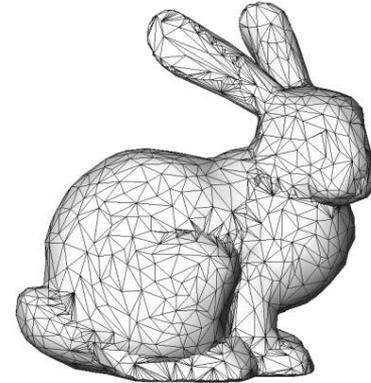
- Fast GPU-friendly tessellation
- Fine-grained view-adaptivity
- Cracks and Pinholes



Representing Smooth Surfaces

Polygonal Meshes

- Lack View-Adaptivity
- Inefficient storage/transfer



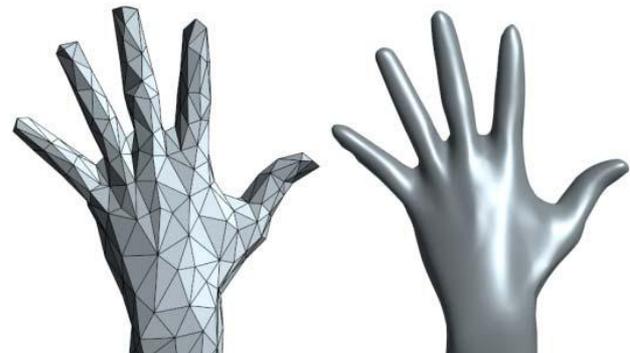
Parametric Surfaces

- Collections of smooth patches
- Hard to animate



Subdivision Surfaces

- Widely popular, ease of modeling
- Flexible Animation

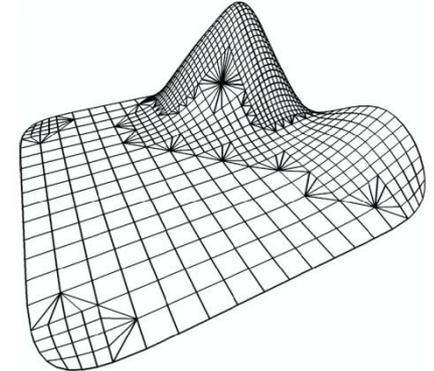


[Boubekeur and Schlick 2007]

Existing Work

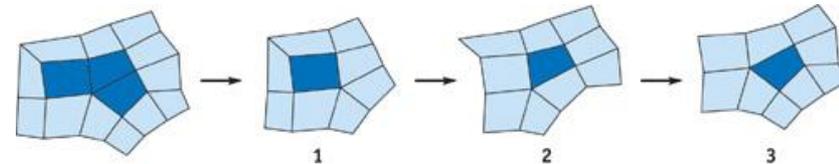
Subdivision Surfaces in real-time graphics

- CPU-based tessellation [Bolz and Schröder 2002]
 - Frequent CPU-GPU data transfer
 - Fixed maximum levels of subdivision



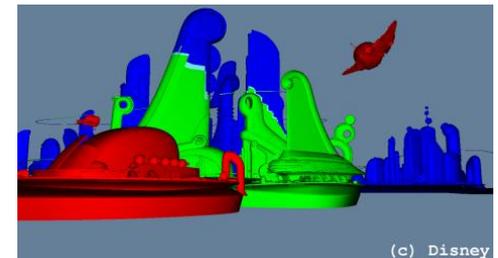
[Bolz and Schröder 2002]

- Subdivision using graphics shaders [Shiu et al. 2005], [Bunnell 2005]
 - Requires patching the input mesh
 - Limited view-adaptivity



[Bunnell 2005]

- Ray tracing of subdivision surfaces [Benthin et al. 2007]
 - Requires patching the input mesh
 - Coarse view-adaptivity



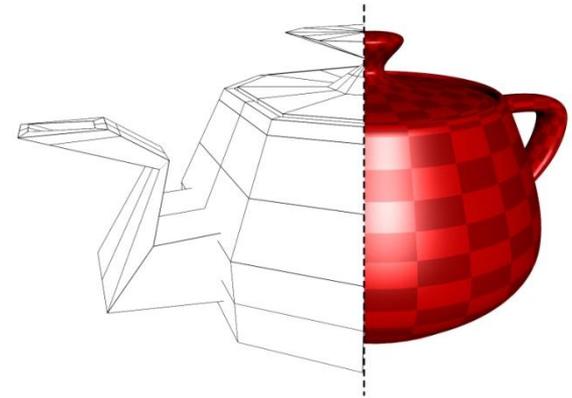
(c) Disney
[Benthin et al. 2007]

Existing Work

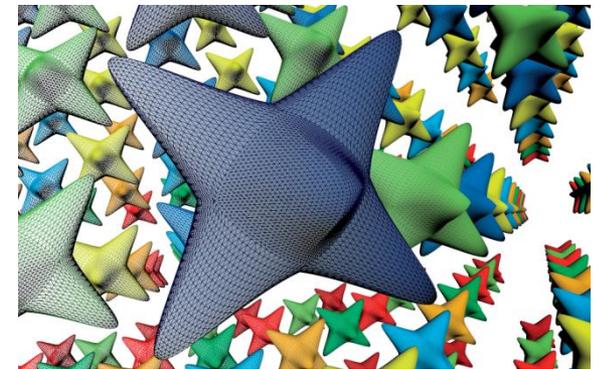
Programmable GPU Tessellation

- **Bézier Patches**
[Patney and Owens 2008], [Eisenacher et al. 2009]
 - Parametric surfaces
 - Cracks and T-junctions

- **CUDA Tessellation Framework**
[Schwarz and Stamminger 2009]
 - Parametric surfaces only



[Eisenacher et al. 2009]

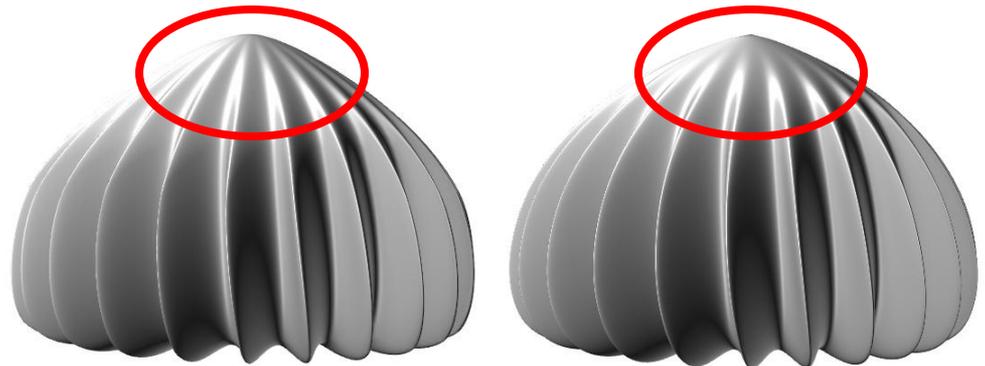


[Schwarz and Stamminger 2009]

Existing Work

Hardware Tessellation

- DirectX 11 Dedicated Tessellation Units
 - Primarily parametric surfaces
- Approximate Catmull-Clark Surfaces [Loop and Schaefer 2008]
 - Lost C^1 continuity, require separate normals
 - Inexact surfaces

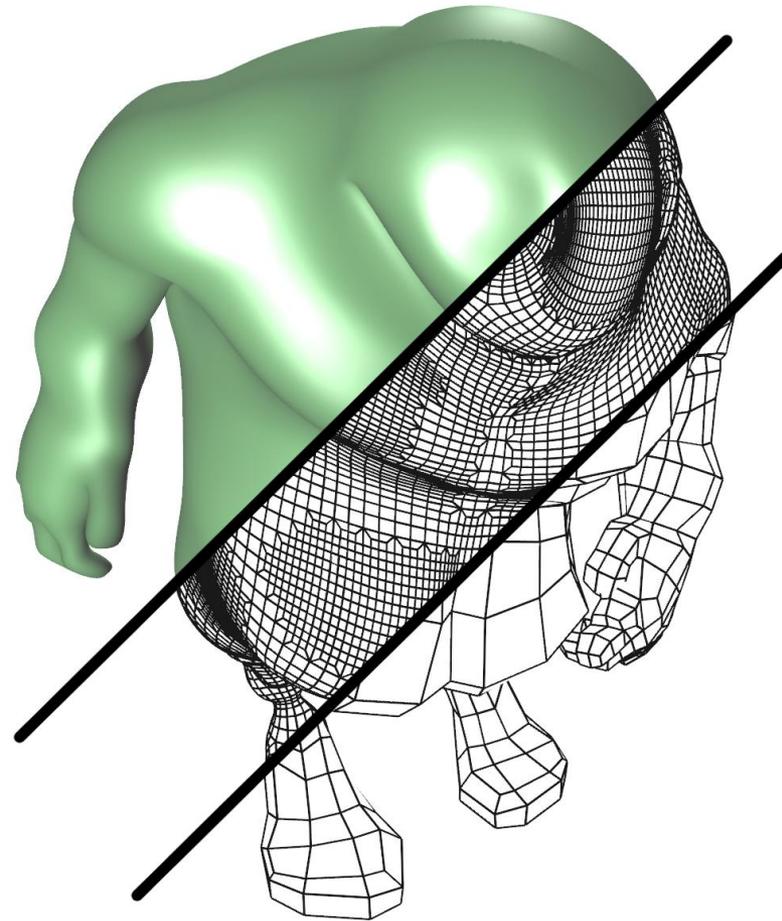


[Loop and Schaefer 2008]

This Paper: Contributions

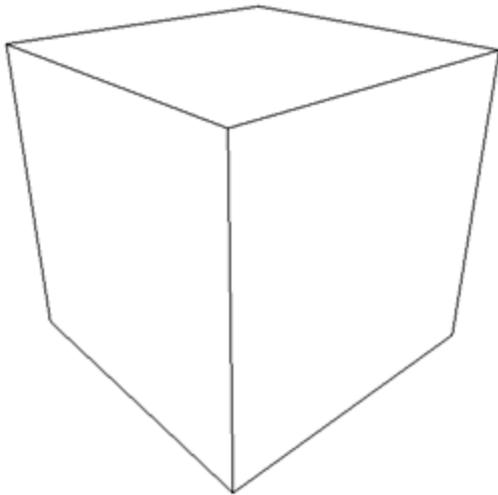
A parallel approach to Catmull-Clark subdivision

- Highly parallel, GPU-friendly
- Simple, robust data management
- Dynamic view-dependence
- No cracks or T-junctions

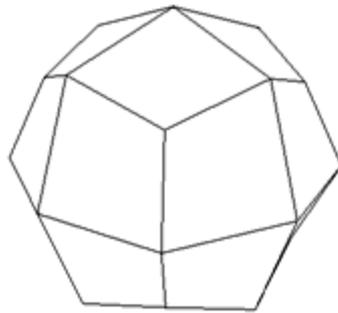


Review: Catmull-Clark Subdivision

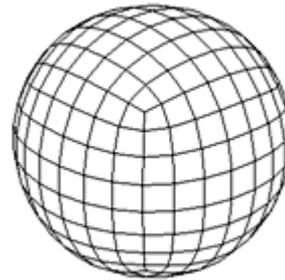
- Given: An arbitrary base mesh M_0
- Recursively refine to produce $M_1, M_2, M_3 \dots$
 - Fast convergence to a smooth surface



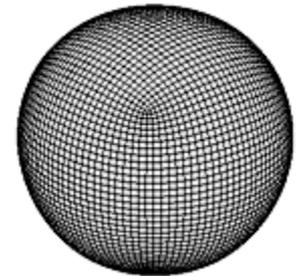
M_0



M_1

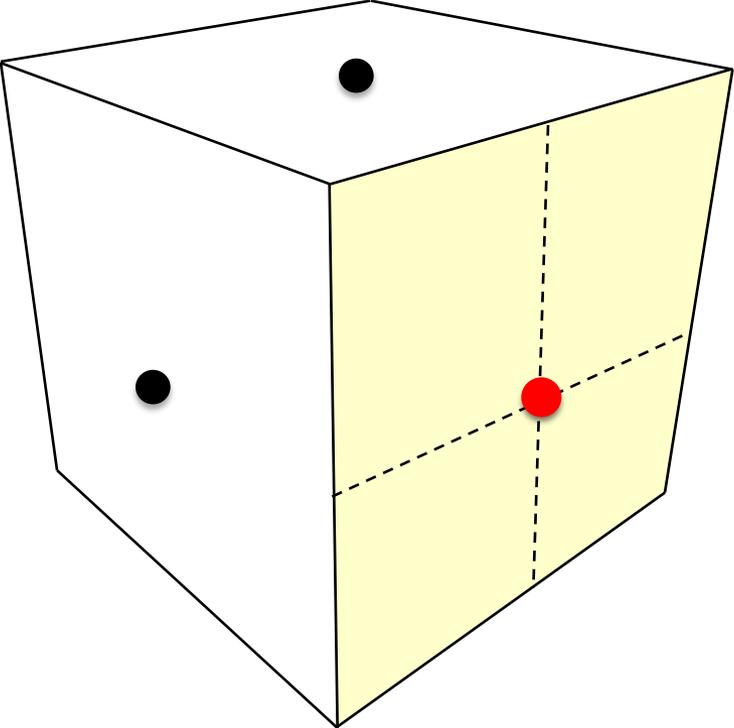


M_2



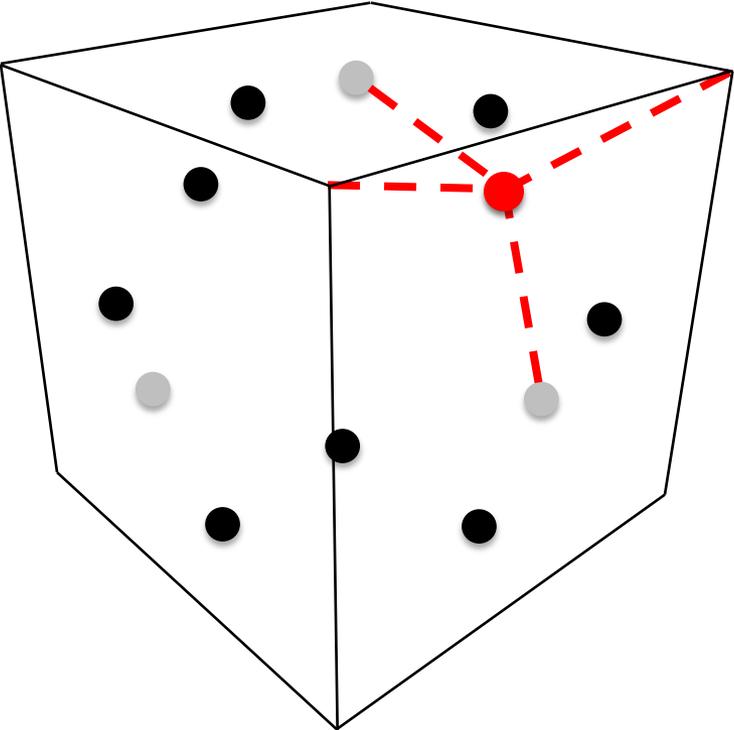
M_3

Refinement Procedure



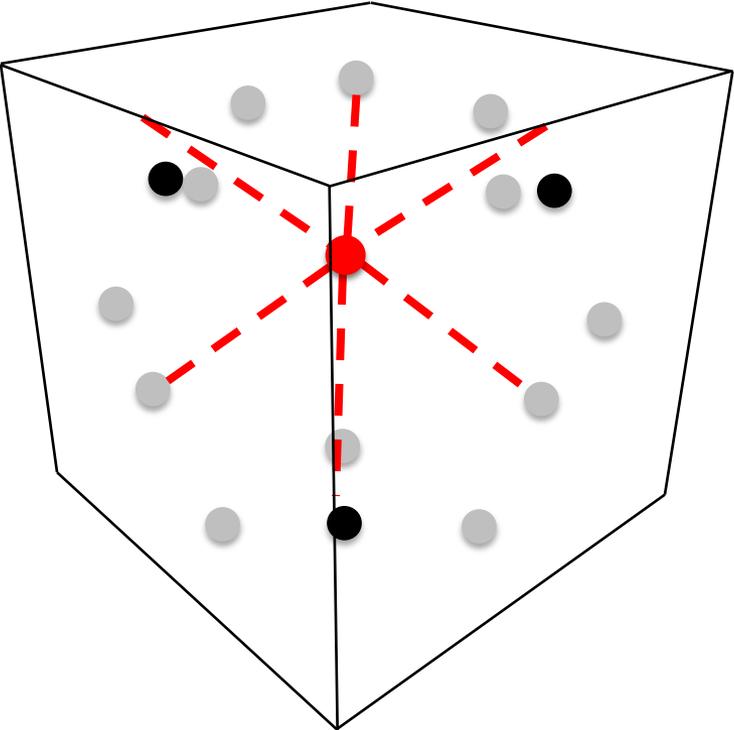
Face Points

Refinement Procedure



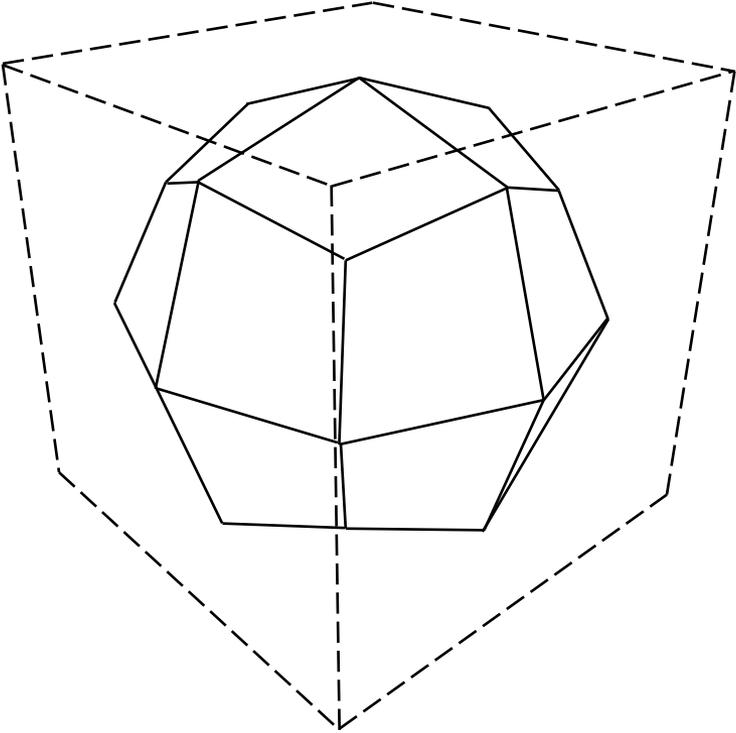
Edge Points

Refinement Procedure



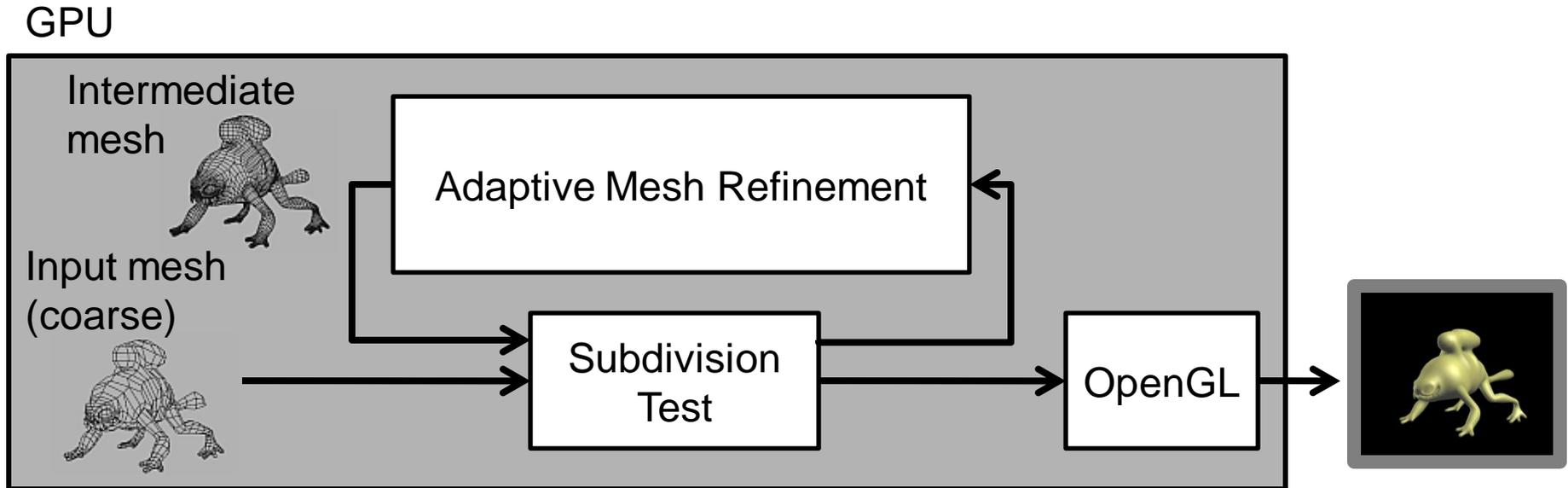
Vertex Points

Refinement Procedure

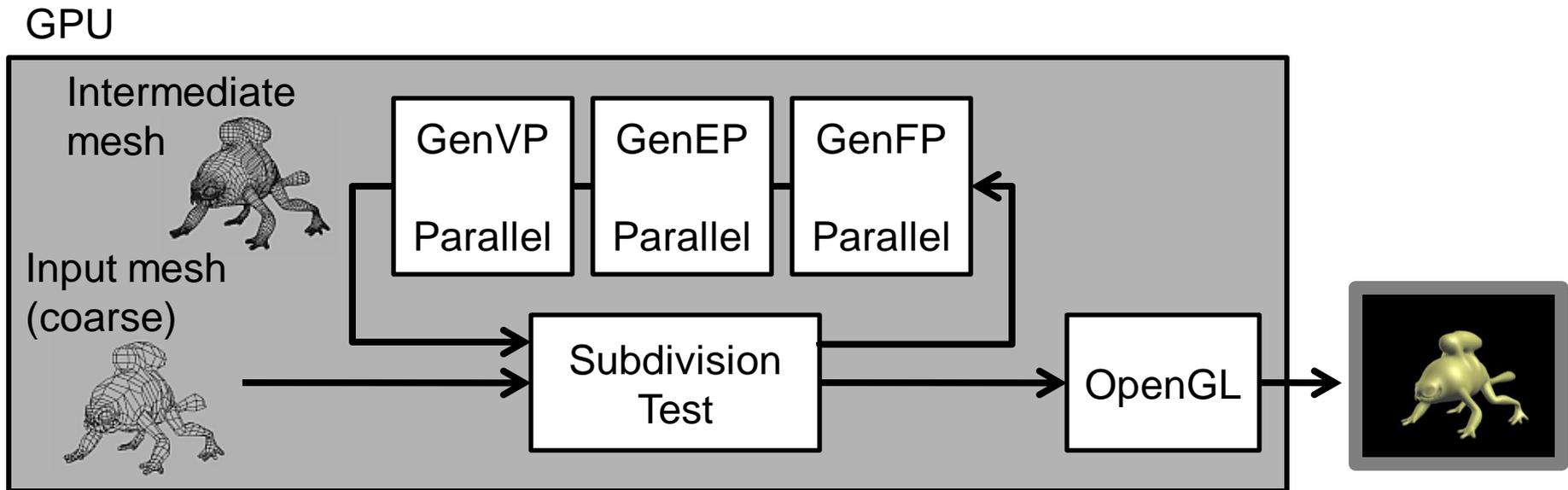


Subdivided Mesh

Basic Approach



Basic Approach



Challenges

- Data structure choice
 - Maintaining mesh structure
 - Efficiency on graphics hardware
- View-Dependence
 - Spatial (along the surface)
 - Temporal (dynamic viewpoint)
- Cracks and T-junctions
 - Avoid visual artifacts
 - Should not degrade performance

Data Structure

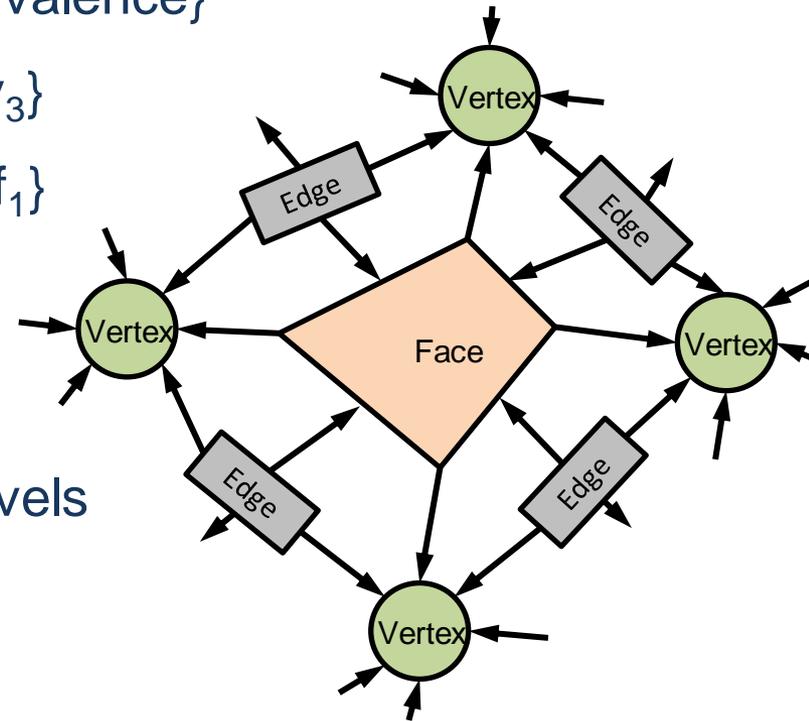
- Mesh Representation Goals
 - Easily exposed parallelism
 - Minimal communication
 - Low storage overhead
 - Straightforward rendering
- Problems with common Mesh Representations
 - Serialized pointer-accesses
 - Variable-size elements
 - Often cannot be rendered directly

Our Choice

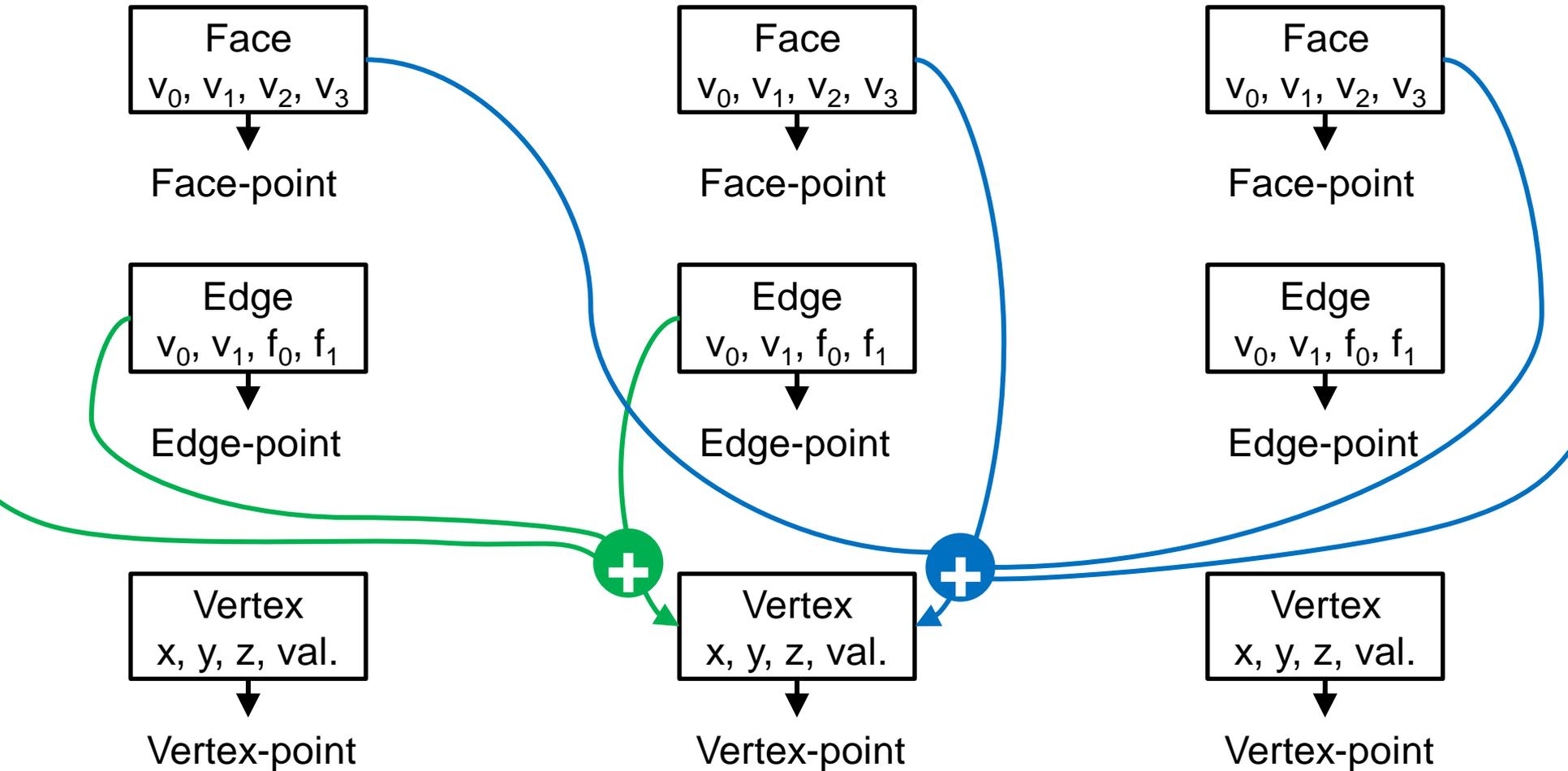
- Maintain three arrays
 - **VertexBuffer**, with vertex = {x, y, z, valence}
 - **FaceBuffer**, with face = {v₀, v₁, v₂, v₃}
 - **EdgeBuffer**, with edge = {v₀, v₁, f₀, f₁}

- Justification

- Parallelism expressed at all three levels
- Fixed-size elements
- Straightforward rendering
- Low storage overhead
- **Complexity of updates?**

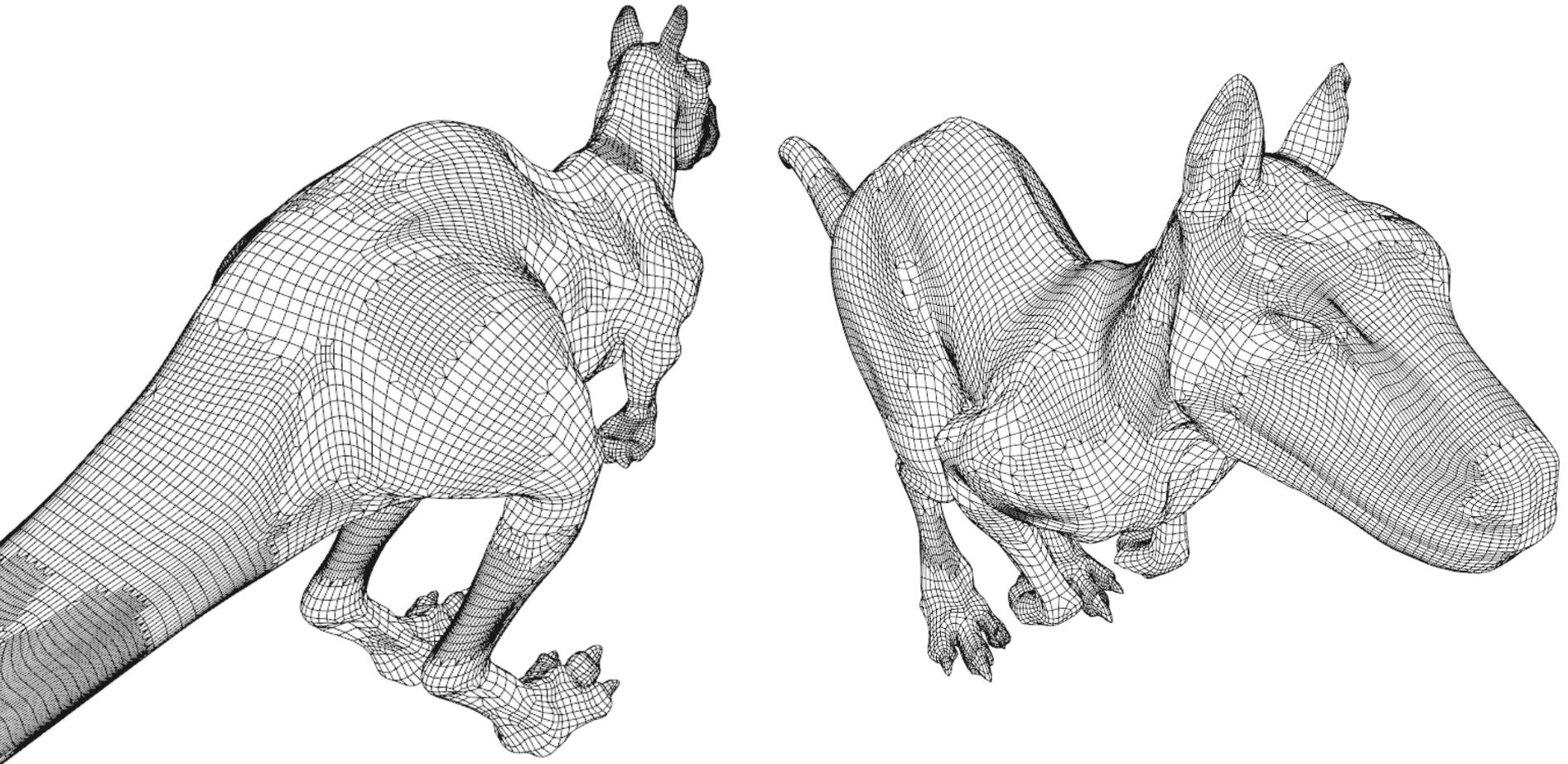


Data Structure Updates



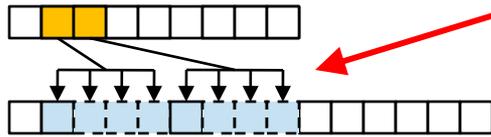
- Scattered atomic add (face-point)
- Scattered atomic add (edge midpoint)

View Dependence



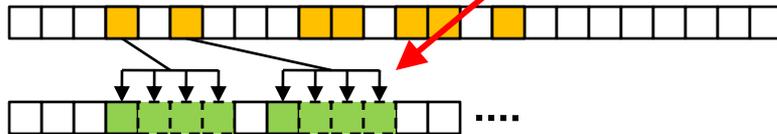
Subdivision Test

Faces

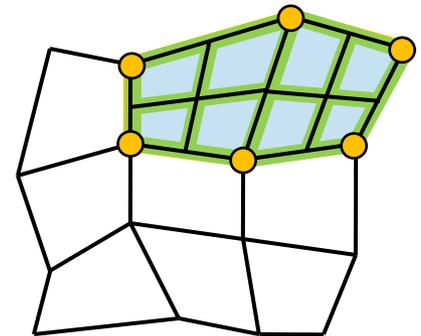


Parallel Prefix-Sum
[Sengupta et al. 2007]

Edges

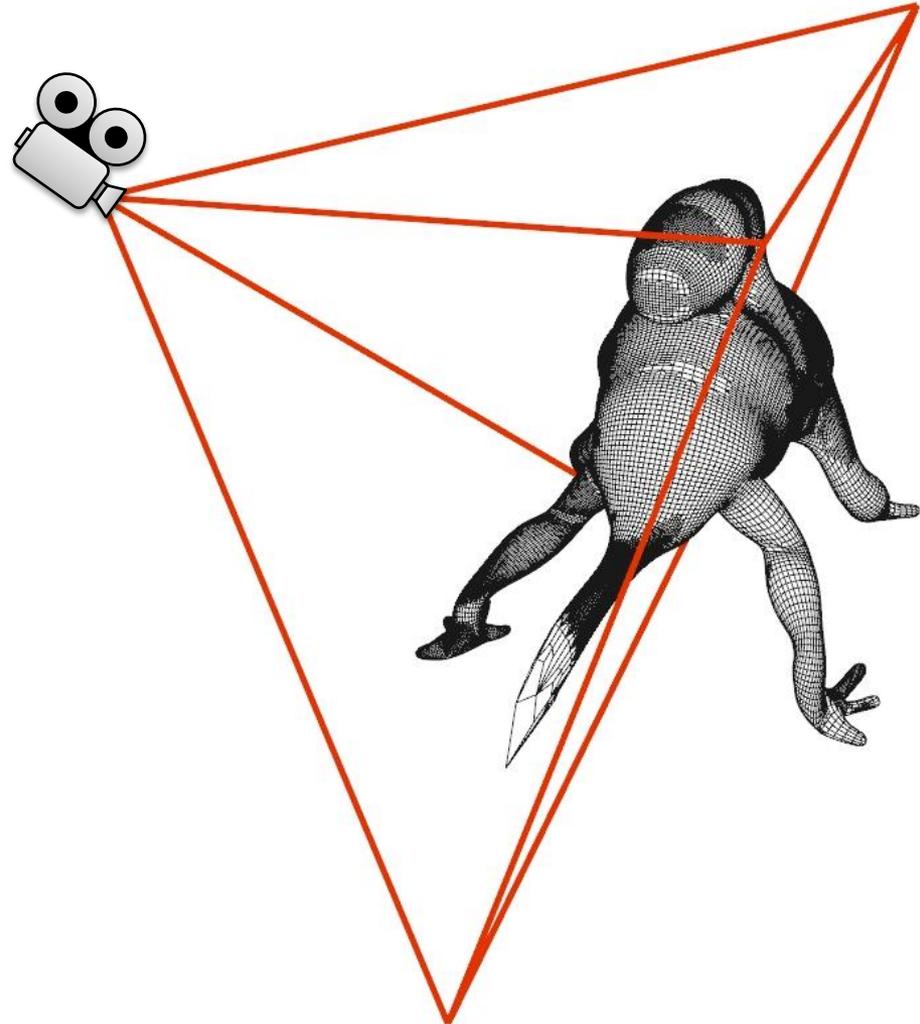
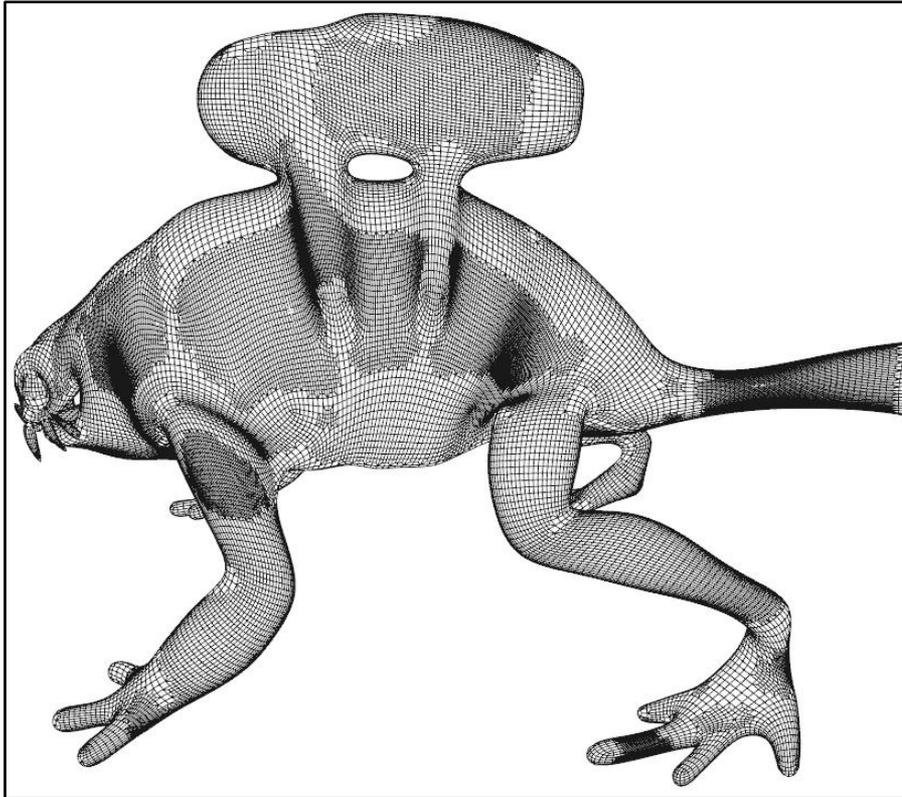


Vertices



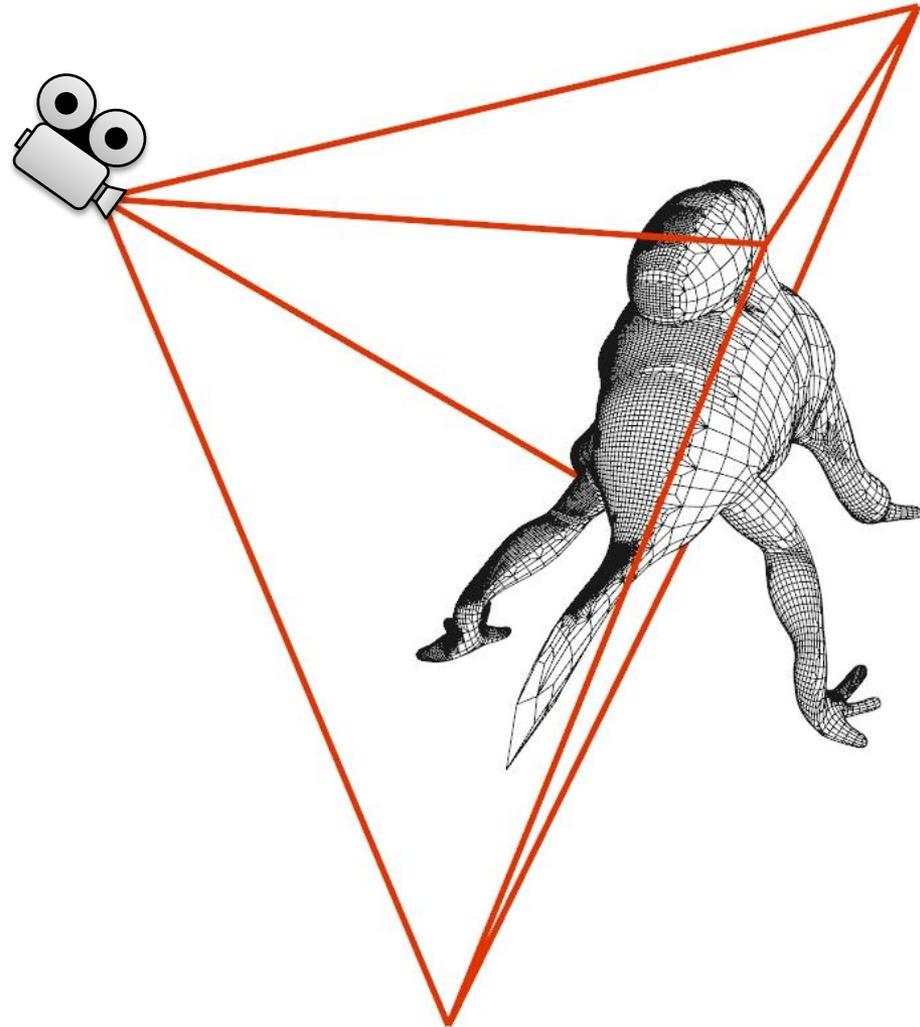
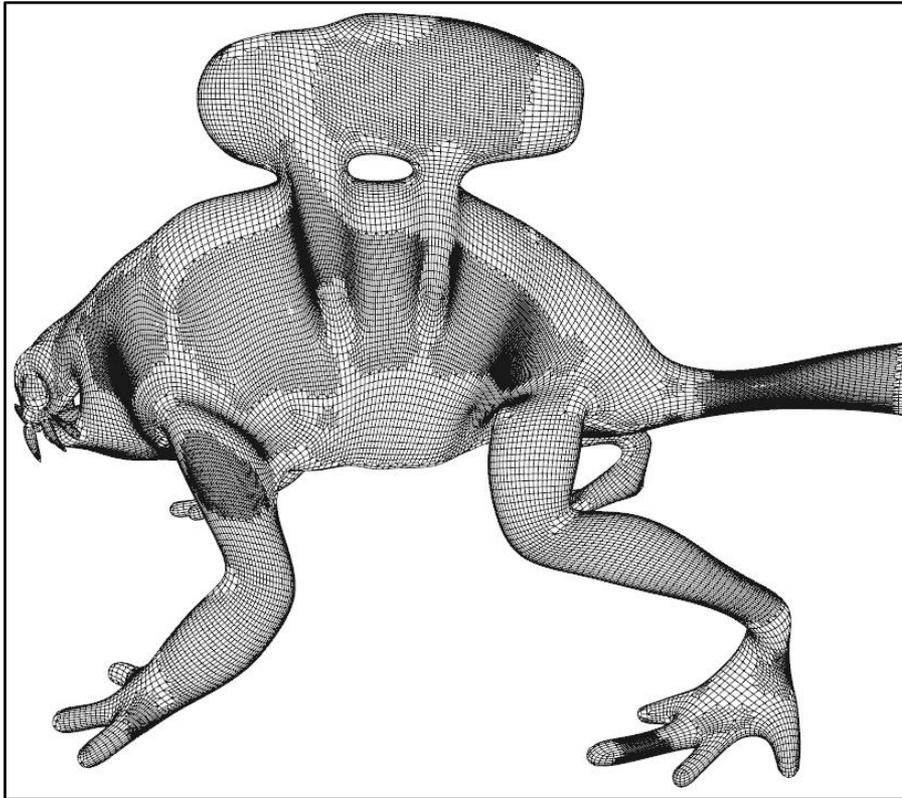
Varying Subdivision Criteria

Screen-space extent



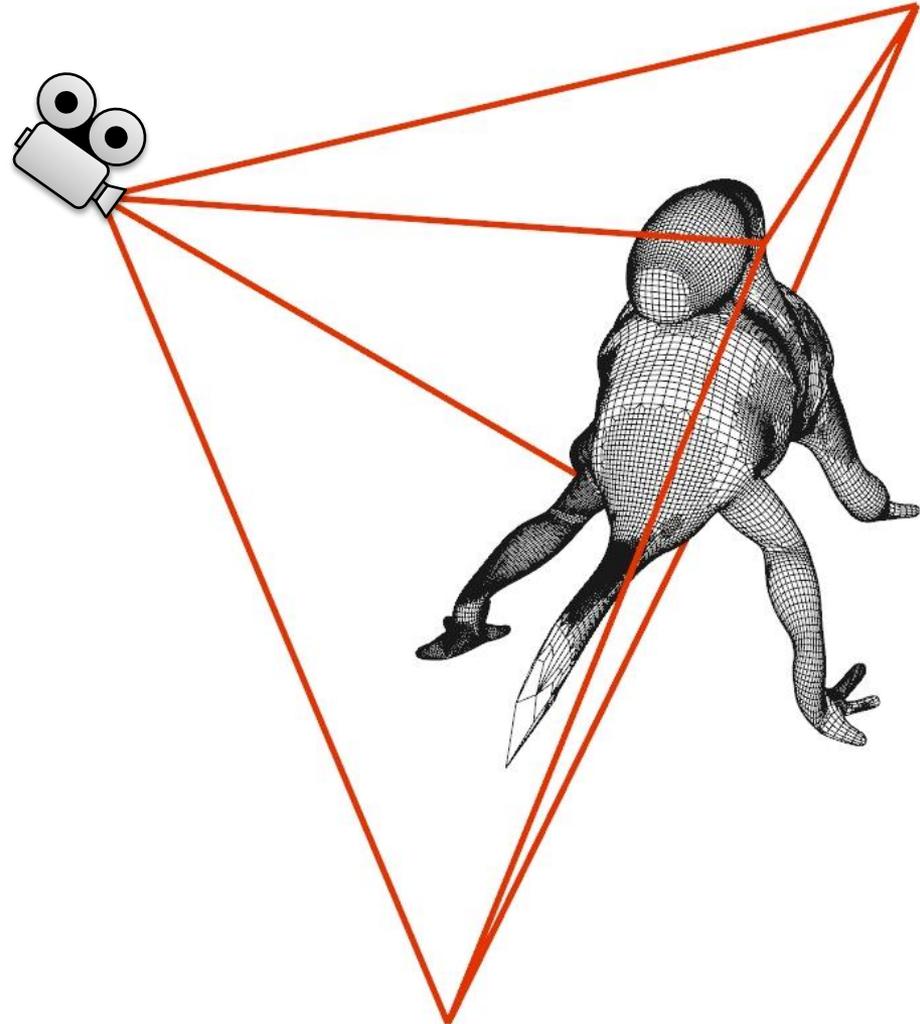
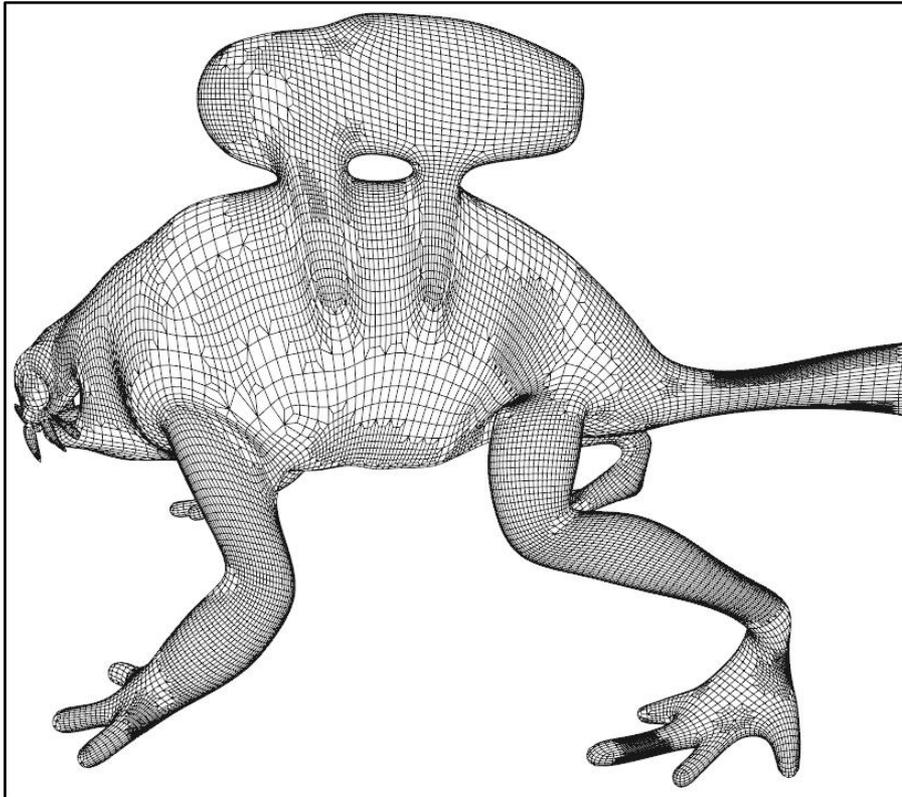
Varying Subdivision Criteria

Surface Orientation



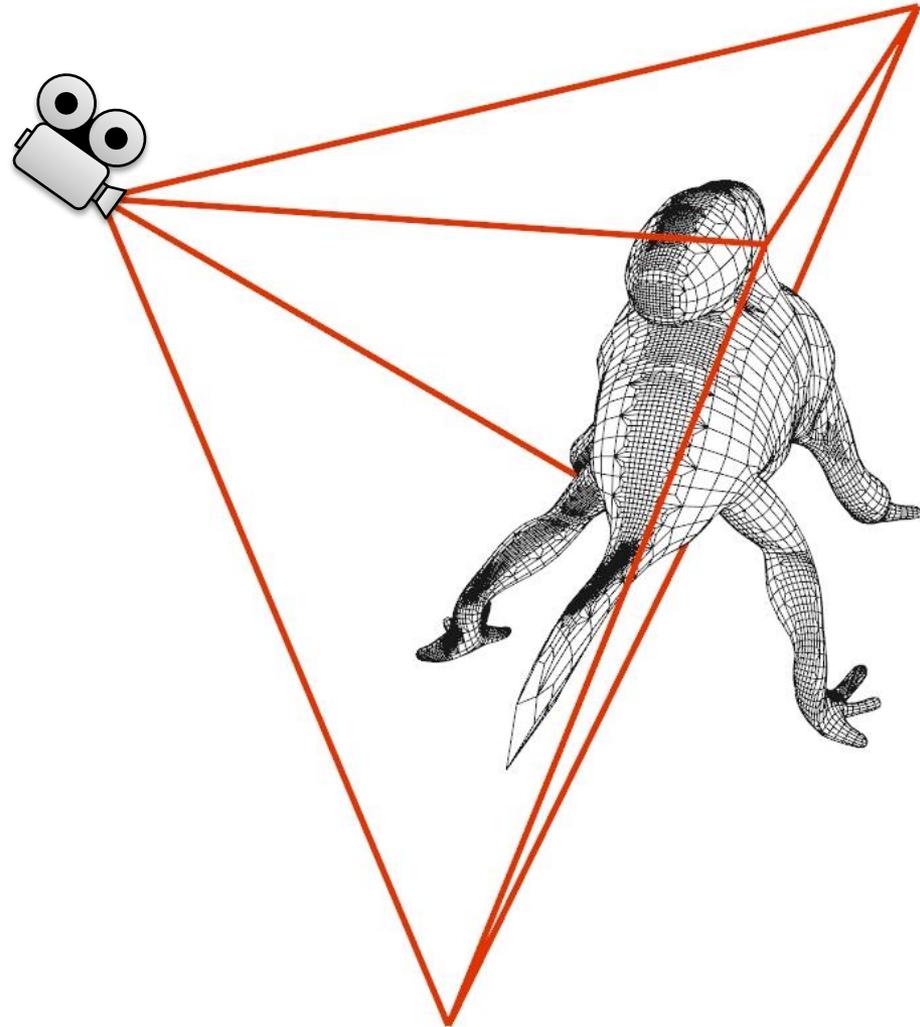
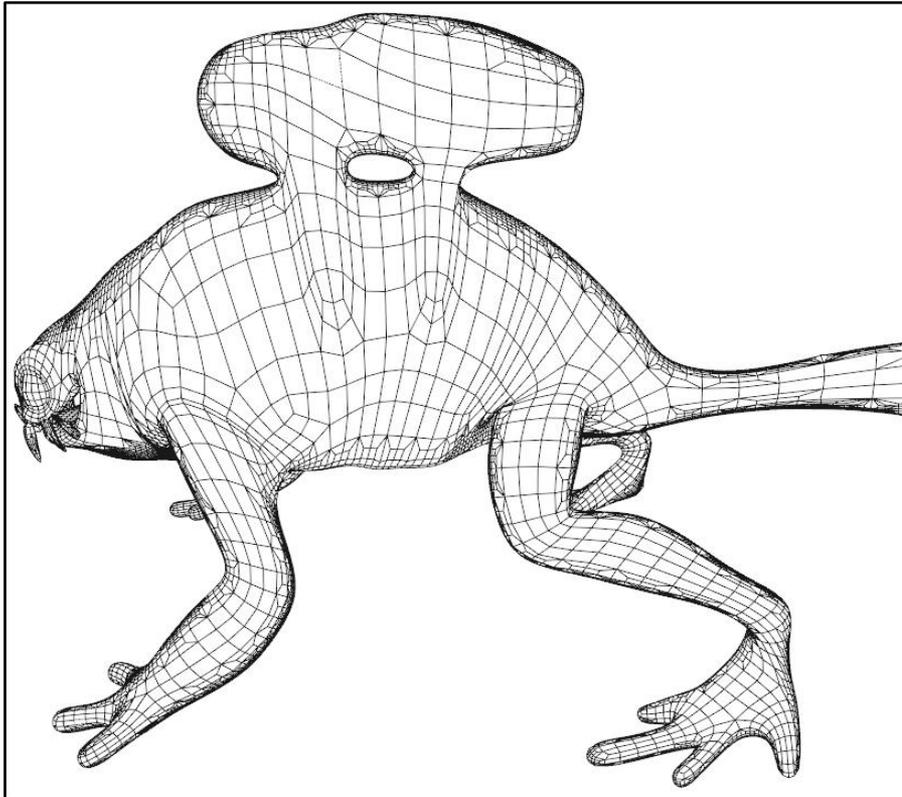
Varying Subdivision Criteria

Curvature



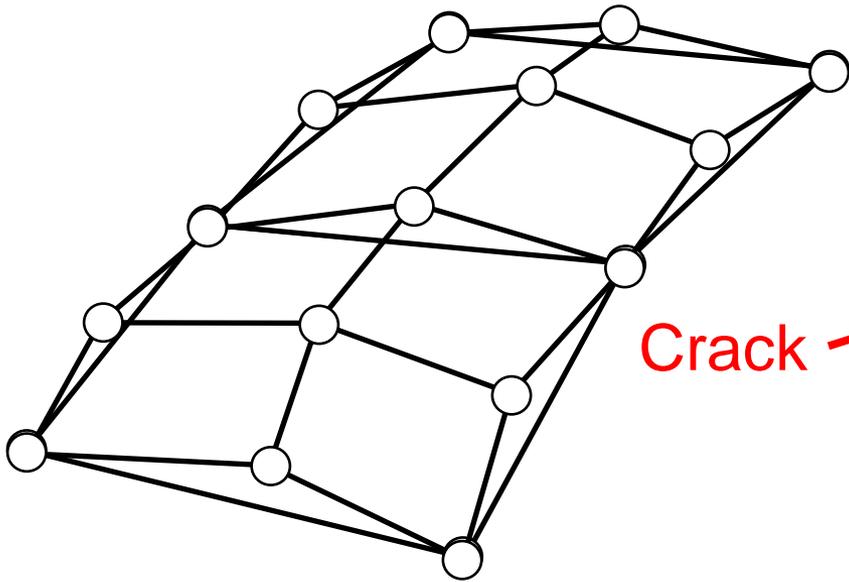
Varying Subdivision Criteria

Silhouette-enhancement

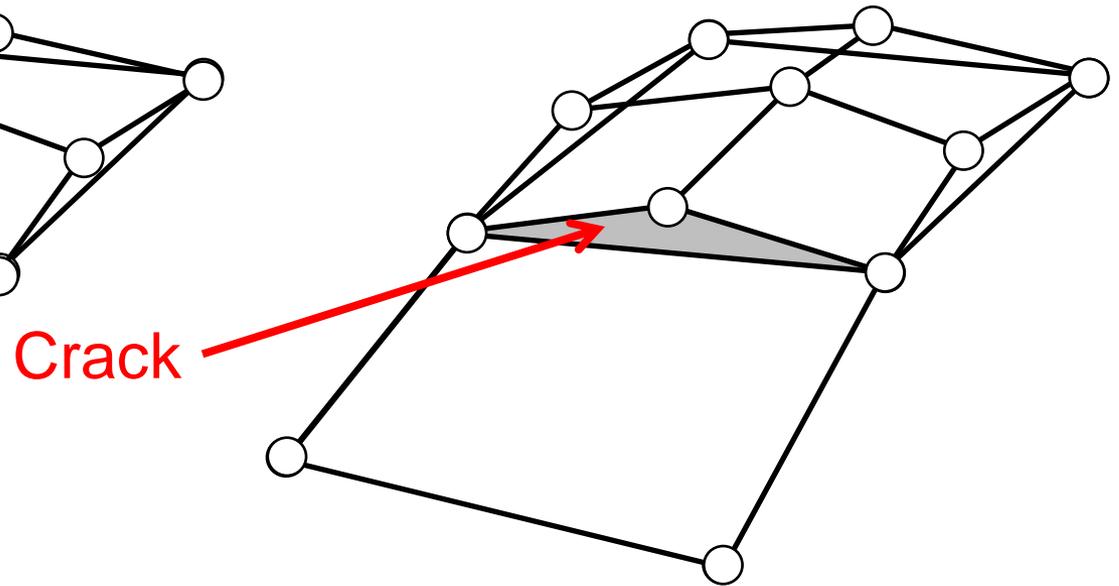


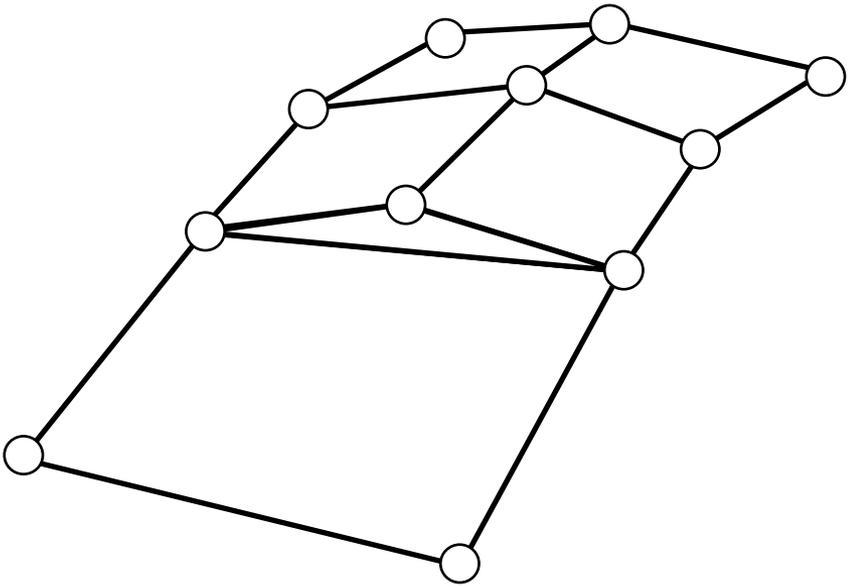
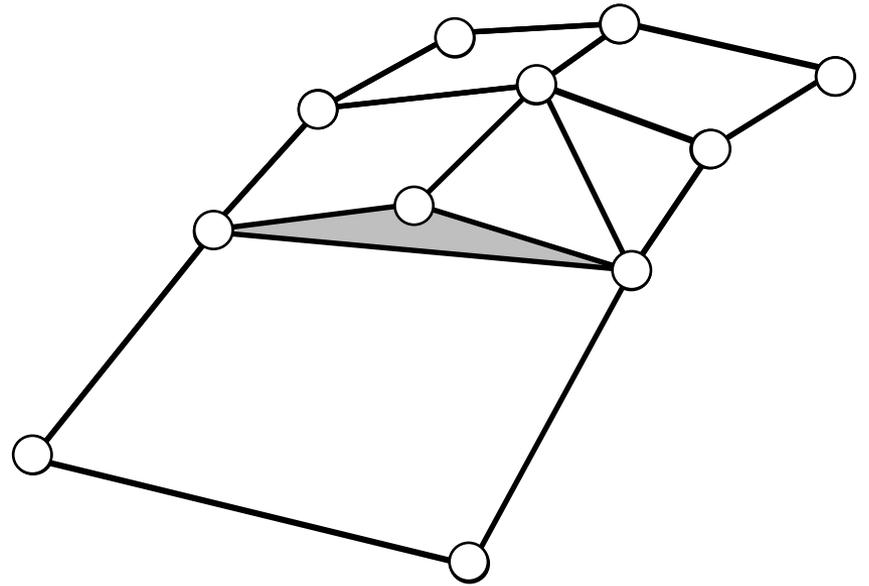
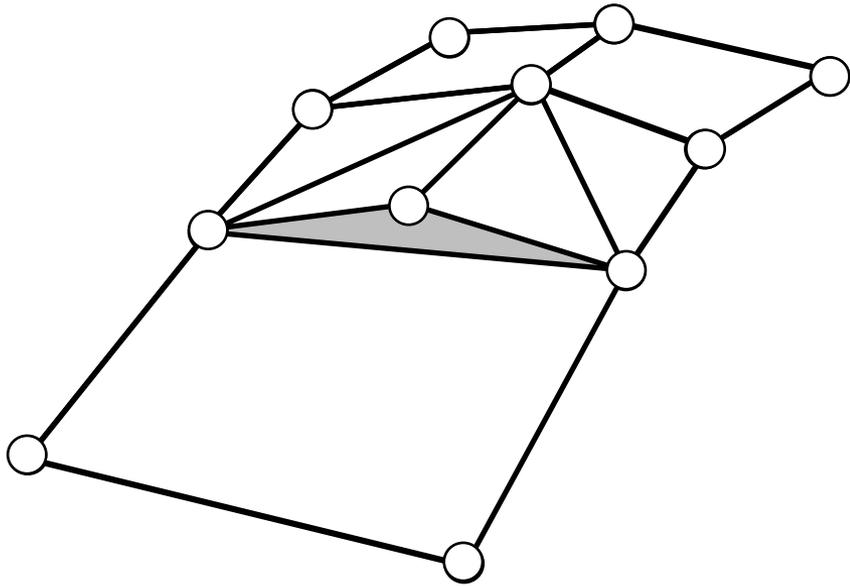
Cracks

Uniform Refinement

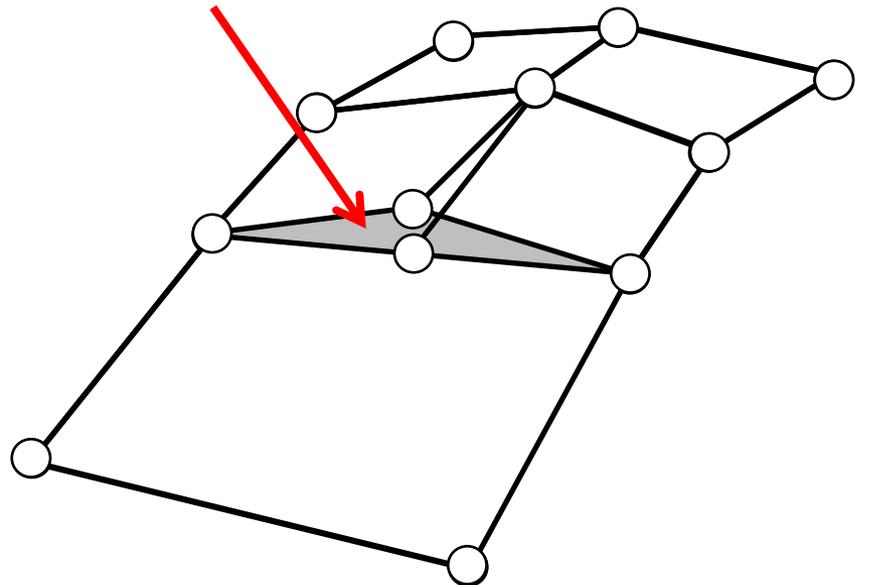


Adaptive Refinement

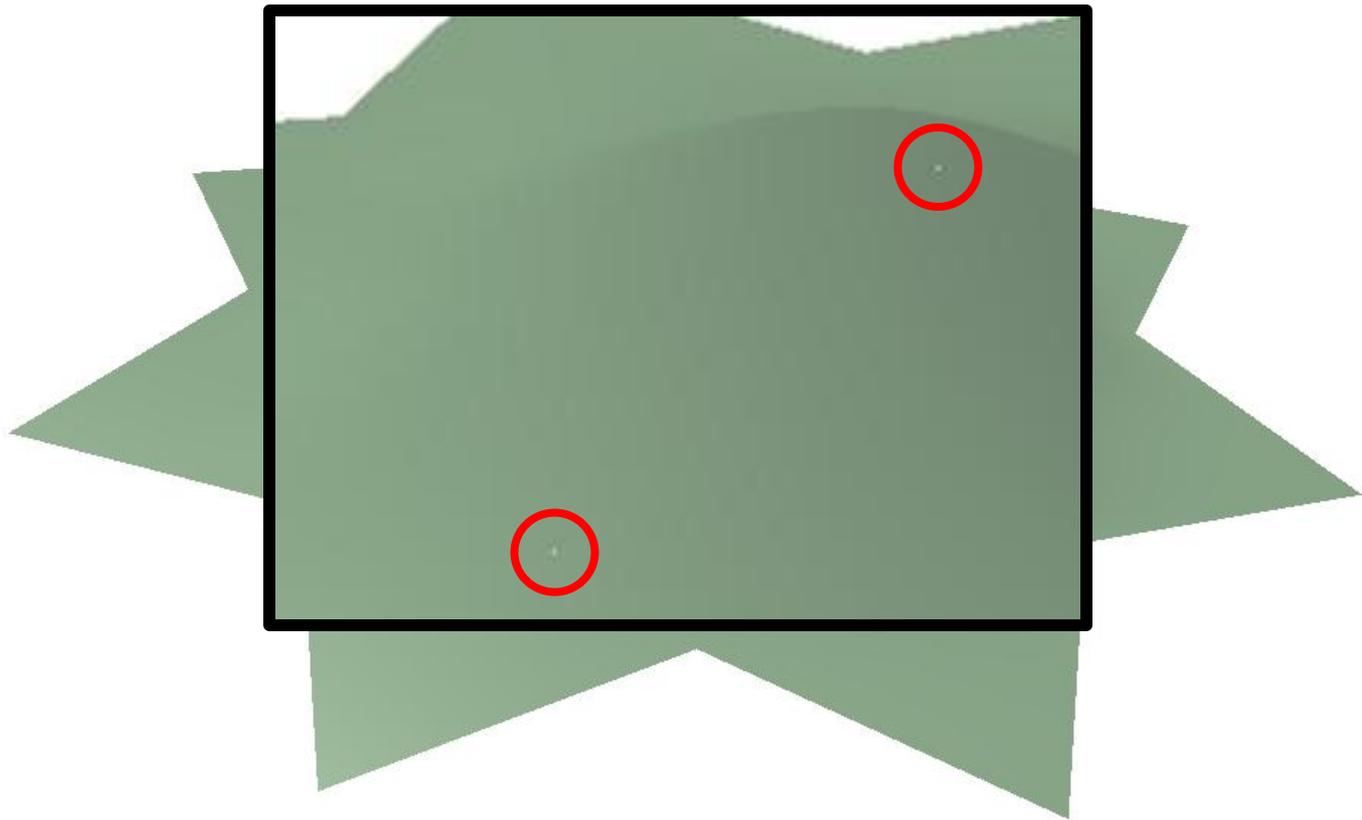


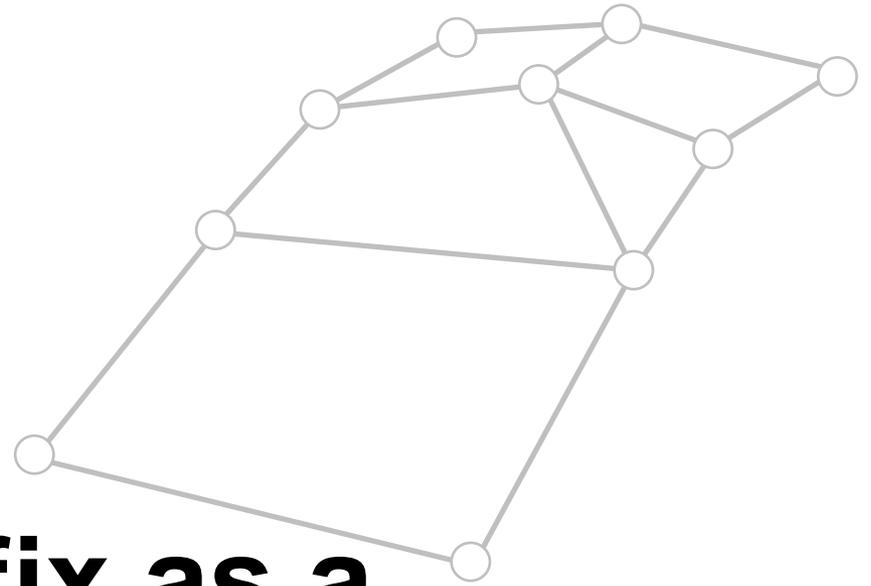
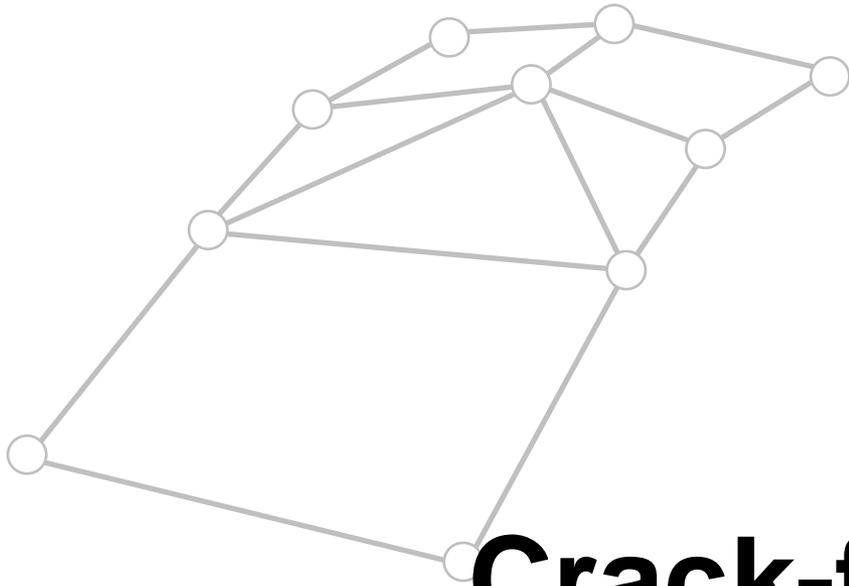


T-Junction

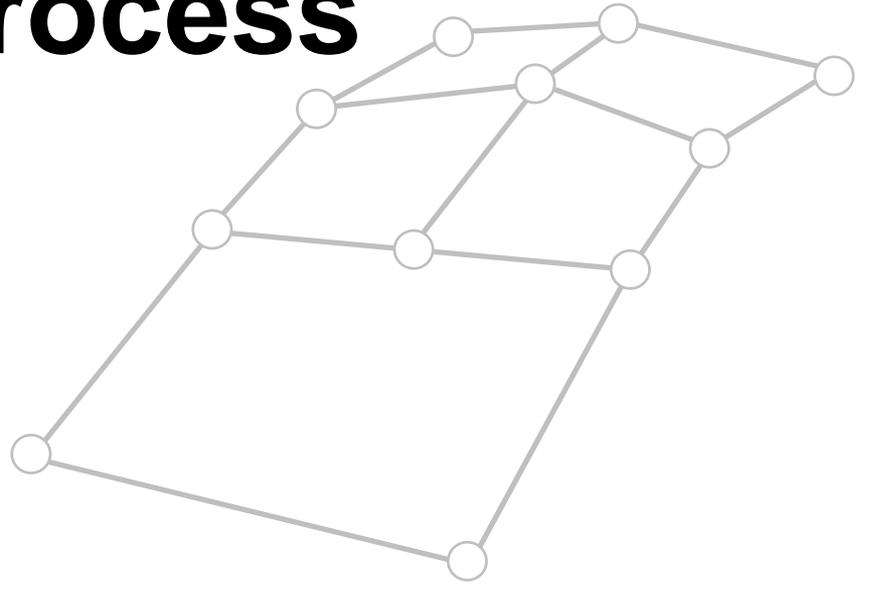
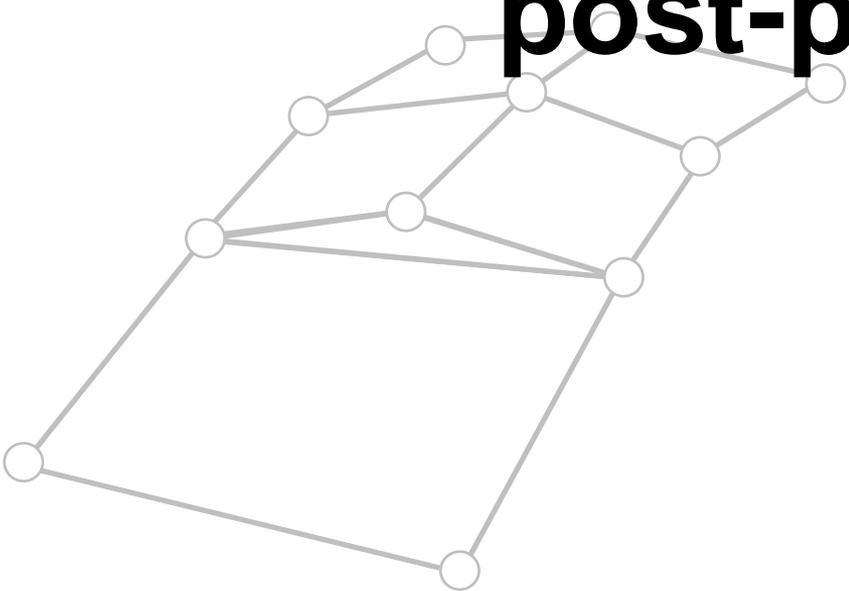


T-Junctions



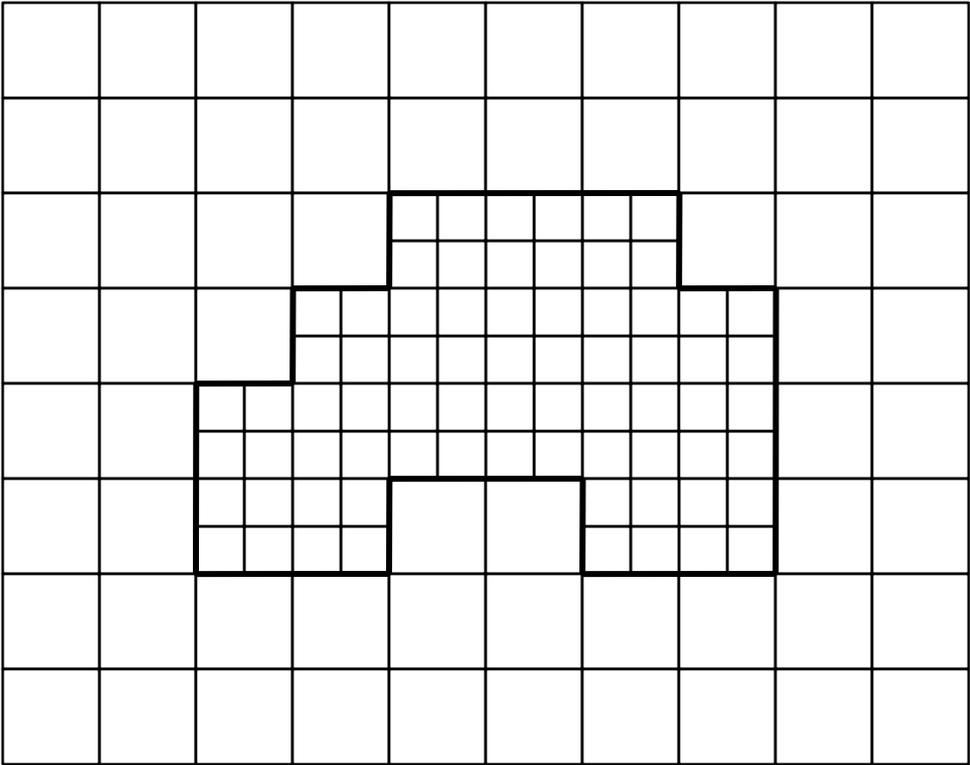
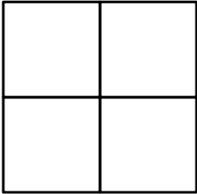


**Crack-fix as a
post-process**



Incrementally Resolving Cracks

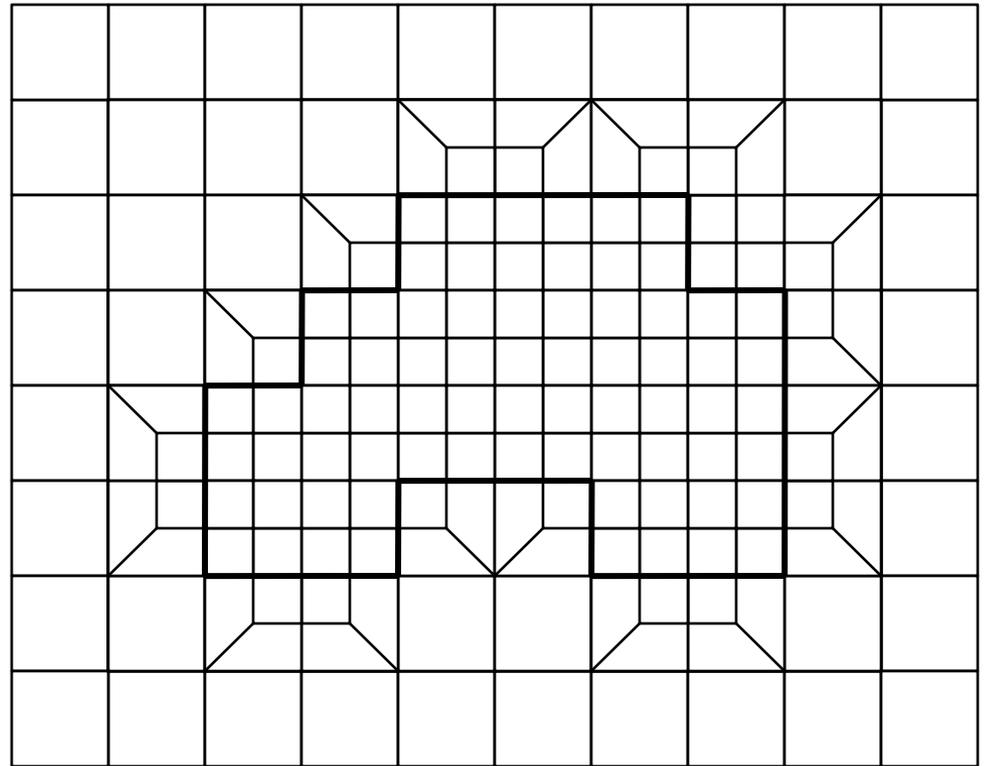
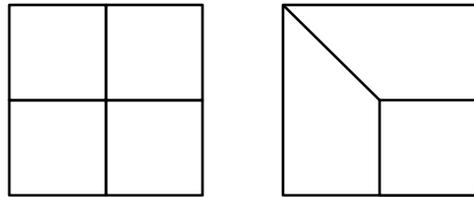
Quad-only Refinement



Incrementally Resolving Cracks

2-Refinement Templates

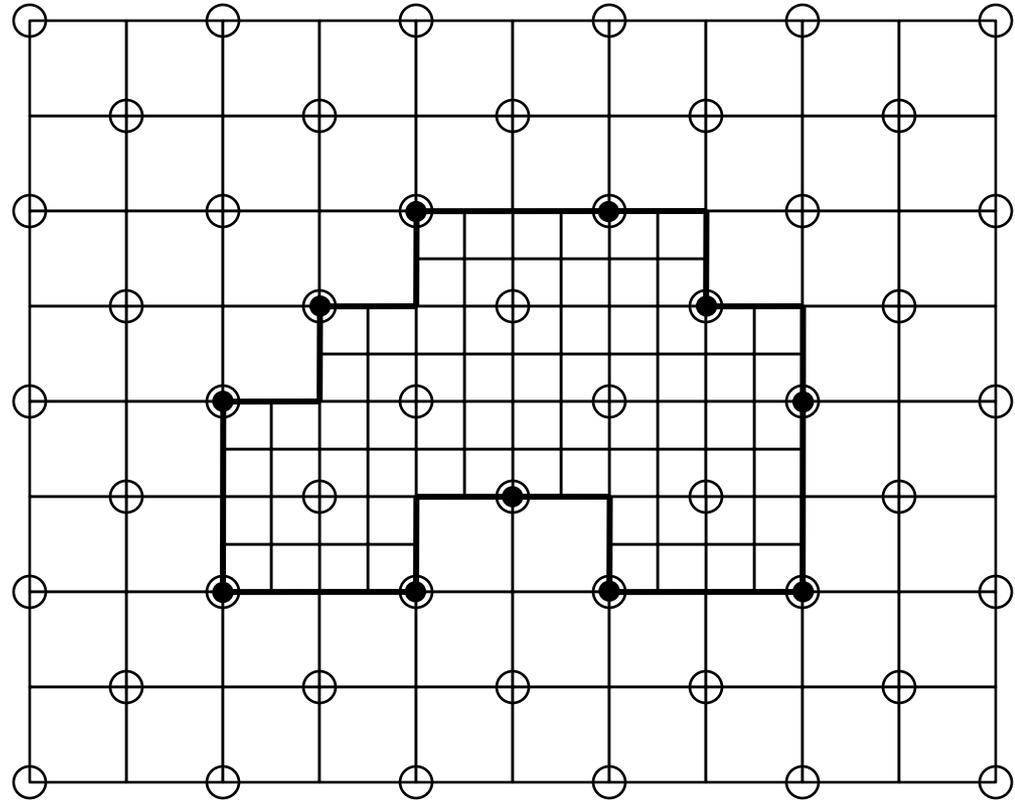
[Schneiders 96]



Incrementally Resolving Cracks

Tagging Active Vertices

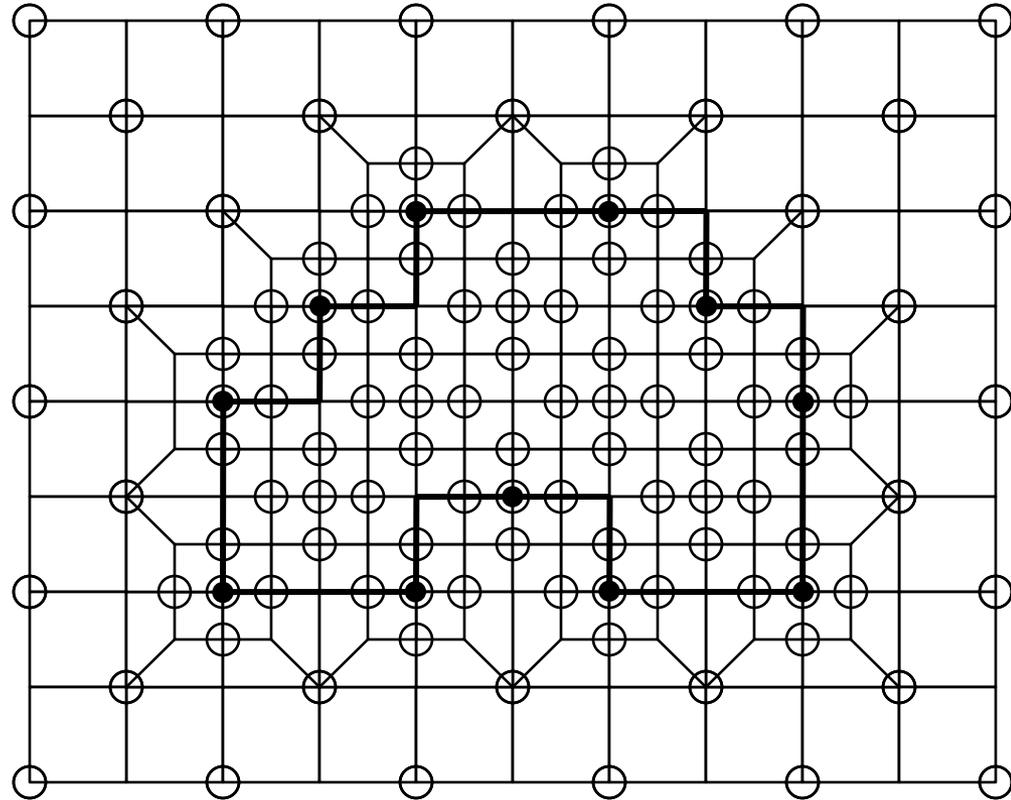
- Initialized potential tags with the base mesh
- Convert to active at transition
- Apply templates
- Update potential tags



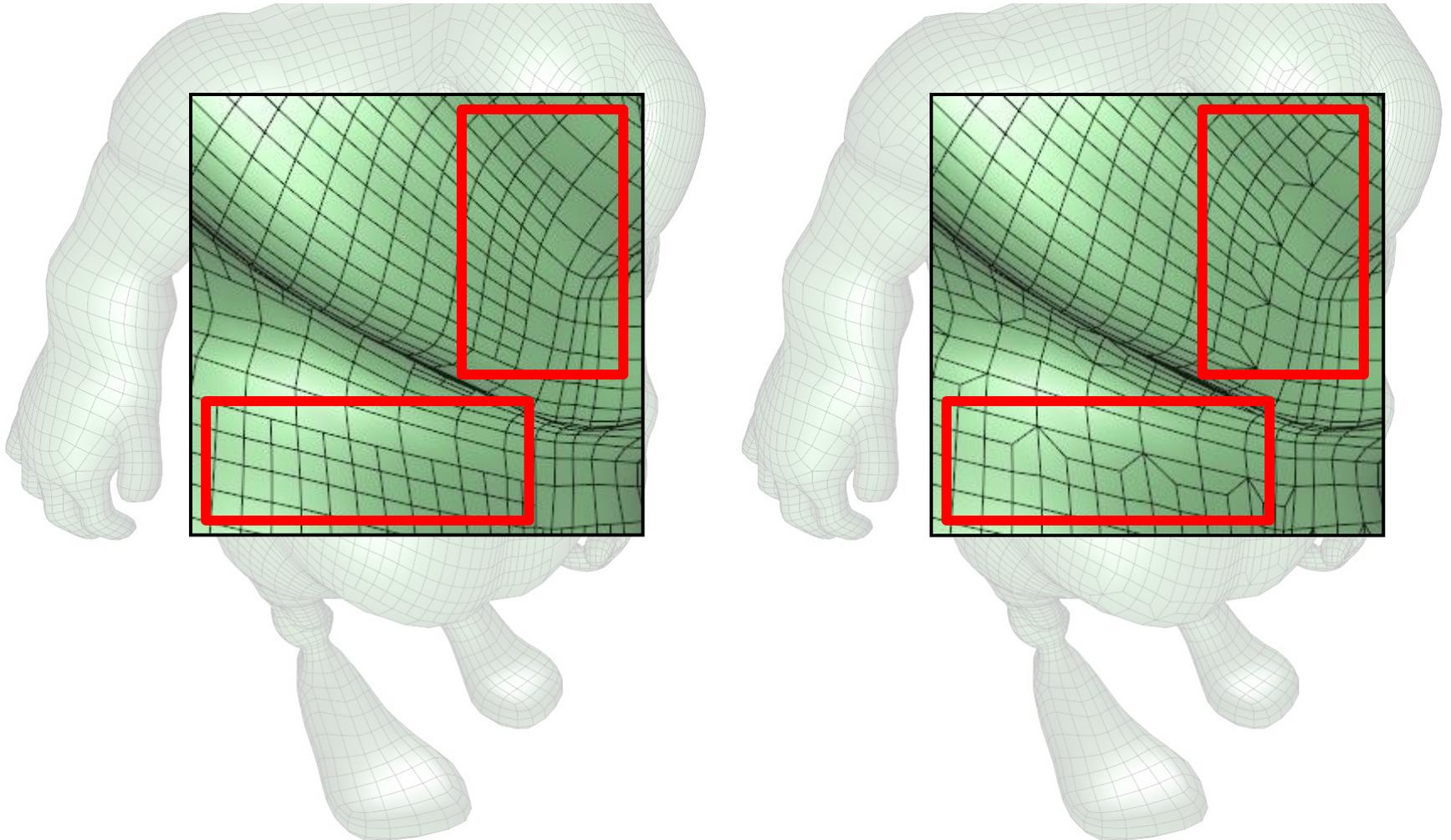
Incrementally Resolving Cracks

Tagging Active Vertices

- Initialized potential tags with the base mesh
- Convert to active at transition
- Apply templates
- Update potential tags



Example



Implementation

Hardware Platform

- NVIDIA GeForce GTX 280

Computing Architecture

- NVIDIA CUDA 2.2

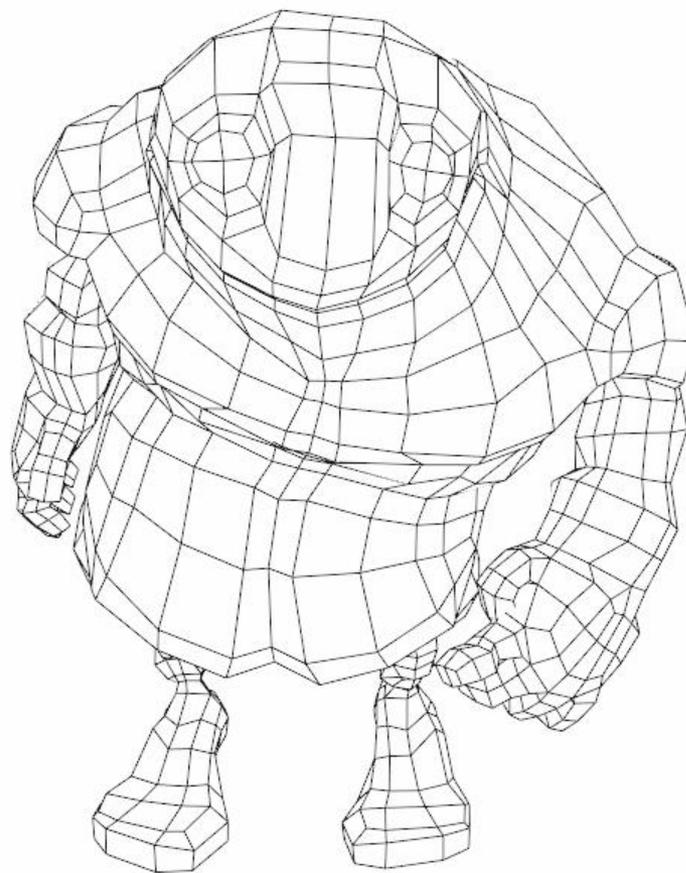
Subdivision criterion used for results

- Screen-space extent

Results



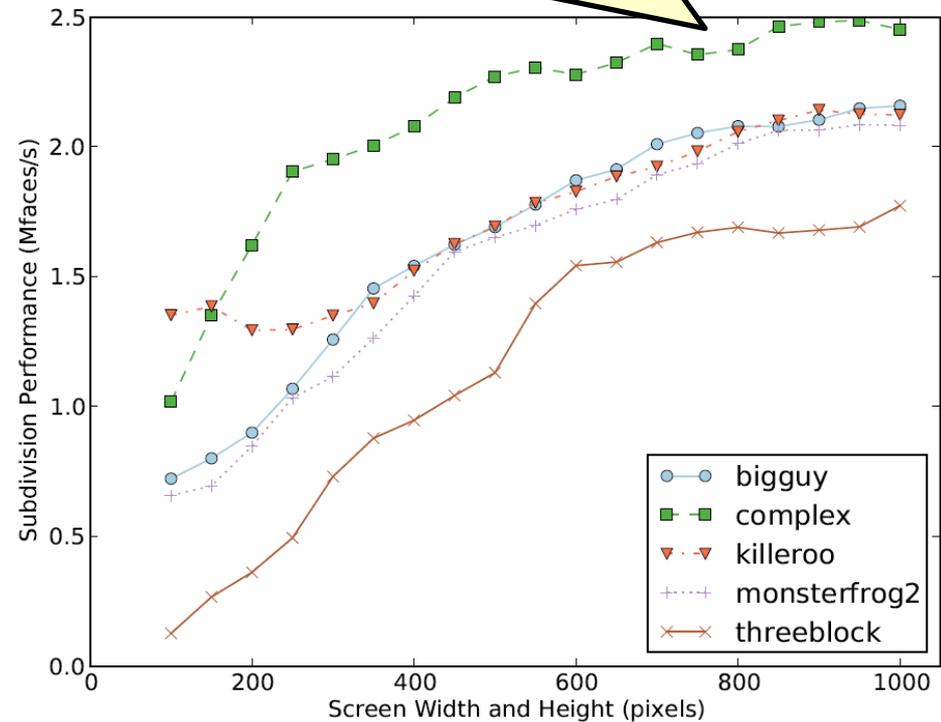
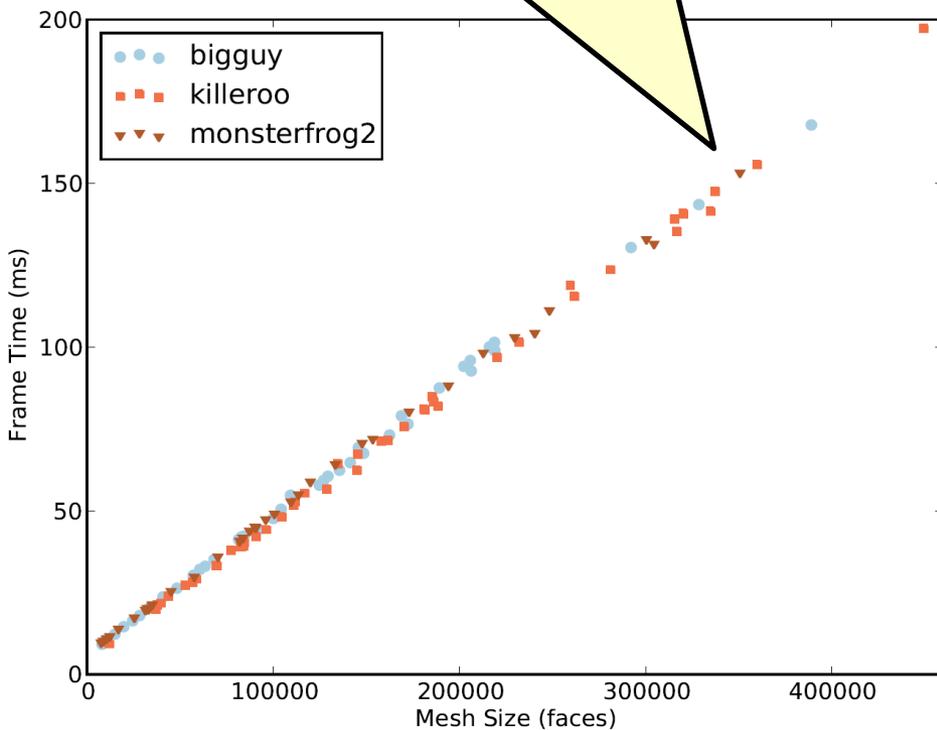
Results



Results – Performance Behavior

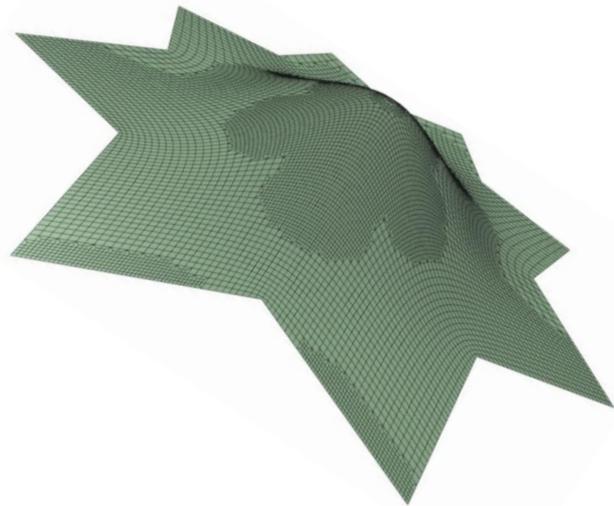
Linear with output complexity

Performance approaches 2.5–3M faces/sec



Summary

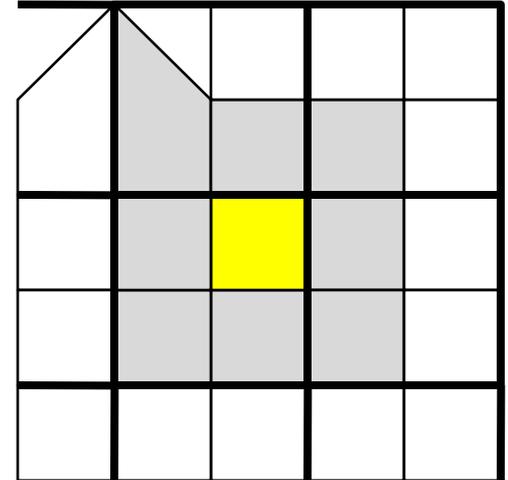
- Parallel GPU tessellation of Catmull-Clark surfaces
- Robust data management for subdivision
- Dynamically view-dependent
- Fixing cracks in parallel



Support for mesh boundaries and textures

Limitations

- Quad-only meshes
 - Bloats off-line storage and transfer
- Crack-fix affects the 1-ring neighborhood
 - Aggressive subdivision can help
 - Rarely a problem in practice
- Memory access patterns
 - Several non-coalesced dependent lookups
 - Implementation limited by memory bandwidth



Future Work

- Extension to alternate refinement schemes
 - Loop
 - Doo-Sabin
- Efficient memory management
 - Programmable geometry caching

Acknowledgments

Feedback and Suggestions

- Eric Lengyel
- Anonymous paper reviewers

Models

- Bay Raitt (Valve Software)
- Headus Inc.

Funding and Equipment

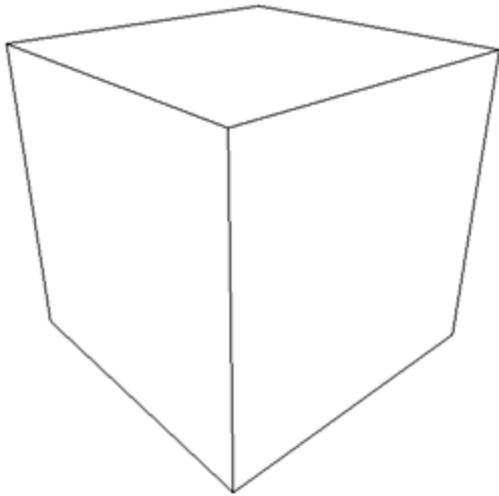
- NSF Award 0541448
- SciDAC Institute for Ultrascale Visualization
- NVIDIA

Thanks!

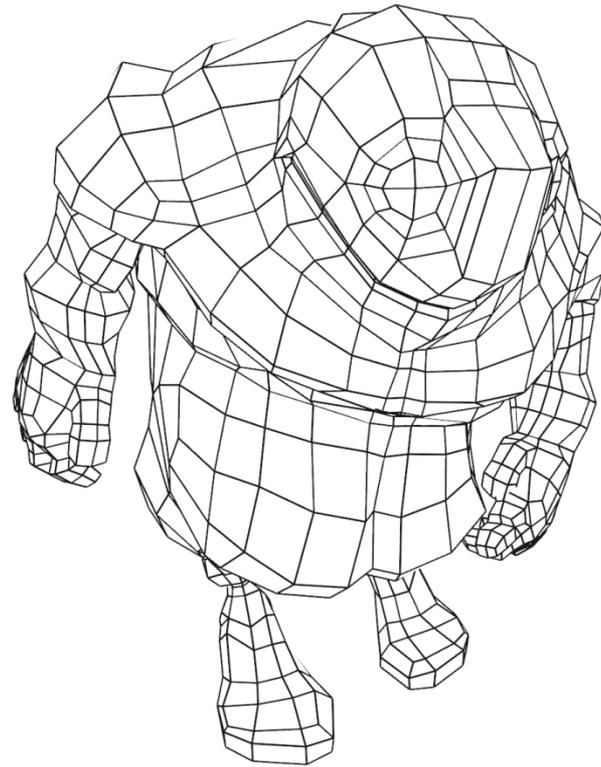
EXTRA SLIDES

Breadth-first Vs Depth-first Subdivision

- One thread for one face
 - Limited parallelism



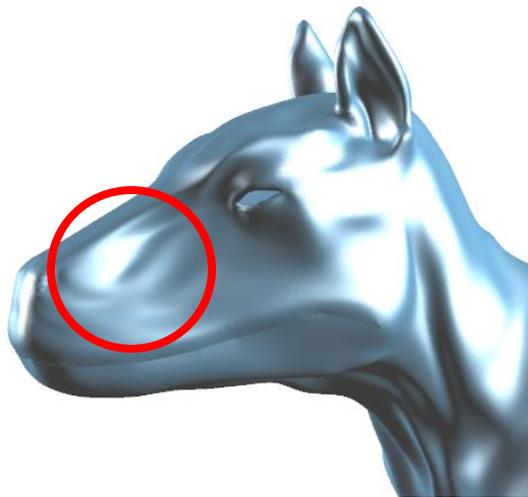
Initial number
of faces = 6



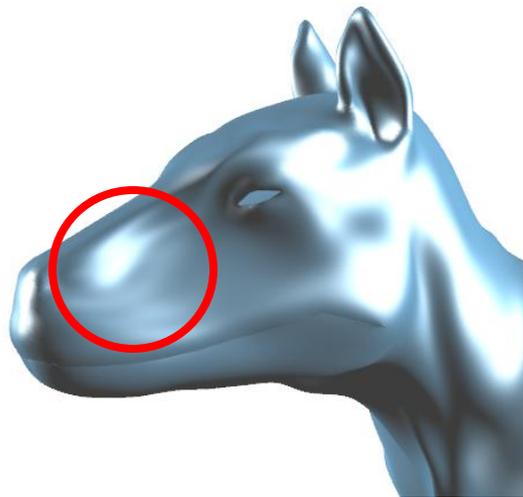
Initial number
of faces = 1450

Calculating vertex normals

- Two methods
 1. Averaged normals of adjacent faces
 - a. Regular mean
 - b. Weighted mean
 2. Subdivision shading



1A

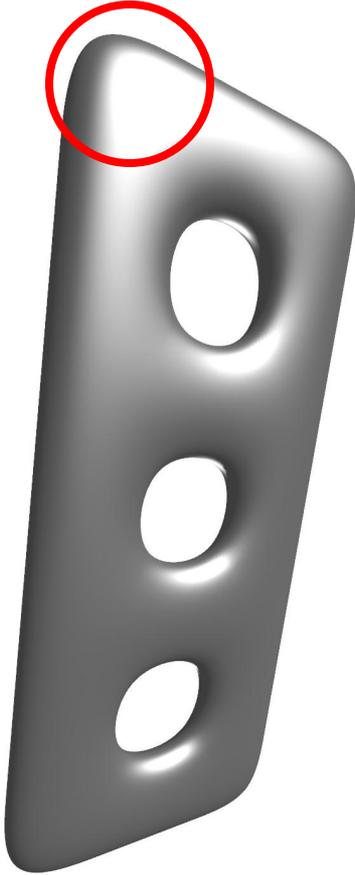


1B

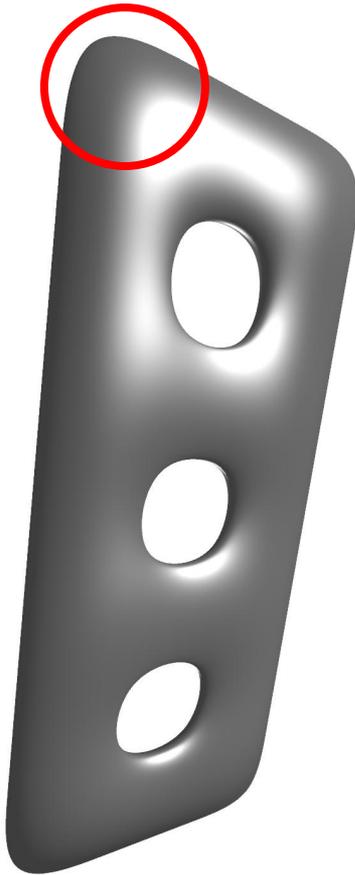


2

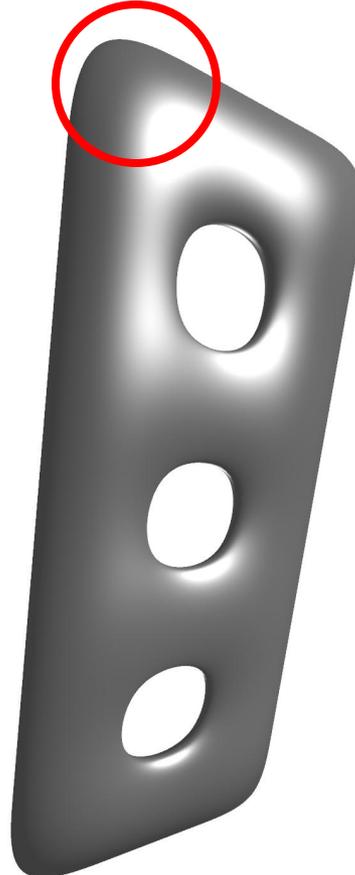
Calculating vertex normals



1A

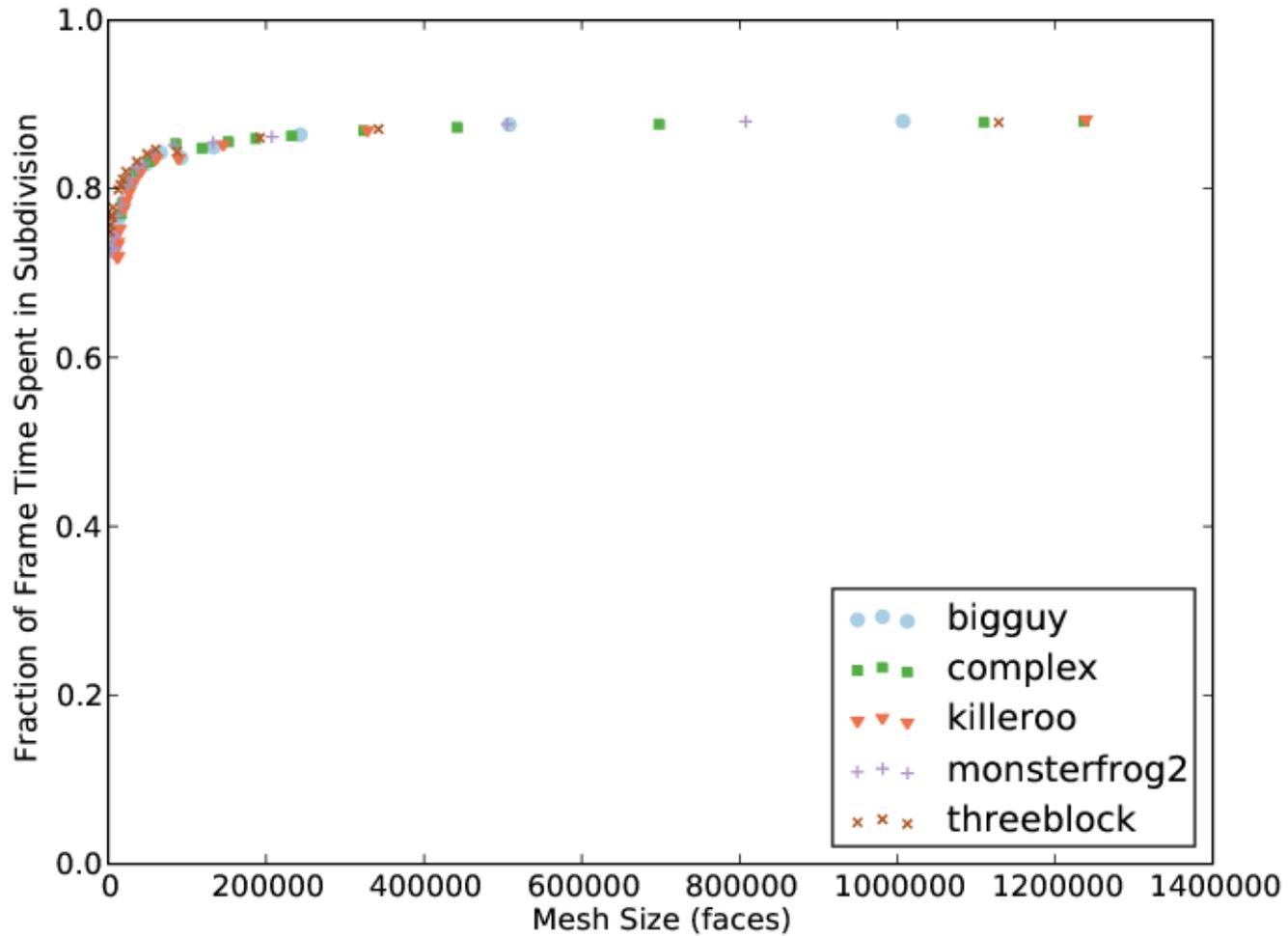


1B

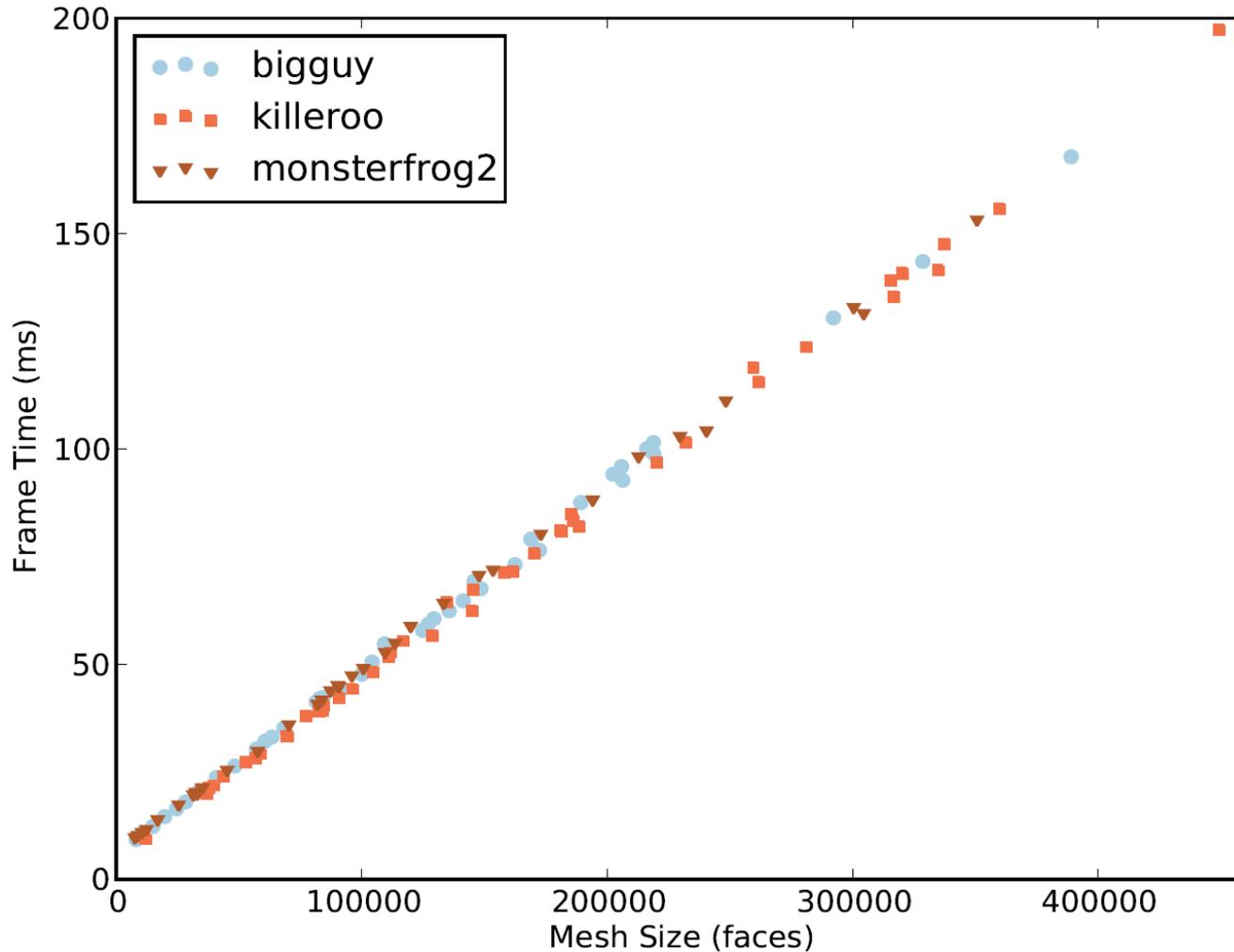


2

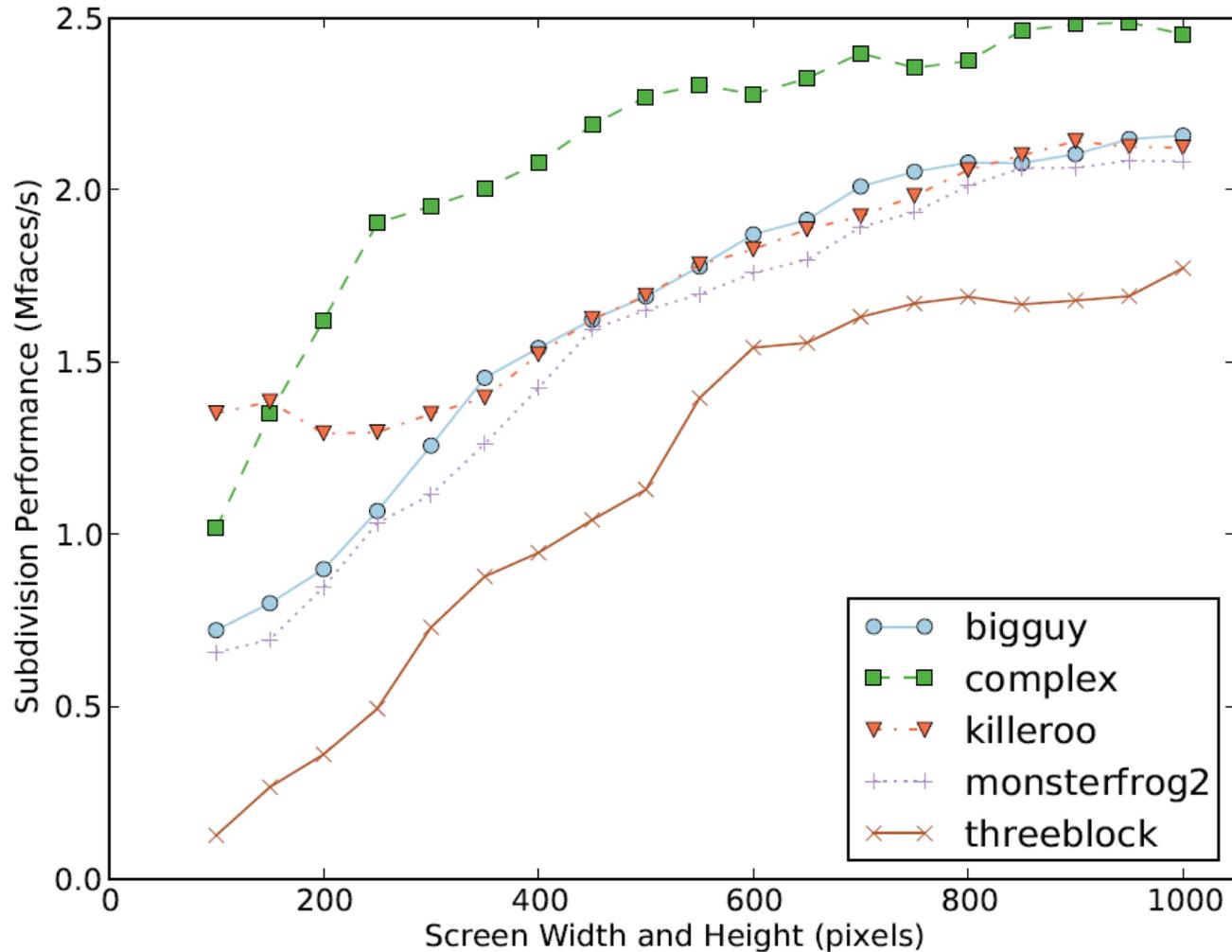
Frame time division



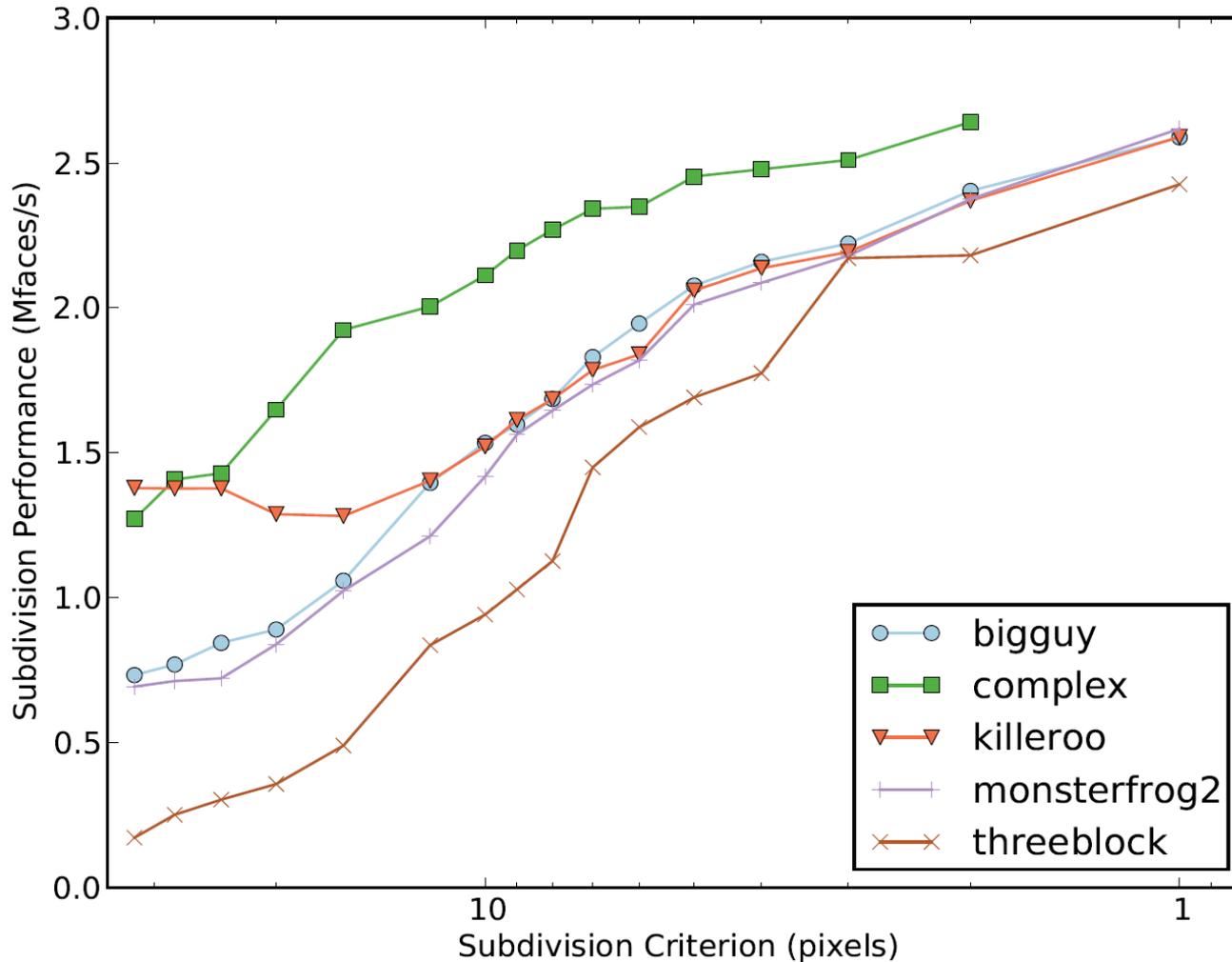
Results – Performance Scaling



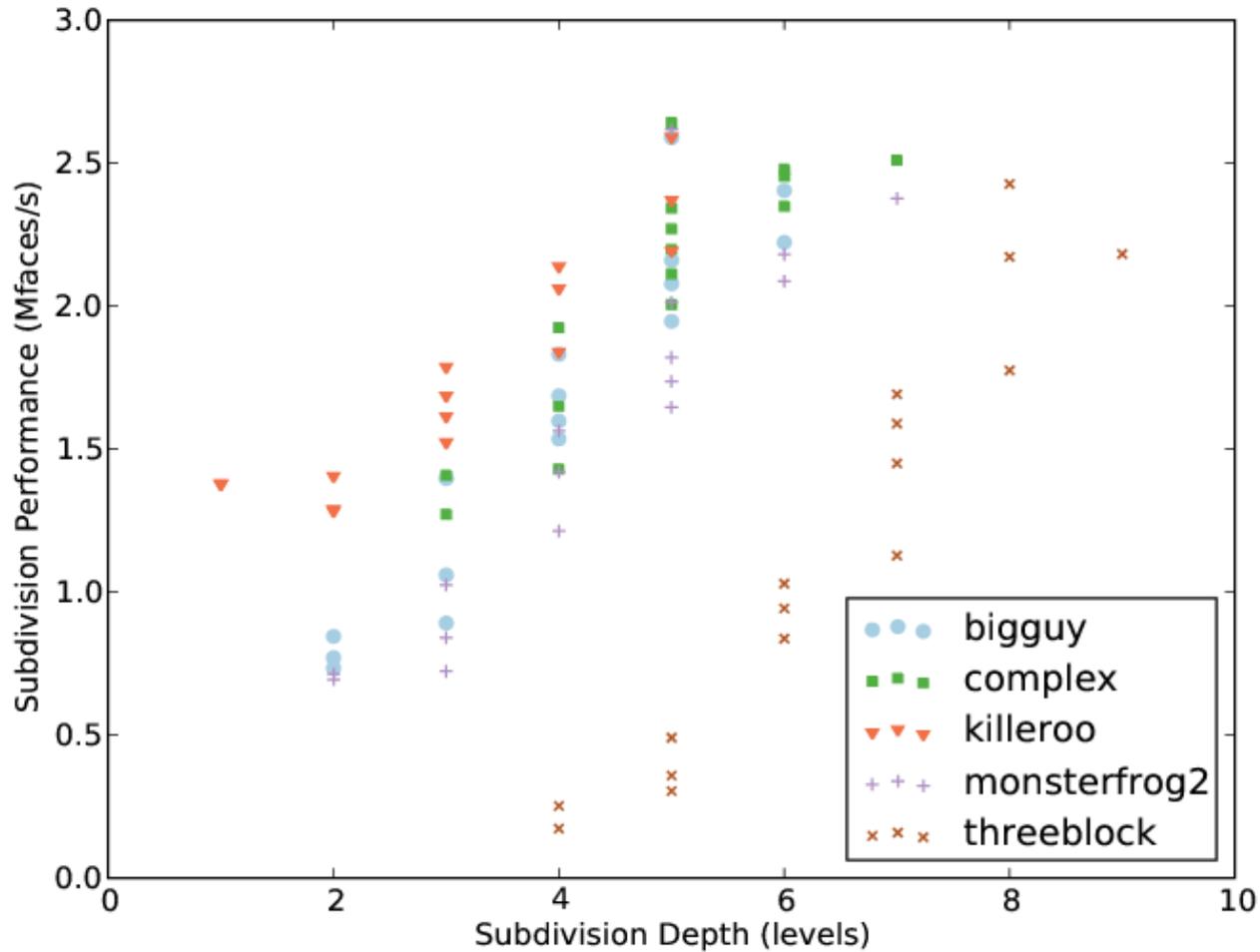
Results – Performance Scaling



Results – Performance Scaling



Varying subdivision depth



Need for silhouette enhancement

