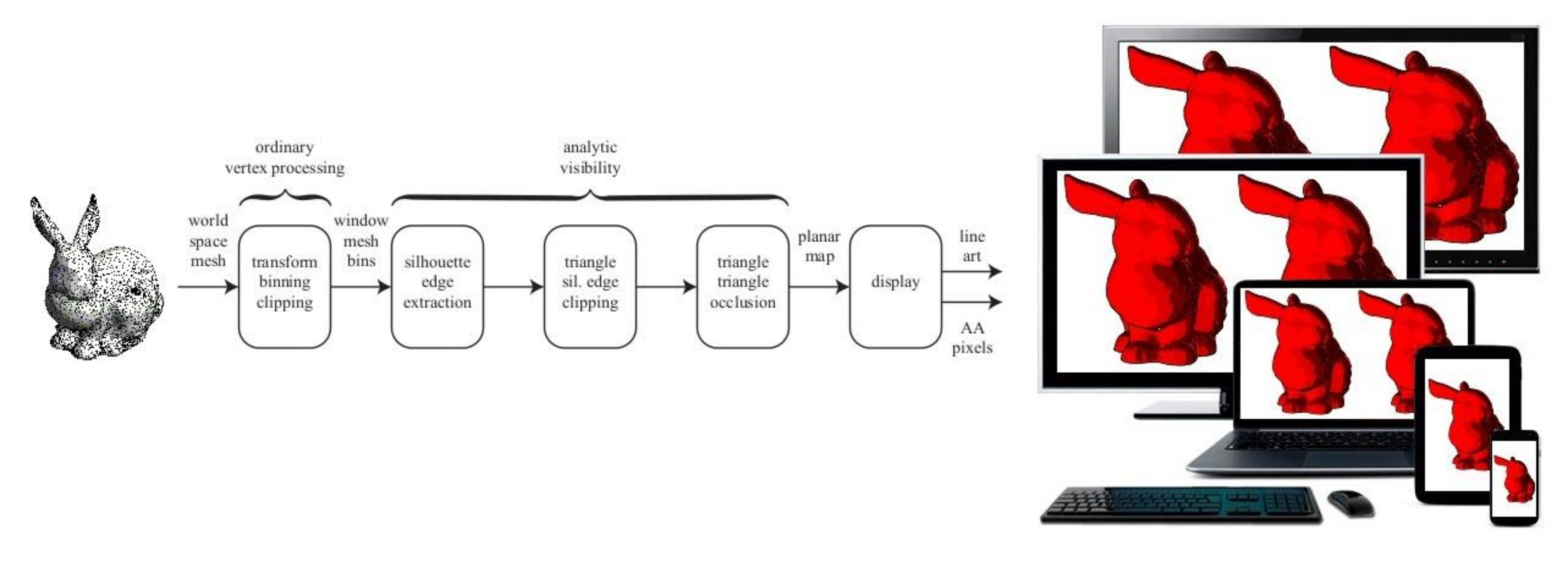
SVGPU
Real Time 3D Rendering to Vector Graphics Formats

Apollo I. Ellis *University of Illinois* (Presenting)
Warren Hunt *Oculus Research*
John C. Hart *University of Illinois*

SVGPU (Scalable Vector Graphics on the GPU)

- Renders vector images from 3D scenes, fast
- Applications in client server graphics domain

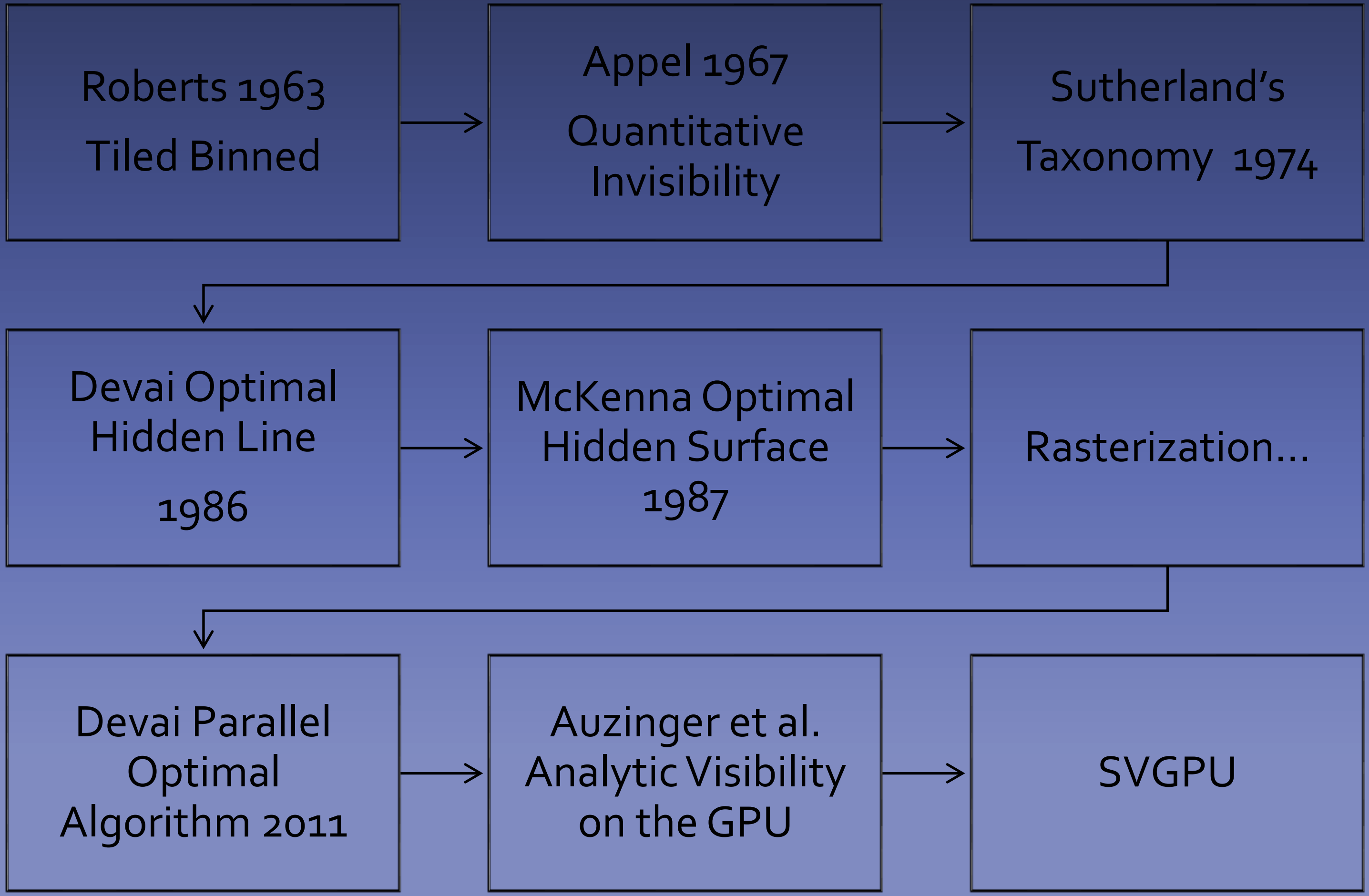


ordinary vertex processing
analytic visibility
world space mesh
transform binning clipping
window mesh bins
silhouette edge extraction
triangle sil. edge clipping
triangle triangle occlusion
planar map
display
line art
AA pixels

Bunny via cs.mun.ca [Mun]

Display Image via image search [Google]

Hidden Surfaces



Roberts 1963
Tiled Binned

Appel 1967
Quantitative Invisibility

Sutherland’s
Taxonomy 1974

Devai Optimal Hidden Line
1986

McKenna Optimal Hidden Surface 1987

Rasterization…

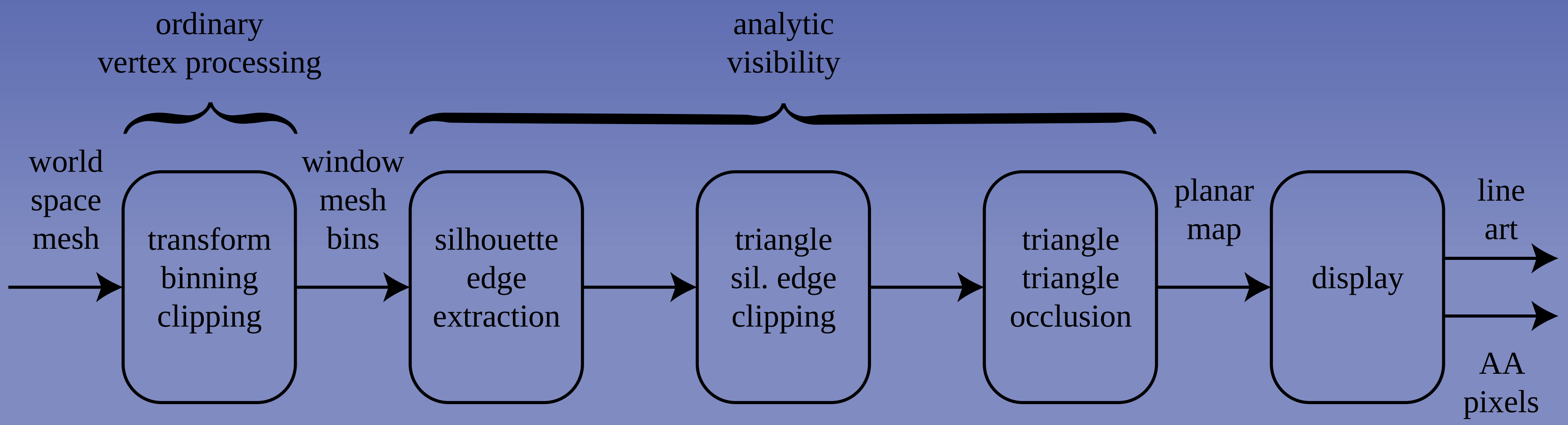
Devai Parallel Optimal Algorithm 2011

Auzinger et al. Analytic Visibility on the GPU

SVGPU

Pipeline

- Vertex shade and bin to screen tiles
- Hash edges and extract silhouettes
- Clip triangles to silhouette edges
- Check for occlusion


ordinary vertex processing
analytic visibility
world space mesh
transform binning clipping
window mesh bins
silhouette edge extraction
triangle sil. edge clipping
triangle triangle occlusion
planar map
display
line art
AA pixels

Silhouette Edge Extraction

- Hash all triangles by each edge
- Sweep the hash buckets
- Check collisions for front-back pairs
- Bin silhouette edges by screen tile

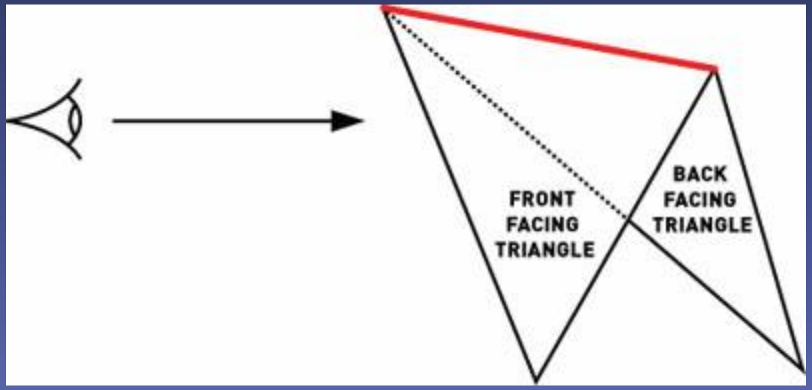
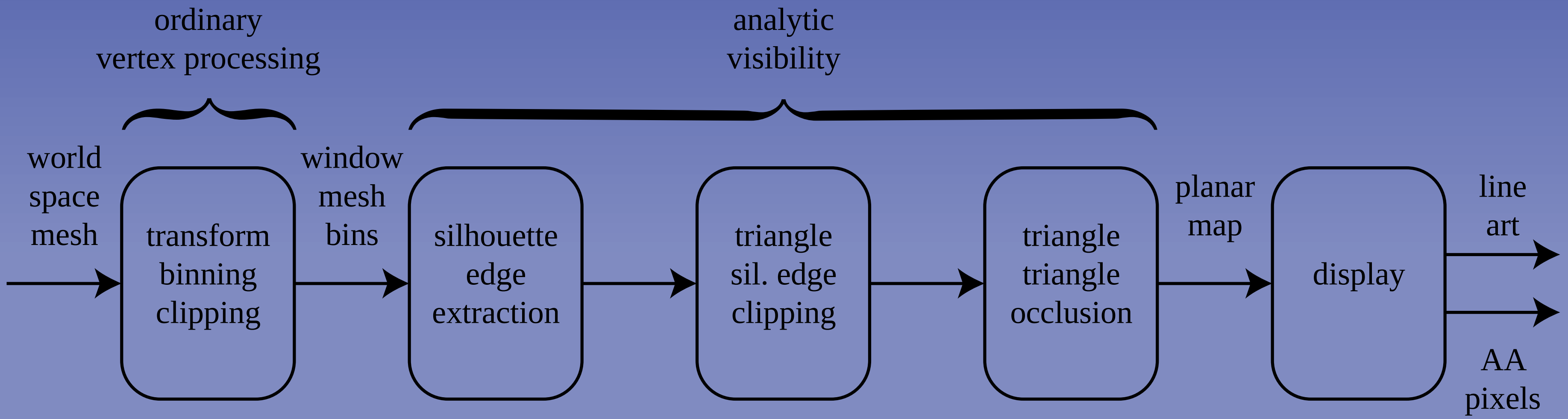


Image by Joshua Doss [JDoss]



ordinary vertex processing

analytic visibility

world space mesh

transform binning clipping

window mesh bins

silhouette edge extraction

triangle sil. edge clipping

triangle triangle occlusion

planar map

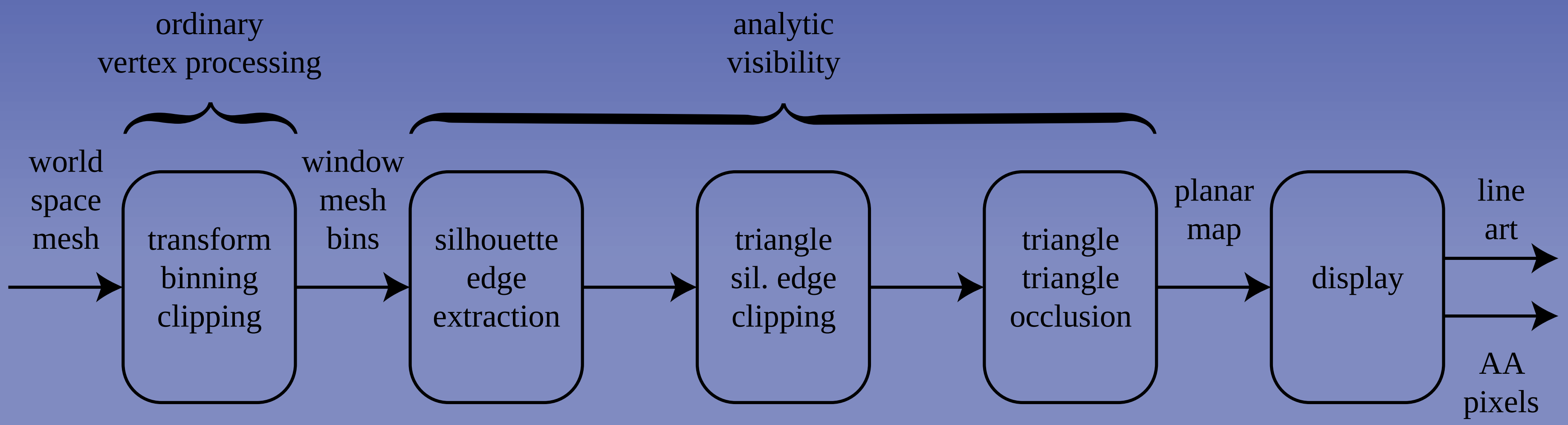
display

line art

AA pixels

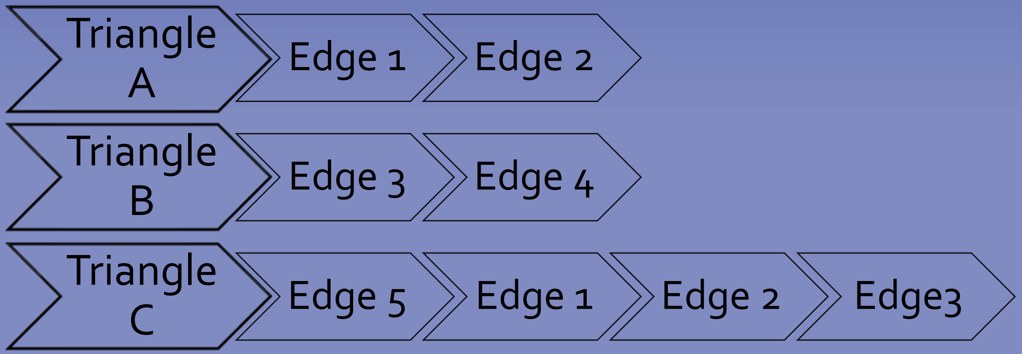
Clip Setup

- Dynamic parallelism parent kernel
- One thread per bin.. Say 64..
- Each thread runs a bin's MxN clipping kernel
- Each thread runs a bin's N'xN occlusion kernel


ordinary vertex processing
analytic visibility
world space mesh
transform binning clipping
window mesh bins
silhouette edge extraction
triangle sil. edge clipping
triangle triangle occlusion
planar map
display
line art
AA pixels

Trivial Rejection

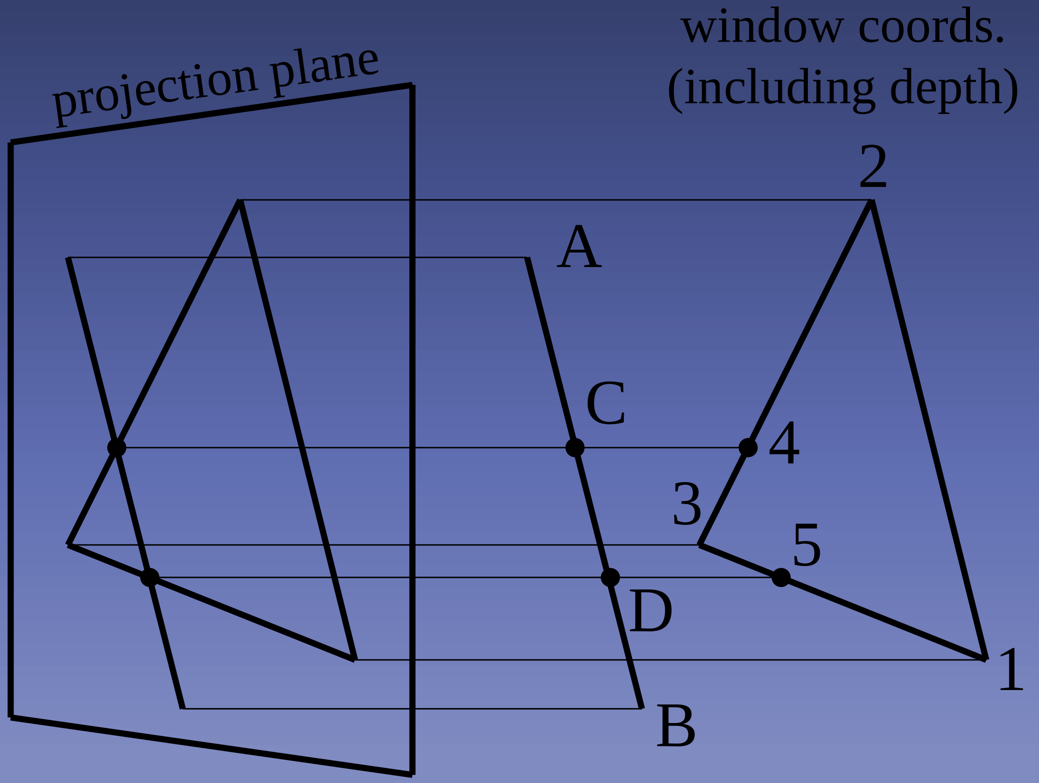
- If AB lies outside 12, 23, or 31
  - Reject.
- If 1,2 and 3 lie outside AB
  - Reject.
- Gather all accepted pairs (AB,123)
- Construct adjacency list for clipper



Triangle A → Edge 1 → Edge 2

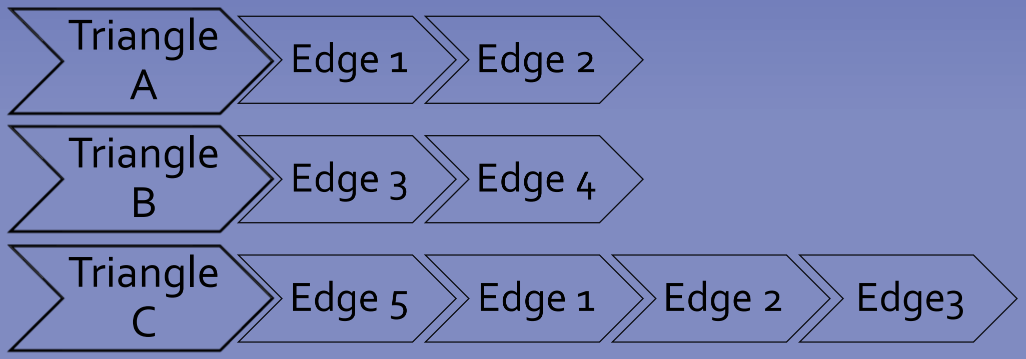
Triangle B → Edge 3 → Edge 4

Triangle C → Edge 5 → Edge 1 → Edge 2 → Edge3



Clipping

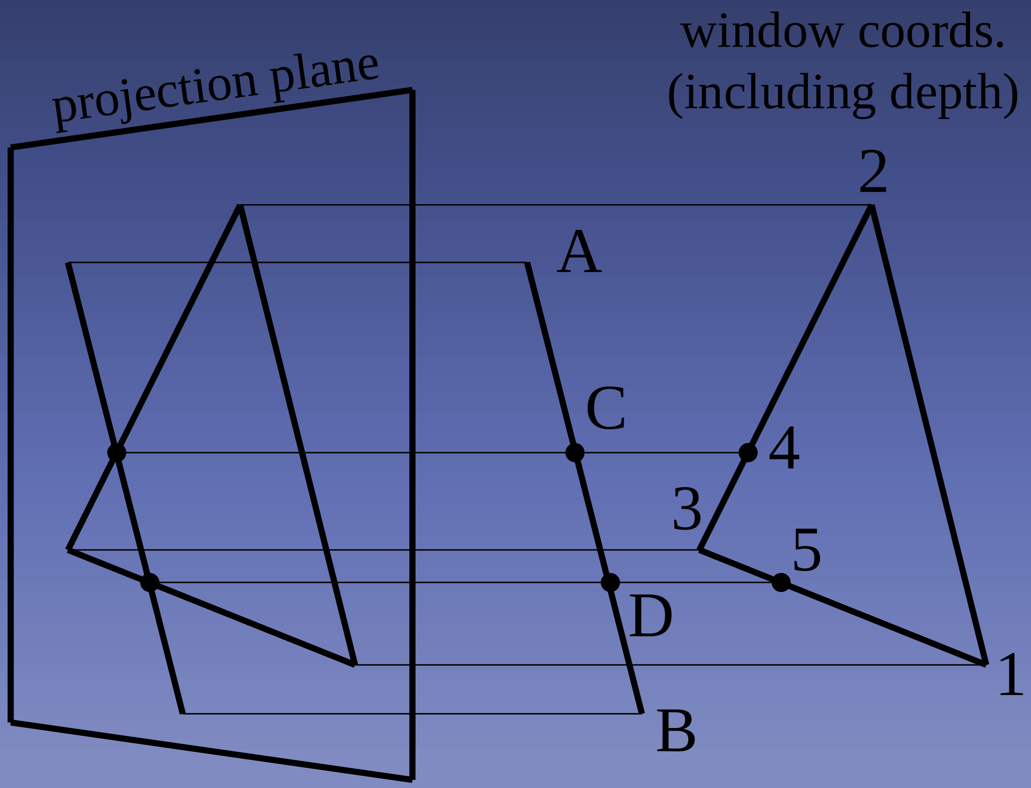
- For all triangles in adjacency
  - Sutherland-Hodgman [BF09]
  - Walk the vertices in turn
  - Classify vertices as In, Out, or On
  - 3 Vertices for ambiguous cases
  - LUT specifies behavior for each edge

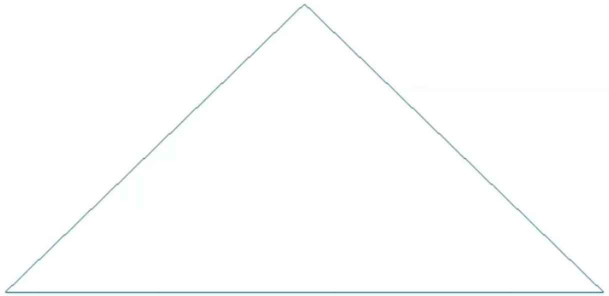


Triangle A — Edge 1 — Edge 2

Triangle B — Edge 3 — Edge 4

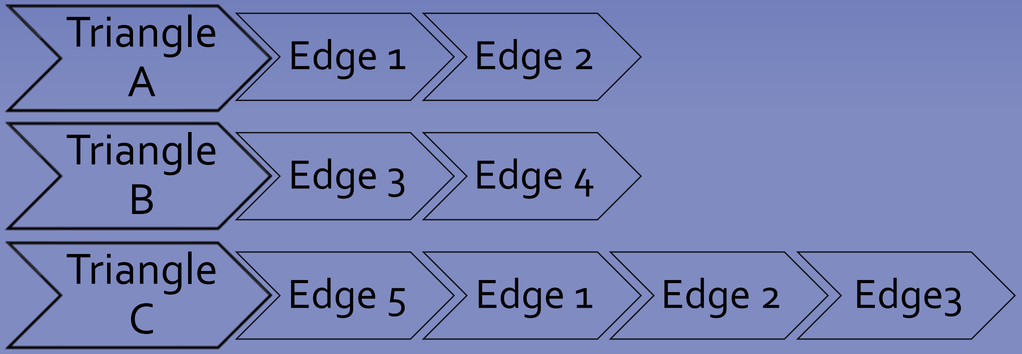
Triangle C — Edge 5 — Edge 1 — Edge 2 — Edge3





Clipping

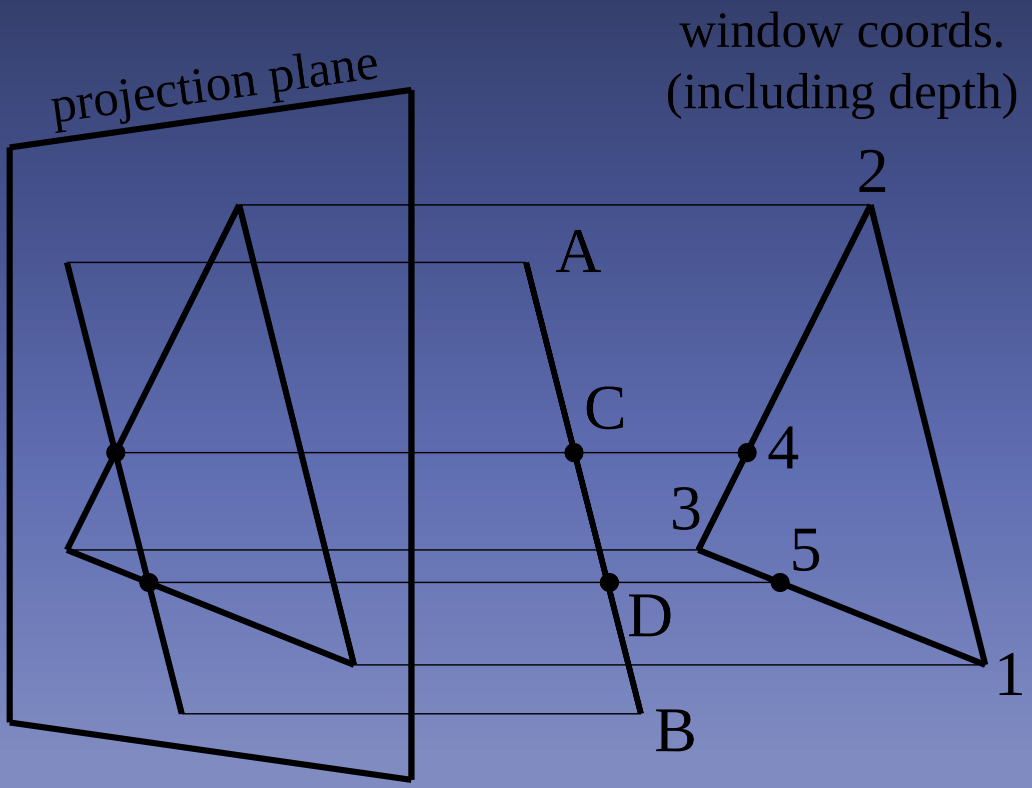
- While(round < longest list)
  - Clip all triangles to next edge
  - Never reuse 4 or 5 for clipping



Triangle A → Edge 1 → Edge 2

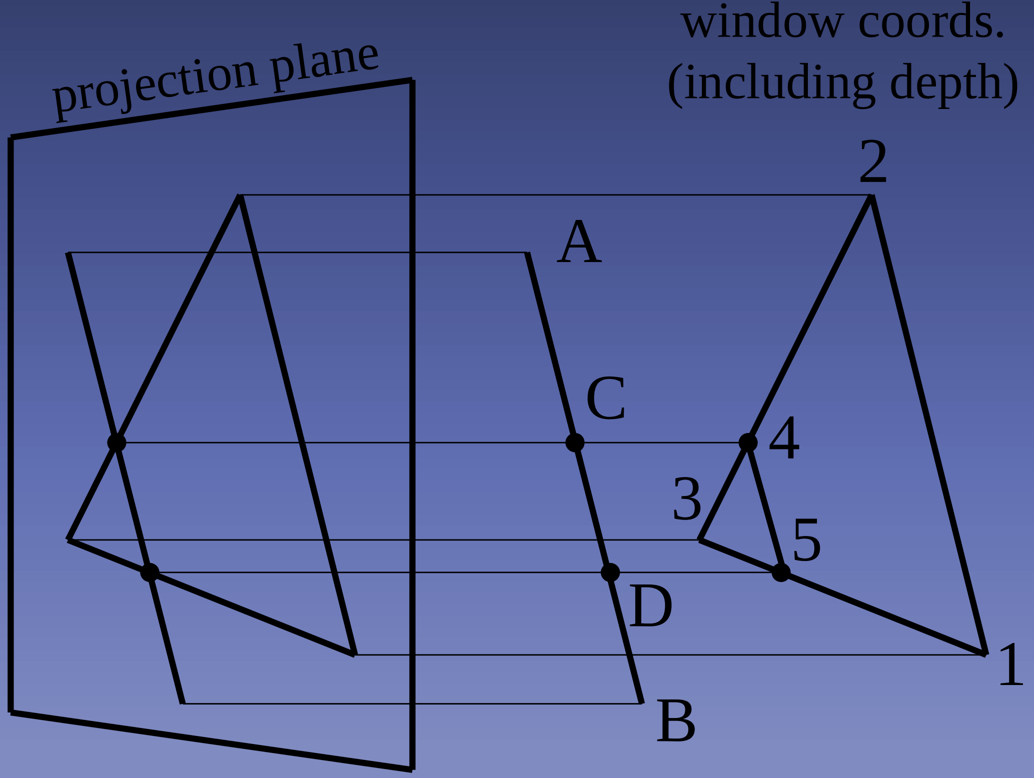
Triangle B → Edge 3 → Edge 4

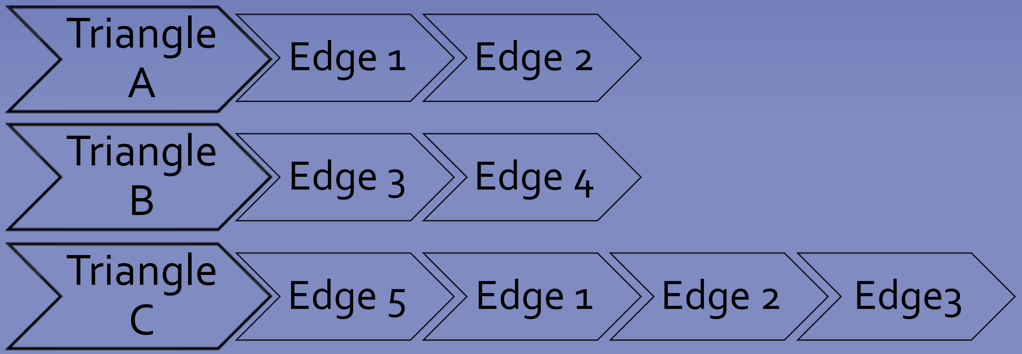
Triangle C → Edge 5 → Edge 1 → Edge 2 → Edge3



Clipping

- While(round < longest list)
  - Clip all triangles to next edge
  - Never reuse 4 or 5 for clipping
  - Consider polygon 1245





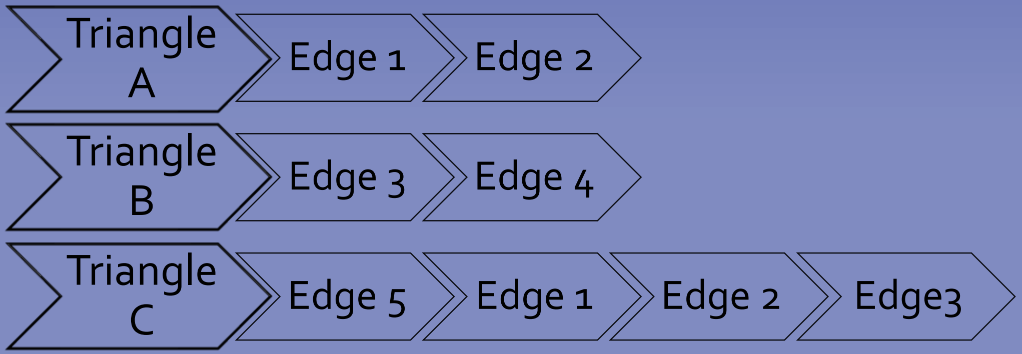
Triangle A
Edge 1
Edge 2

Triangle B
Edge 3
Edge 4

Triangle C
Edge 5
Edge 1
Edge 2
Edge3

Clipping

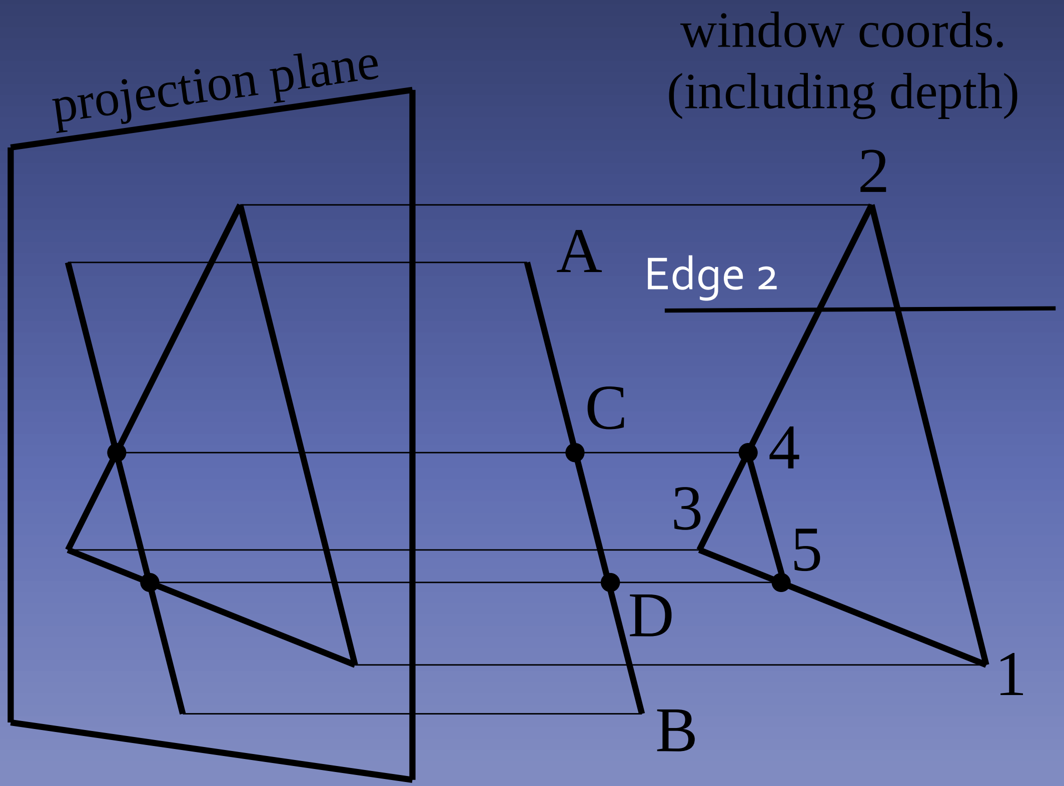
- While(round < longest list)
  - Clip all triangles to next edge
  - Never reuse 4 or 5 for clipping
  - Consider polygon 1245
  - Clipped by edge 2



Triangle A | Edge 1 | Edge 2

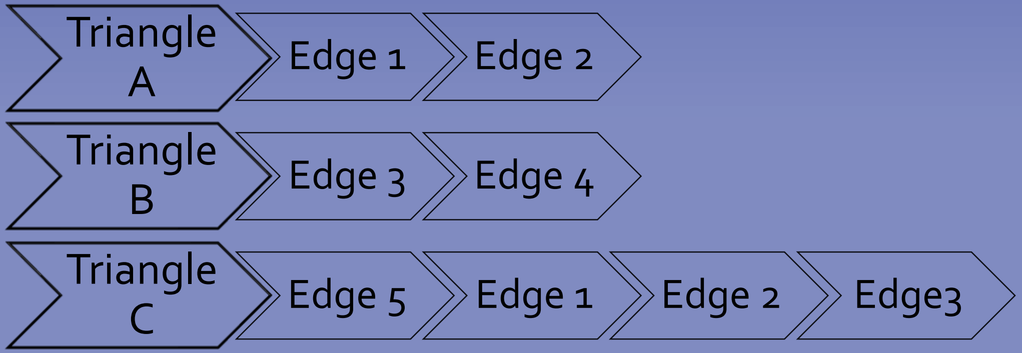
Triangle B | Edge 3 | Edge 4

Triangle C | Edge 5 | Edge 1 | Edge 2 | Edge3



Clipping

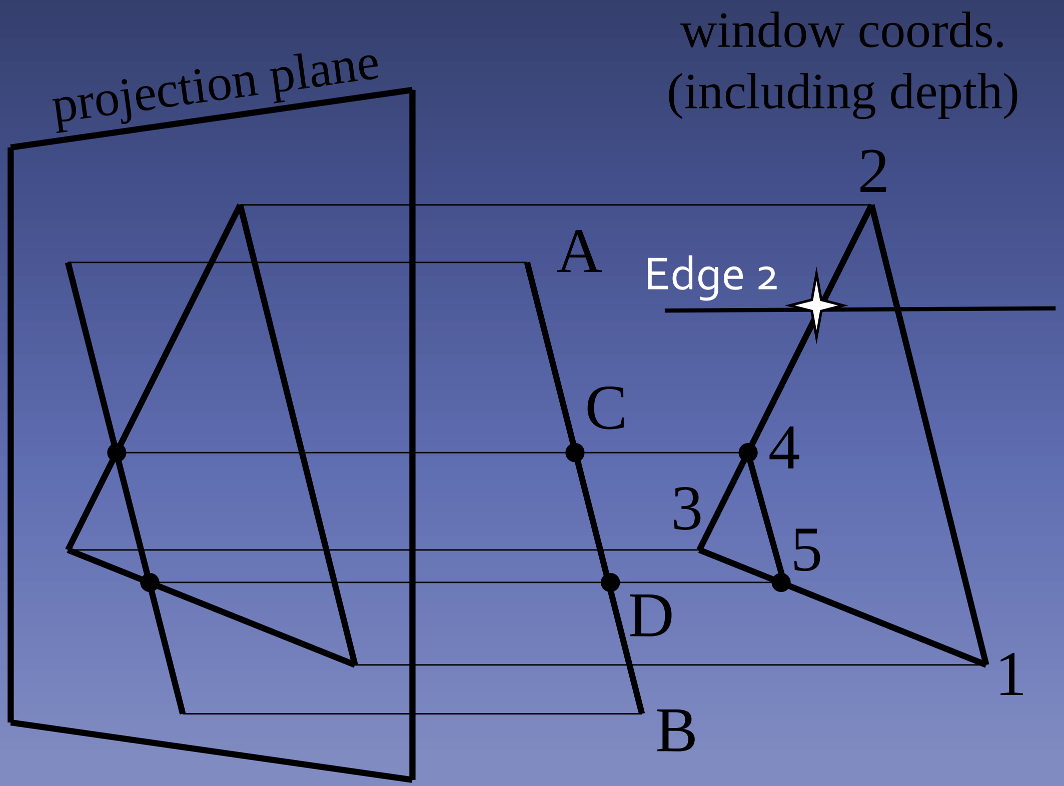
- While(round < longest list)
  - Clip all triangles to next edge
  - Never reuse 4 or 5 for clipping
  - Consider polygon 1245
  - Clipped by edge 2
  - Must use original edge 23 not 24



Triangle A → Edge 1 → Edge 2

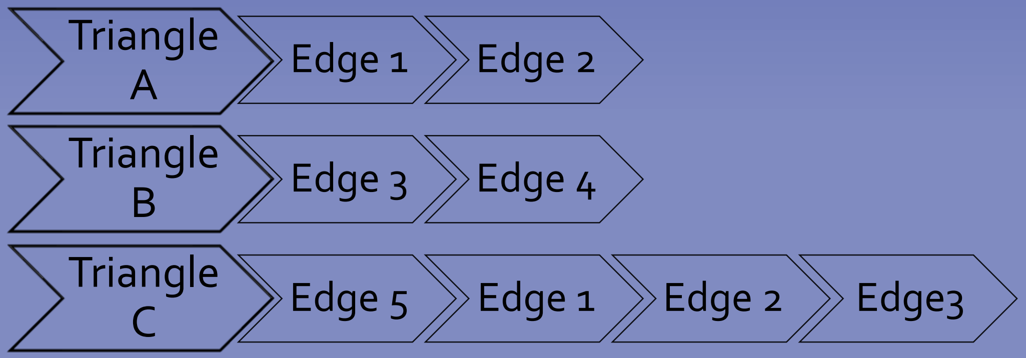
Triangle B → Edge 3 → Edge 4

Triangle C → Edge 5 → Edge 1 → Edge 2 → Edge3



Clipping

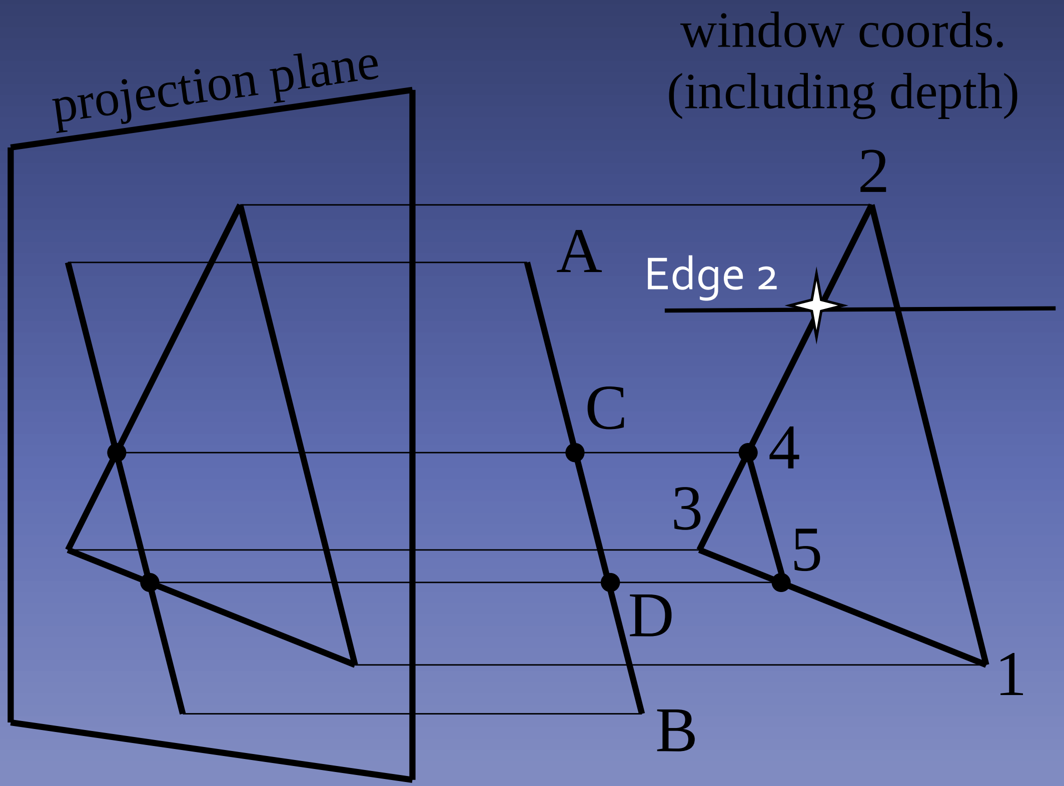
- While(round < longest list)
  - Clip all triangles to next edge
  - Never reuse 4 or 5 for clipping
  - Consider polygon 1245
  - Clipped by edge 2
  - Must use original edge 23 not 24
  - LUT diverges here



Triangle A → Edge 1 → Edge 2

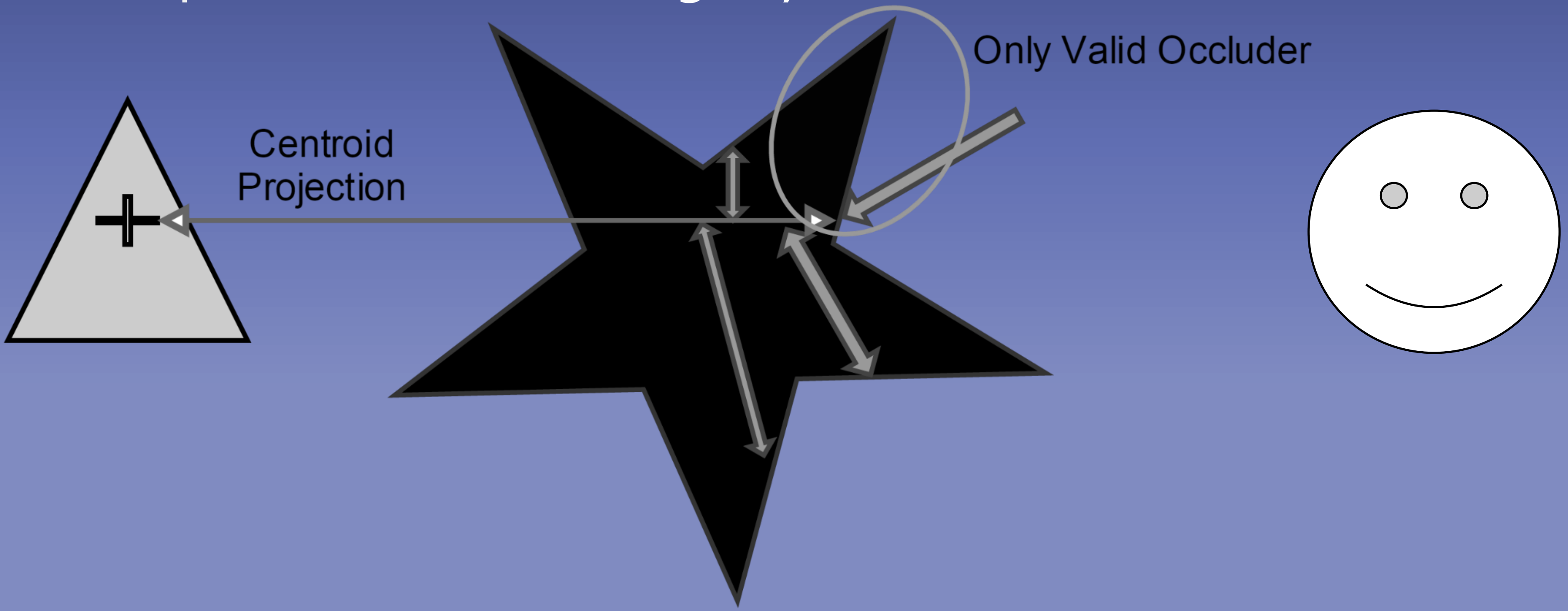
Triangle B → Edge 3 → Edge 4

Triangle C → Edge 5 → Edge 1 → Edge 2 → Edge3

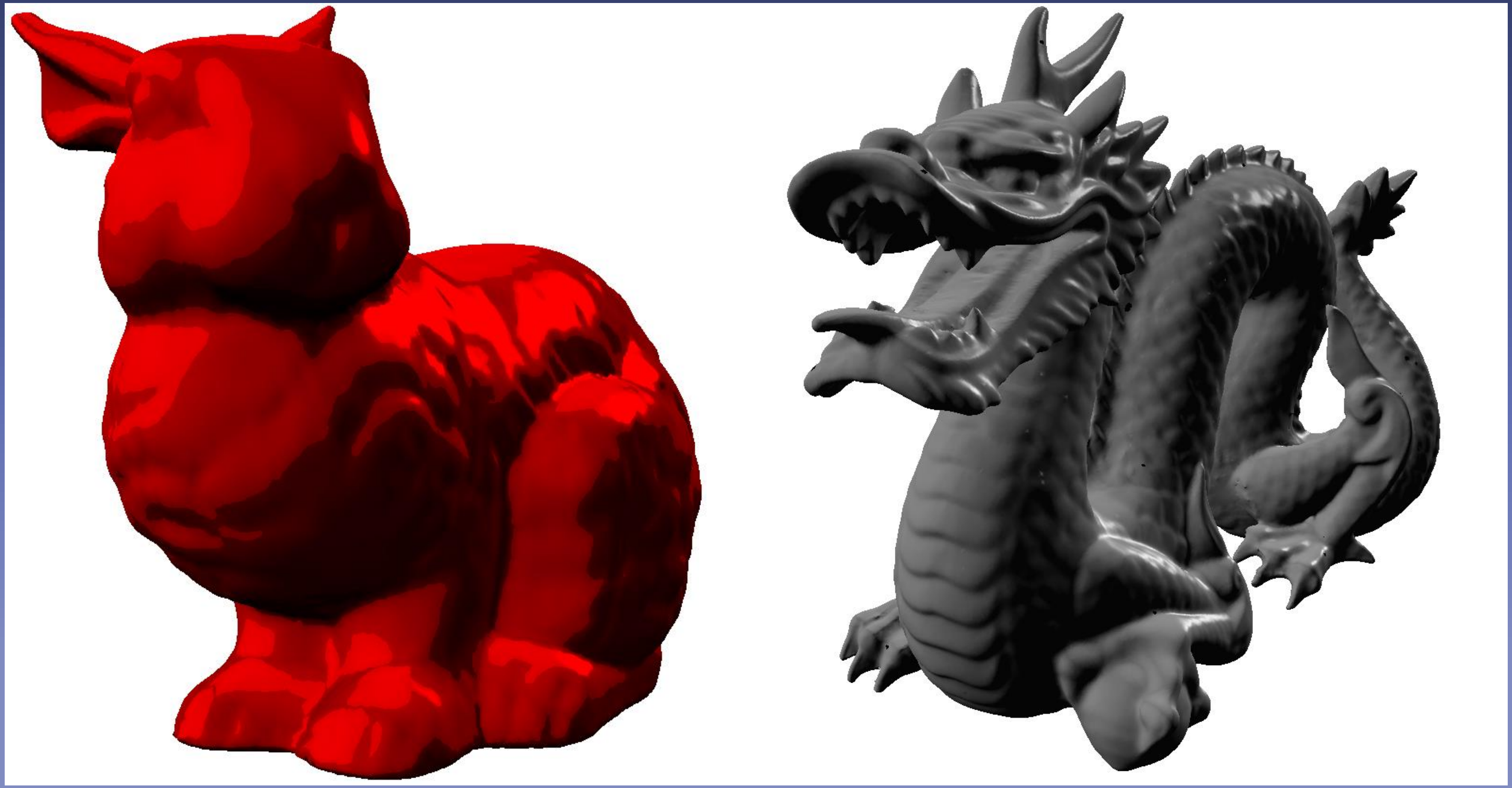


Occlusion

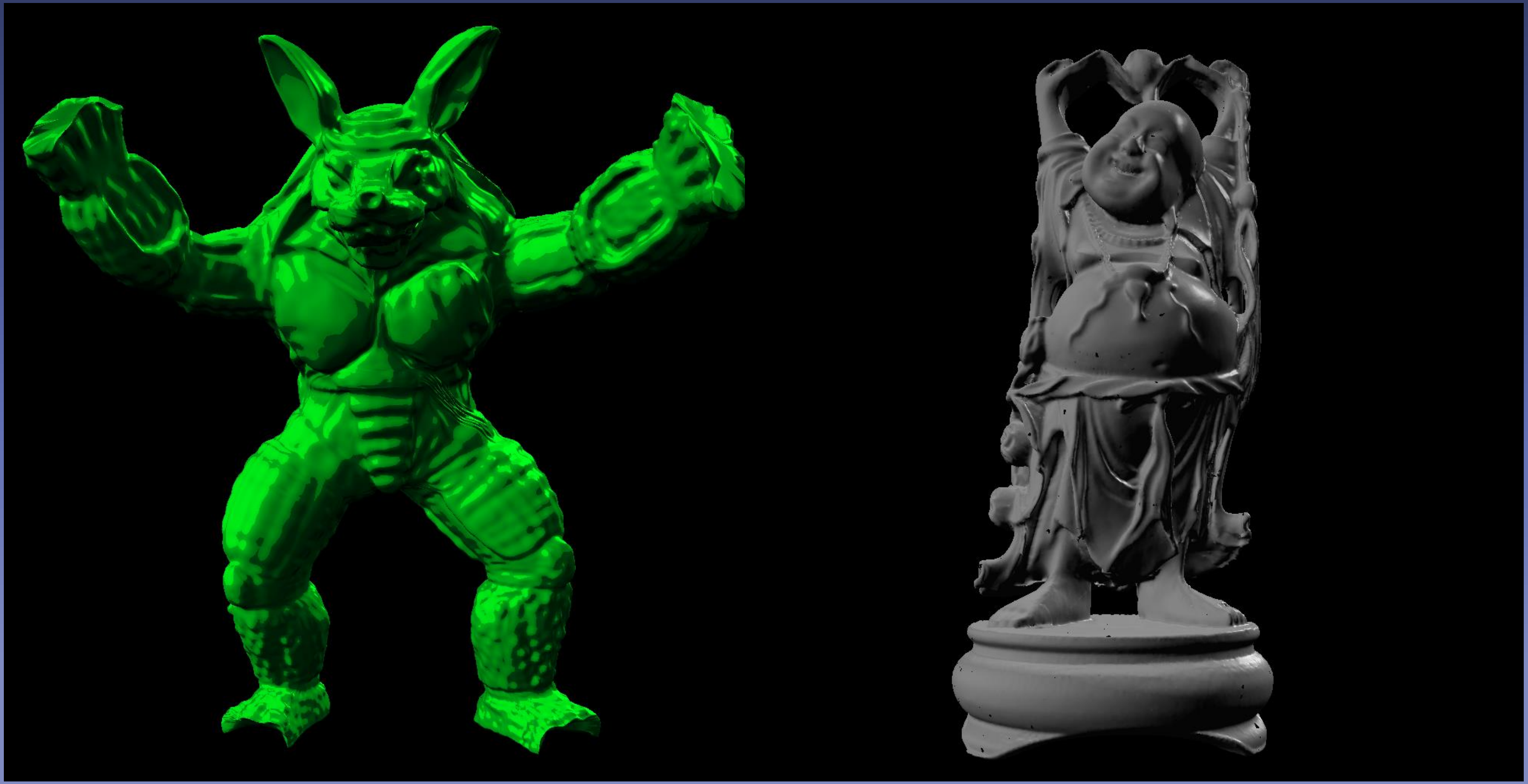
- Triangles now fully occluded *or* fully visible
- One point occluded? Every point occluded.
- Centroid provides least ambiguity



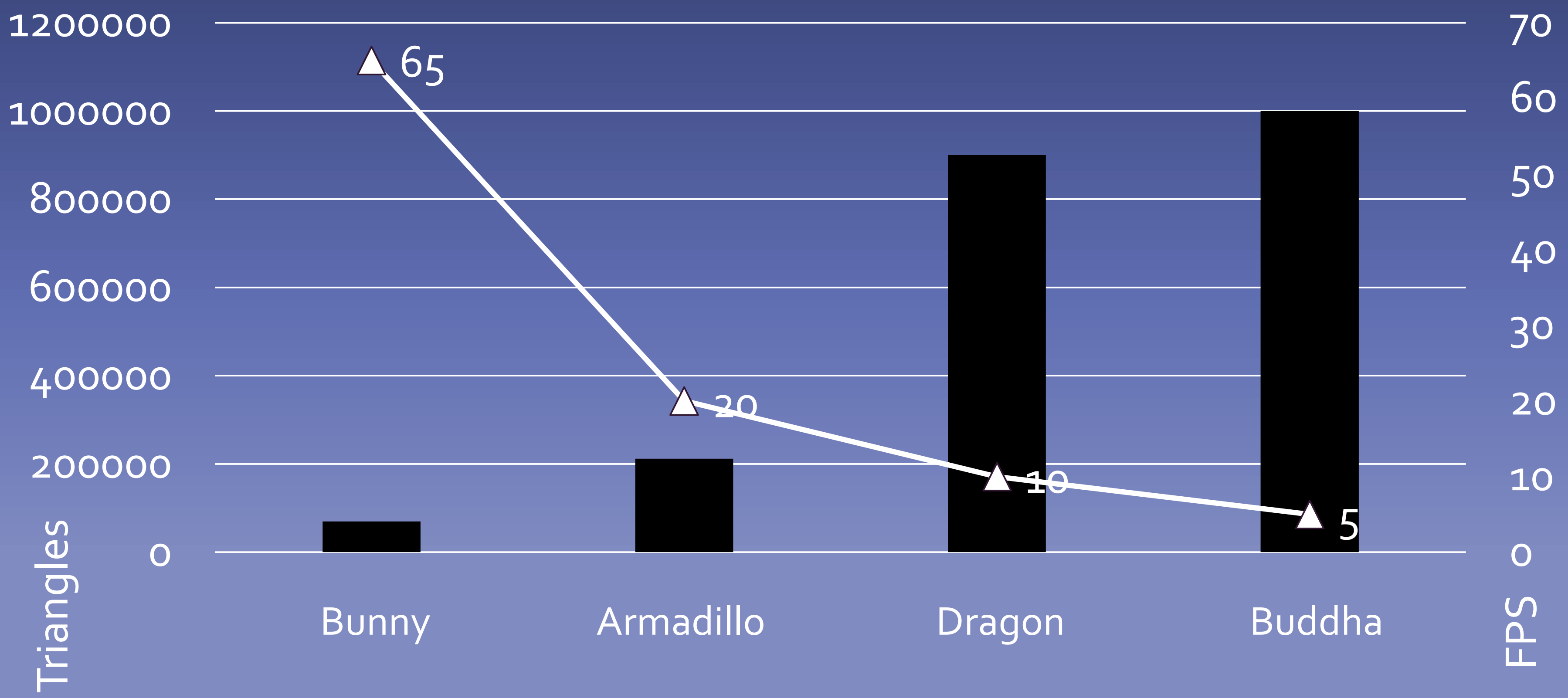
Rasterized Planar Maps



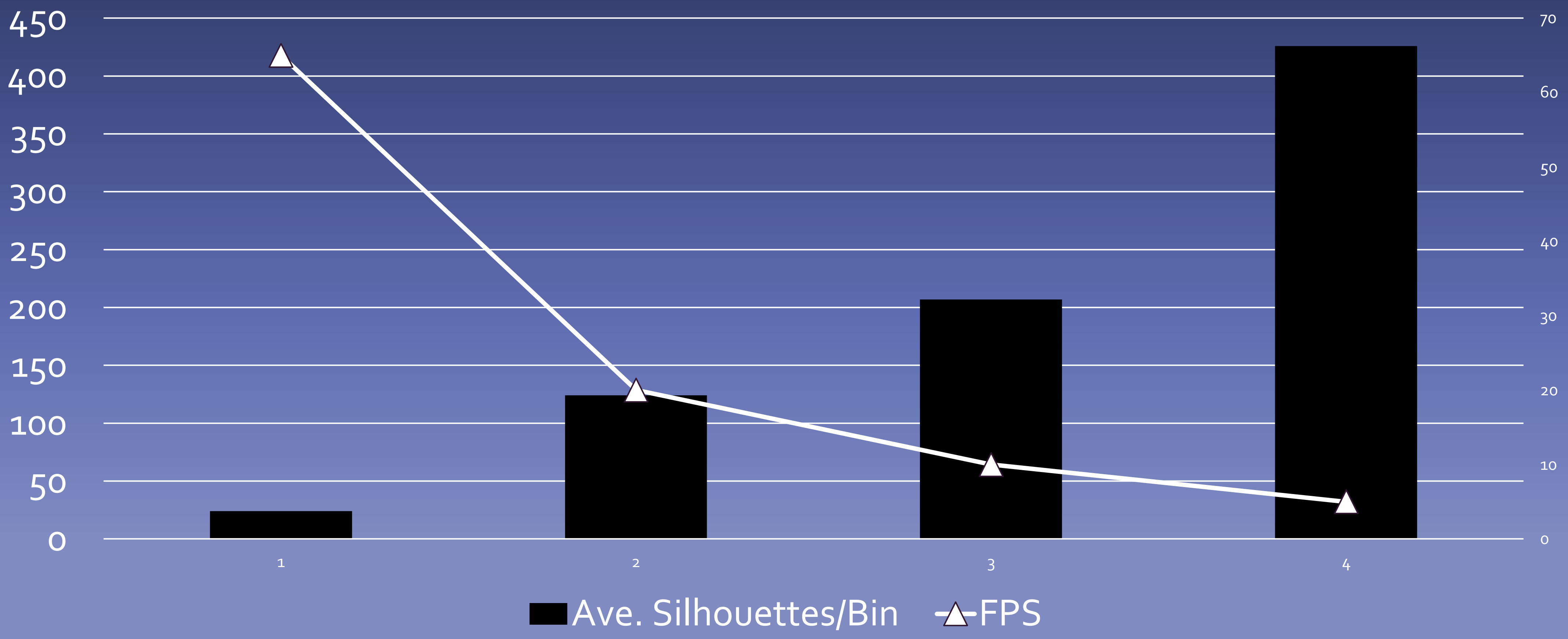
Rasterized Planar Maps



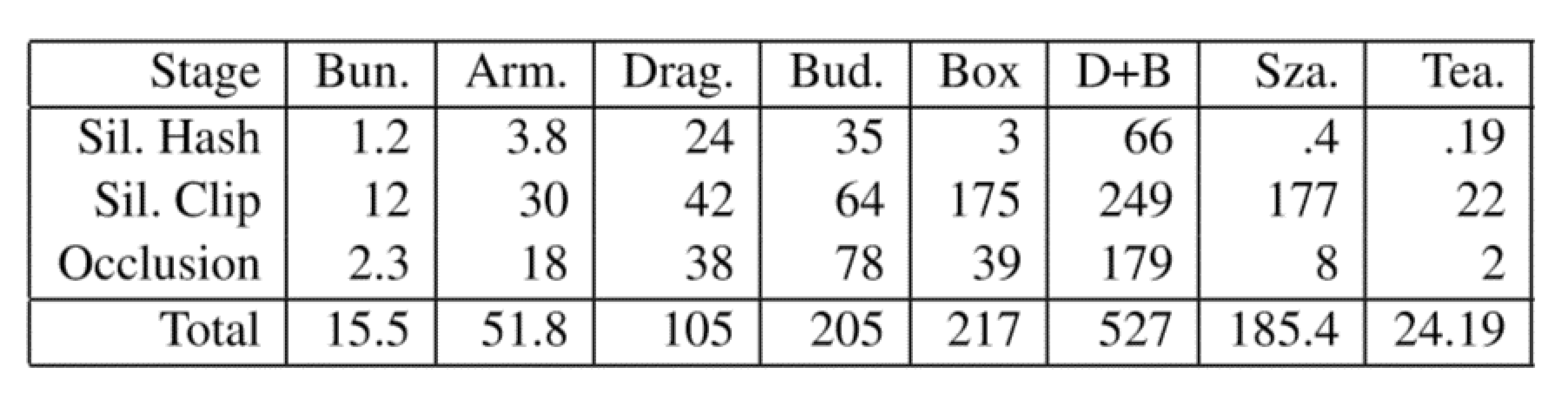
Perf Scaling: Triangle Count



Perf Scaling: Silhouettes per Bin

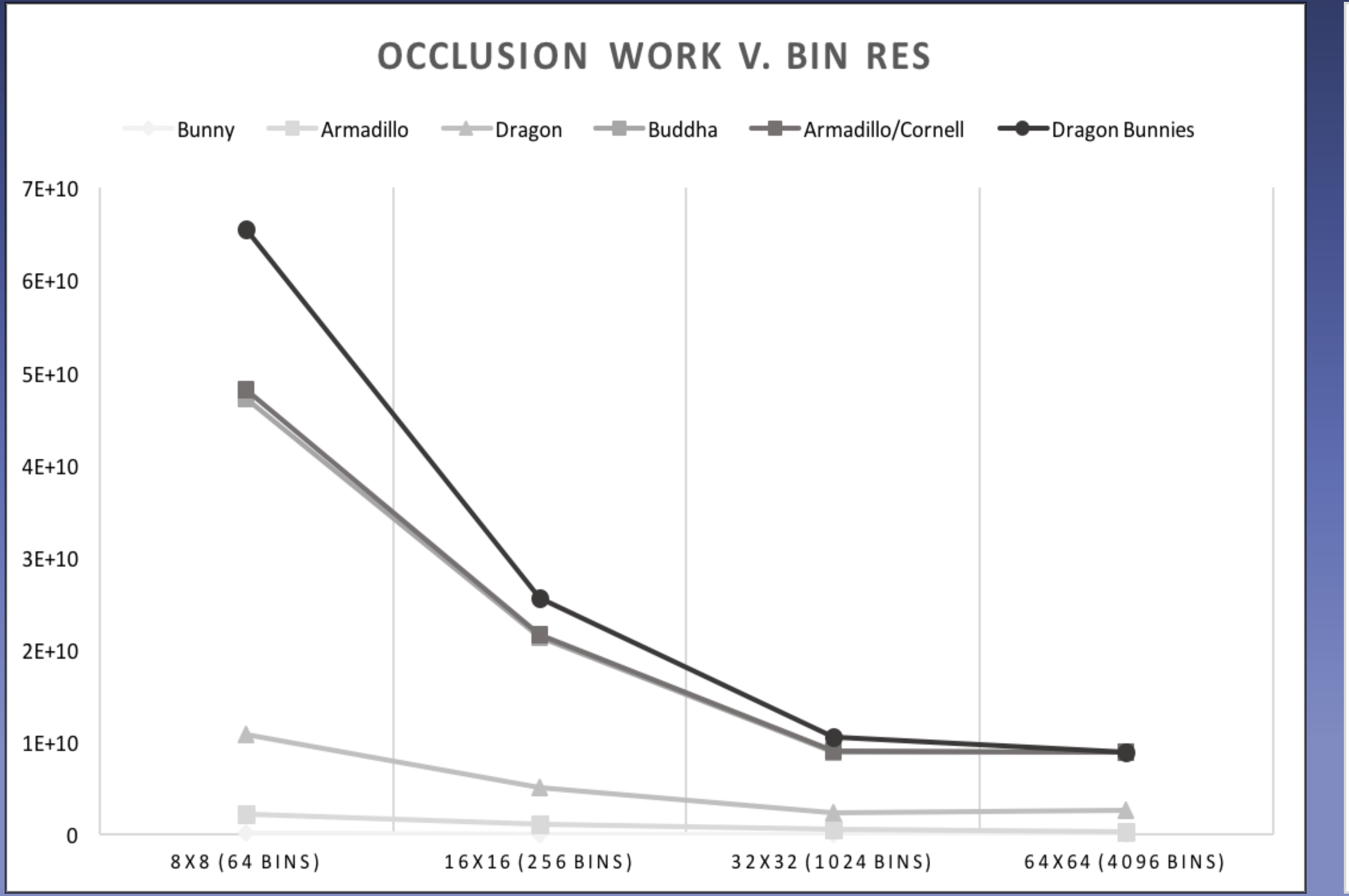


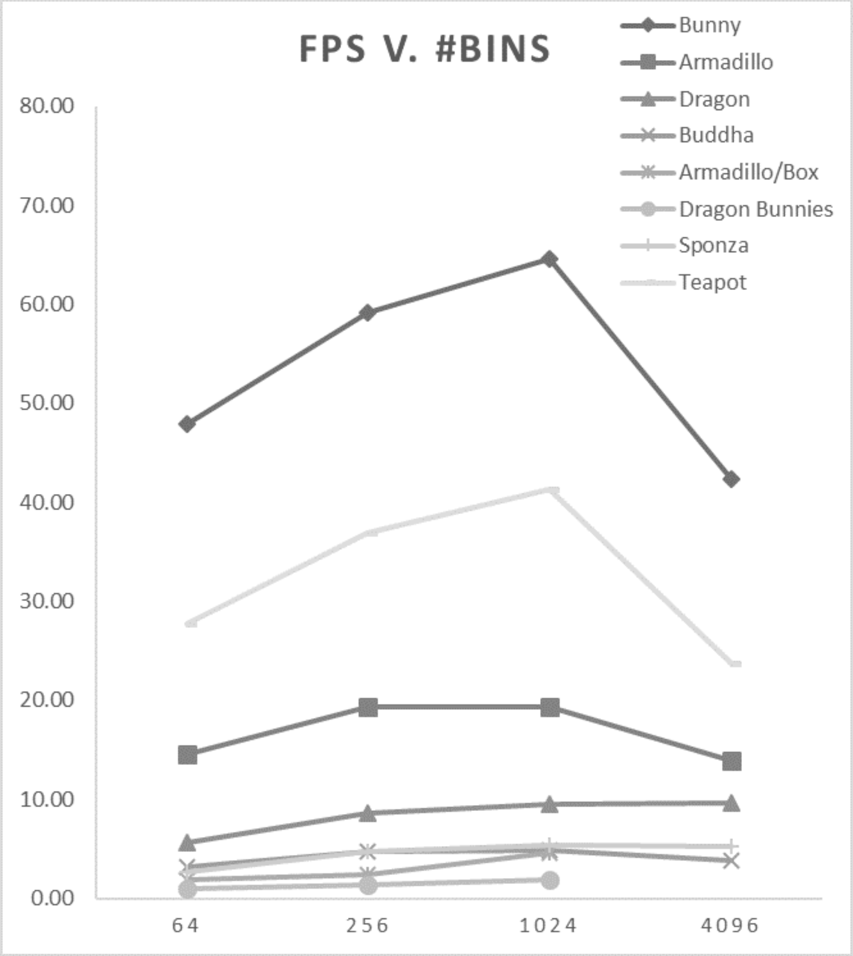
Phase Breakdown



Stage	Bun.	Arm.	Drag.	Bud.	Box	D+B	Sza.	Tea.
Sil. Hash	1.2	3.8	24	35	3	66	.4	.19
Sil. Clip	12	30	42	64	175	249	177	22
Occlusion	2.3	18	38	78	39	179	8	2
Total	15.5	51.8	105	205	217	527	185.4	24.19

Motivation for Re-binning

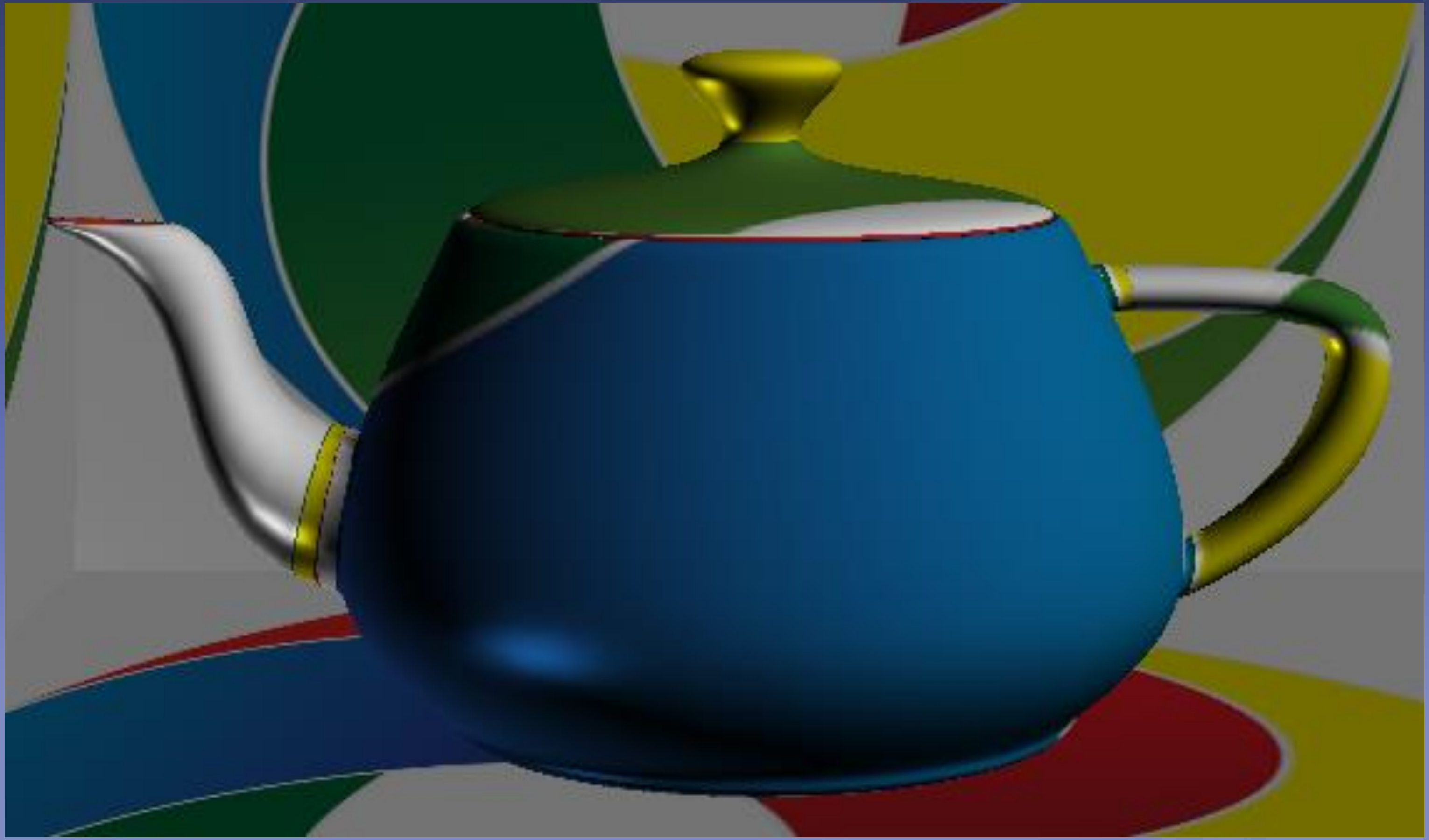




Pathological Clipping Case

- Many silhouettes one poly
- Adjacency list very long
- After one clip many edges invalid
- Need to re check trivial reject
- Need to re check in parallel

Attribute Interpolation



Summary

- We can generate planar maps fast
- ~5X previous approaches
- We've evaluated binning and scaling considerations
- 1024 (32x32) bins performs best in most cases
- Highlighted pathological issues needing mitigation
- Robert's based approach performs reasonably well

Future Work

- Precision issues need filtering
- Uniform binning precluding teapot in stadium
- Adaptive binning i.e. quad/kd trees are attractive
- In progress work on cloud gaming
- Theorizing about adaptive sampling and shading
- So called “Free” effects need validation and POC

References

- [AWJ13] AUZINGER T., WIMMER M., JESCHKE S.: *Analytic visibility on the gpu*. Computer Graphics Forum (Proc. Eurographics) 32, 2 (May 2013), 409–418.
- [Rob63] ROBERTS L.: *Machine perception of three-dimensional solids*. Tech. Rep. TR 315, Lincoln Laboratory, MIT, 1963.
- [BF09] BERNSTEIN G., FUSSELL D.: *Fast, exact, linear booleans*. In Proceedings of the Symposium on Geometry Processing (Aire-la-Ville, Switzerland, Switzerland, 2009), SGP '09, Eurographics Association, pp. 1269–1278.
- [McKenna87] Michael McKenna. 1987. *Worst-case optimal hidden-surface removal*. *ACM Trans. Graph.* 6, 1 (January 1987), 19-28.
- [Dév11] F. Dévai. 2011. *An optimal hidden-surface algorithm and its parallelization*. In Proceedings of the 2011 international conference on Computational science and its applications - Volume Part III(ICCSA'11), Beniamino Murgante, Osvaldo Gervasi, Andrés Iglesias, David Taniar, and Bernady O. Apduhan (Eds.), Vol. Part III. Springer-Verlag, Berlin, Heidelberg, 17-29.
- [Dév86] F Devai, *Quadratic bounds for hidden line elimination*, Proceedings of the second annual symposium on Computational geometry, p.269-275, June 02-04, 1986, Yorktown Heights, New York, USA
- [Jdoss] Joshua Doss. *Sponsored Feature: Inking the Cube: Edge Detection with Direct3D 10.* (2008 Aug 27) Retrieved 6/9/16 http://www.gamasutra.com/view/feature/130067/sponsored_feature_inking_the_.php
- [Mun] cs.mun.cs Retrieved 6/13/16 http://www.cs.mun.ca/~omeruvia/philosophy/WireframeBunny.html
- [Google] Retrieved 4/21/2015 https://www.google.com/imghp

Contacts

- *aiellis2* at *illinois* dot *edu*
- *warren* dot *hunt* at *gmail* dot *com*
- *jch* at *illinois* dot *edu*
- *http://graphics.cs.illinois.edu/*