Fast and Precise: GPU Techniques for 2D

Sean James
Carnegie Mellon University
AutoCAD Product Line, Autodesk Inc.

Ravi Krishnaswamy
Sr. Software Architect
AutoCAD Product Line, Autodesk Inc.

AUTODESK

Agenda

- Goals
- Analytical Curves
- Line styles (linetypes)
- Draw order
- Performance
- Conclusion

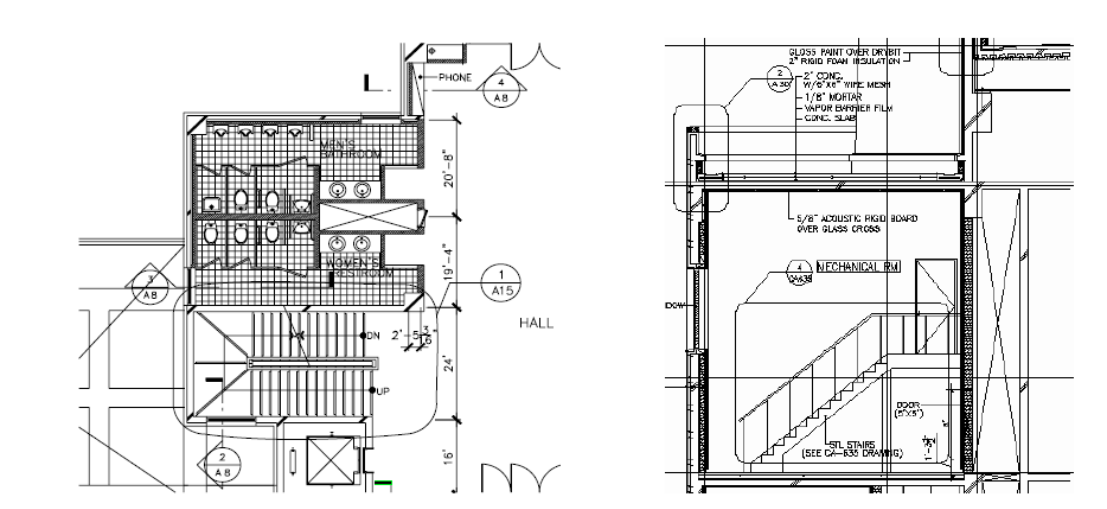
AUTODESK

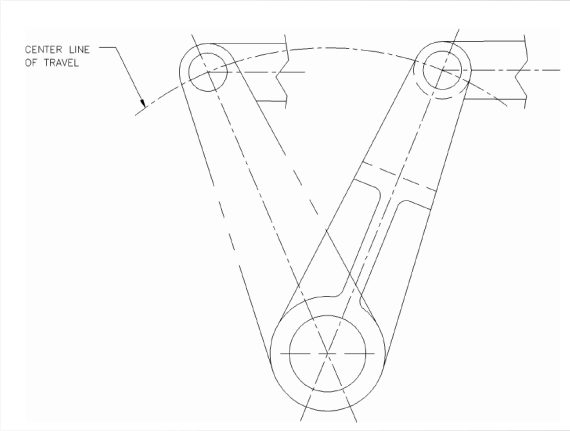
Background: Project Goal

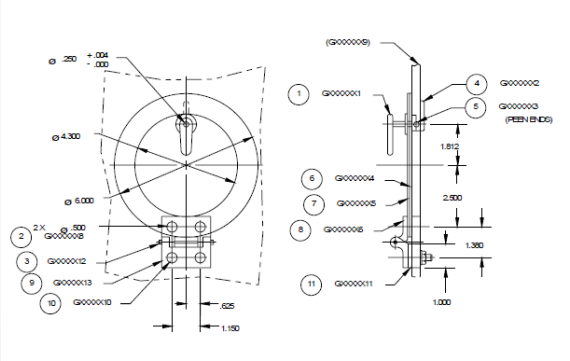
Improve visual quality and performance of 2D curve rendering, particularly for AutoCAD

Move from performance **OR** quality to performance **AND** quality through better use of the GPU

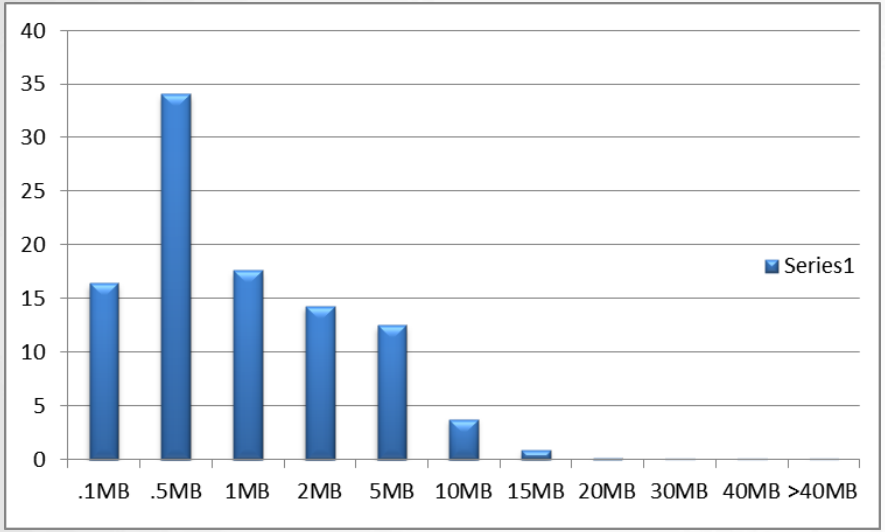
2D: Engineering Documentation

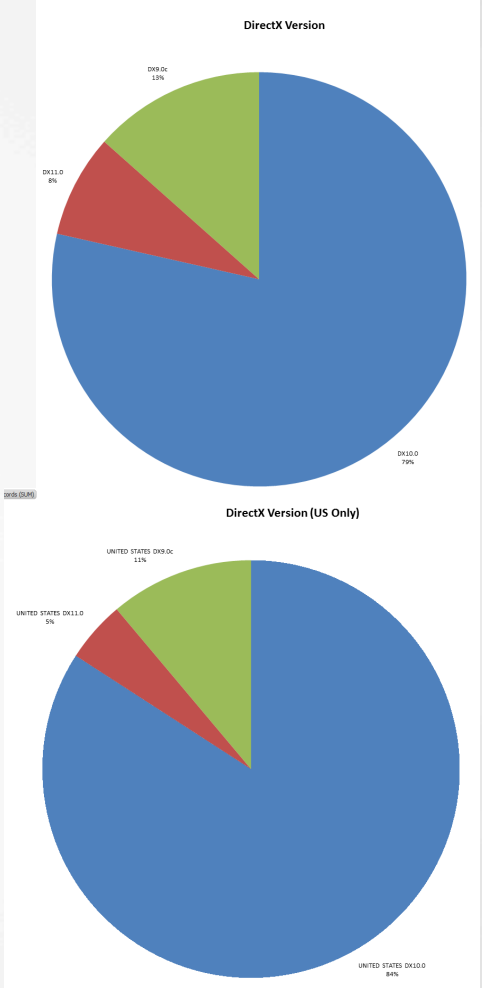






Some Stats





Background: Product Usage

- Many 2D users
- Many DX9 and DX10 feature level users
- Drawing data size typically .5-1MB
- Number of 2D entities in the 1000s to 100000s

Considerations

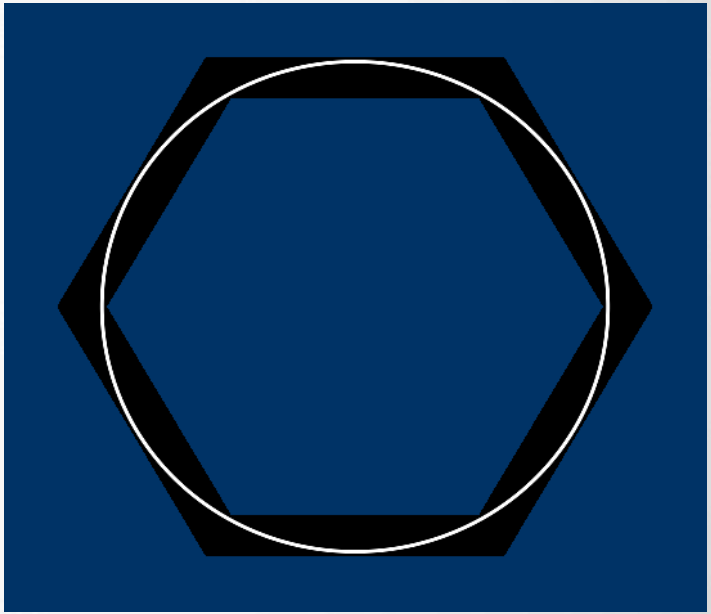
- Productivity: fast open, zoom, pan, etc.
- Standards: ISO/ANSI patterns. Industry and Company standards and conventions
- Visual quality: smooth curves

Analytical Curves

AUTODESK

Analytical Curves: High level technique

- Typically curves are tessellated line segments
- Our solution:
  - Render “envelope” geometry around curve
  - Solve curve equation in pixel shader



Analytical Curves: Benefits

- DX9, OpenGL 3.2 compatibility: no geometry shader tessellation
- Smooth curves and antialiasing at pixel level
- Quality independent of amount of geometry: lower memory/PCIe bandwidth requirement
- High-quality option for users investing in better GPUs

Analytical Curves: Instancing

- Drawing thousands of curves
- GPU instancing:
  - One copy of base geometry
  - GPU duplicates geometry automatically
  - Vertex shader transforms geometry for each instance
- Memory/bandwidth savings and performance improvement over AutoCAD implementation
- Complicates envelope generation and vertex shading

Analytical Curves: Ellipses

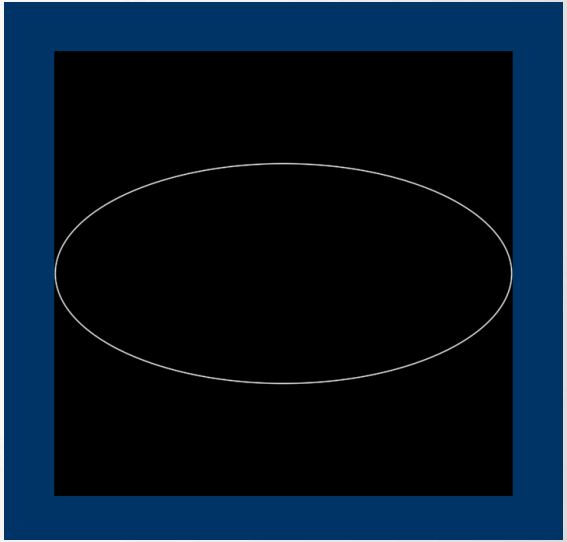
- Problem: No closed form for distance from pixel to curve
- Correct solution: find θ such that error is minimized. Would require iterative solver in pixel shader.

$$\frac{d}{d\theta}\sqrt{(x - a\cos\theta)^2 + (y - b\sin\theta)^2} = 0$$

- Approximation: to find distance *d(p)* from point *p* to curve given implicit ellipse function *f*. Can be implemented with ddx/ddy shader instructions. See Blinn/Loop 05 for details.

$$f(\vec{p} = (x, y)) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1$$

$$d(\vec{p}) \approx \frac{f(\vec{p})}{\|\nabla f(\vec{p})\|}$$



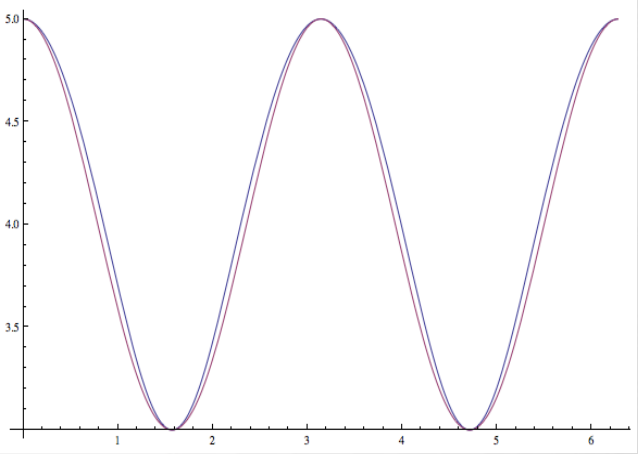
Analytical Curves: Ellipses (cont.)

- Problem: No closed form for arc length. Arc length needed for linetype parameterization (later).
- Correct solution: would need to find integral in pixel shader:

$$s(\theta) = \int_{\theta_0}^{\theta} \sqrt{a^2 \sin^2 \theta + b^2 \cos^2 \theta}\, d\theta$$

- Approximation: note that the term being integrated is very similar to the following, which has a simple integral.

$$s(\theta) \approx \frac{b-a}{2} \cos 2\theta + a + \frac{b-a}{2}$$

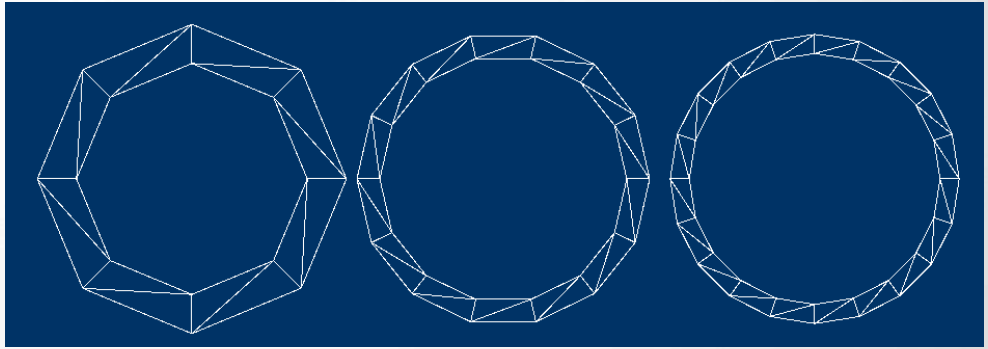


Analytical Curves: Ellipses (cont.)

- Problem: Envelope generation for ellipses is non-trivial (tangent, radii depend on θ)
- Approximation: use envelope for a circle and scale to ellipse radii in vertex shader.
- Excess pixel coverage but simple, fast, works with instancing.

Analytical Curves: Performance

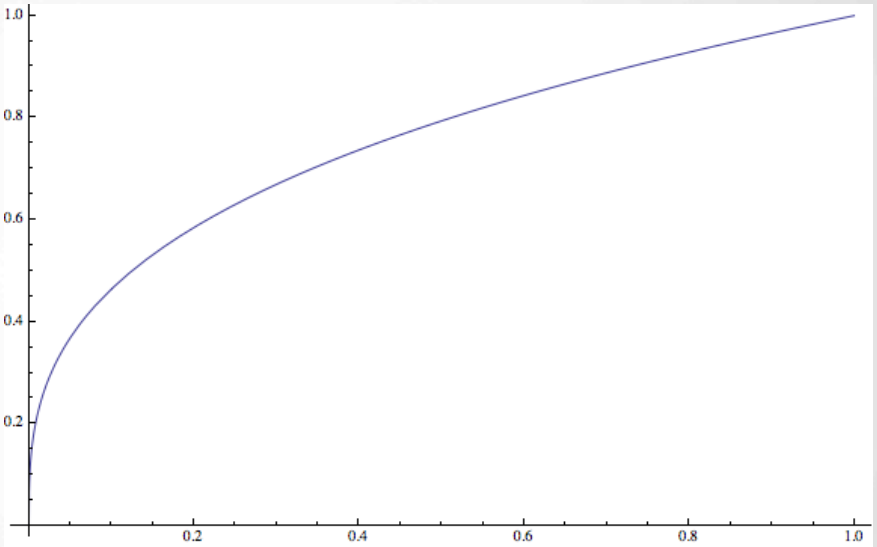
- Slow, expensive pixel shading
- Culling: avoid drawing geometry off screen
- Level of detail: balance between envelope geometry complexity and screen size
- Could use an acceleration structure (ex: quad-tree) or geometry shader. Currently CPU only.



AUTODESK

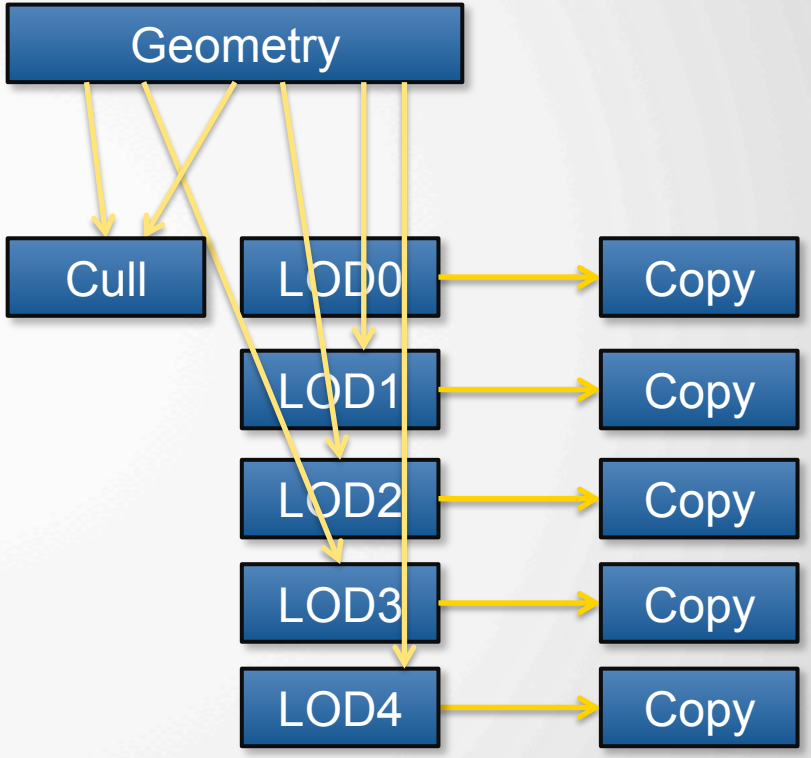
Culling/LOD

- Compute bounding box for each instance, and either cull or choose a level of detail and copy into LOD instance buffer
- Copy instance buffers to GPU
- LOD curve based on fraction of screen covered. Avoids shading too many pixels for small objects; grows more slowly for large objects.



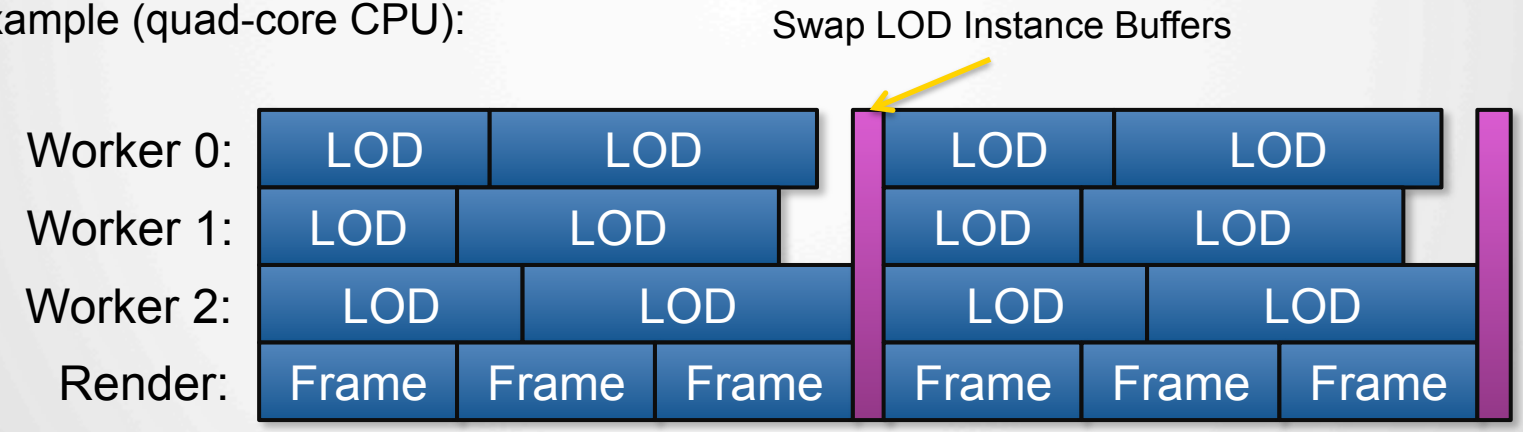
Culling/LOD

- Compute bounding box for each instance, and either cull or choose a level of detail
- Copy to GPU



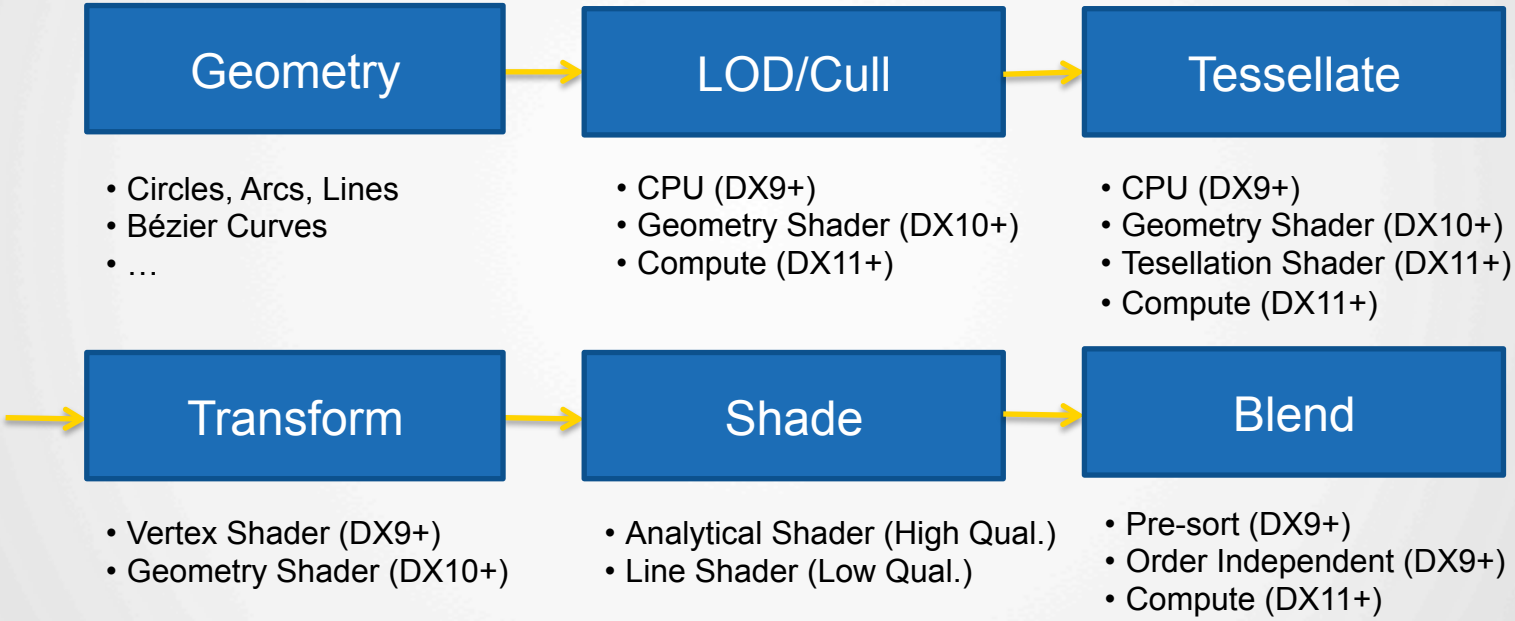
Culling/LOD: Multiple Threads

- Decouples render frame rate and slow LOD/culling/copy
- Worker threads assigned small batches of geometry for culling/LOD to balance work distribution
- Double-buffered level of detail buffers to avoid GPU stalling
- Example (quad-core CPU):



Render Pipeline

- Possible pipeline permutations:



AUTODESK

Linetypes

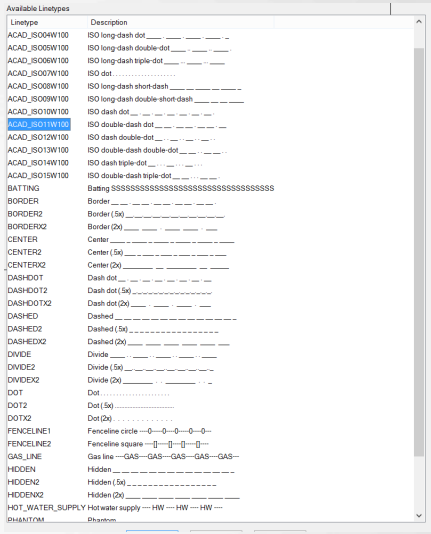
AUTODESK

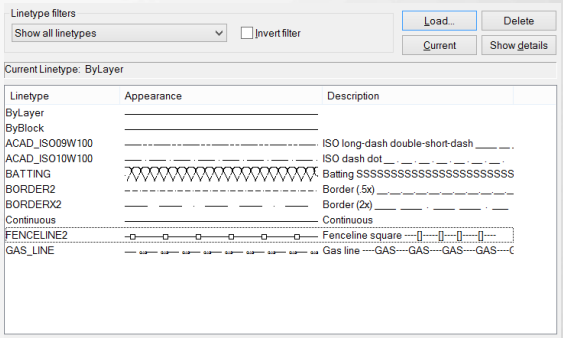
Linetypes: Cases

- 1D and complex linetypes
- Standard and custom linetypes
- Polylines and composite curves
- Linetype scale and zoom invariance

Linetypes: Basic and Complex

- Lines are weighted (2H and HB pencils)
- Dash lengths are paper and scale specific
- Complex linetypes contain symbols and text

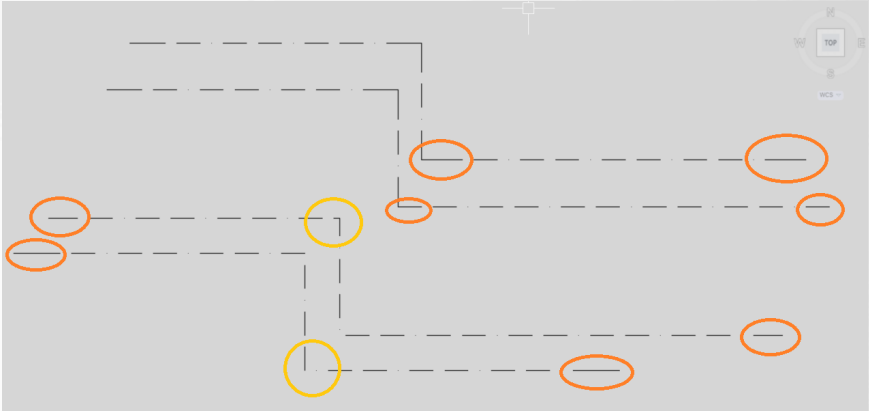


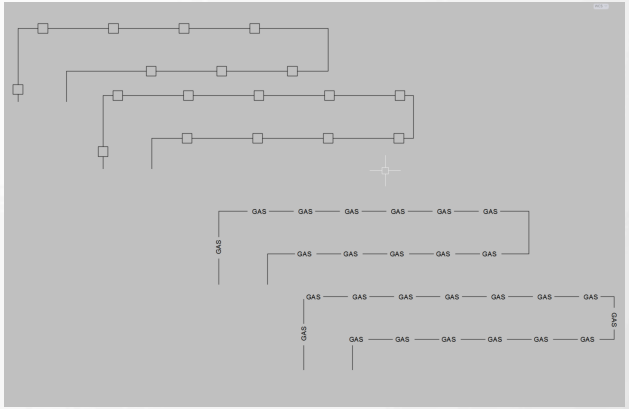


AUTODESK

Linetypes: pattern adjustments

- End conditions shift the pattern or clamp
- Complex linetypes adjust symbol alignment

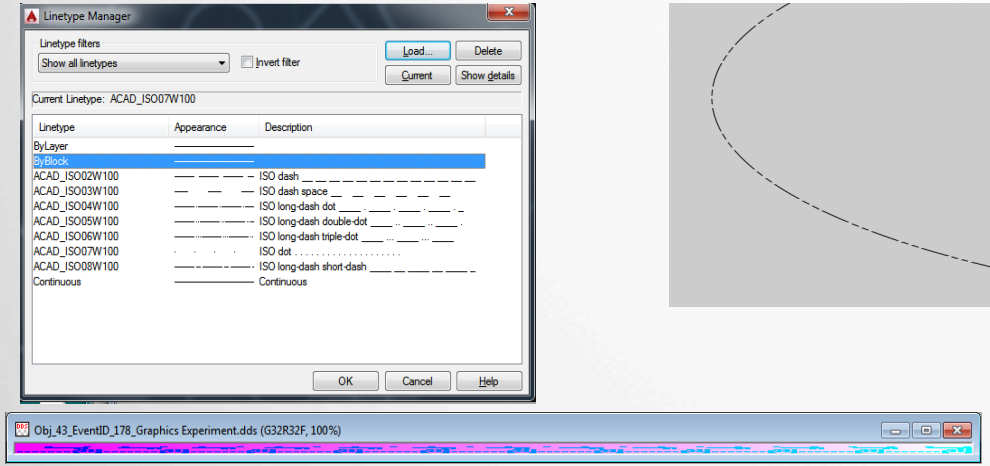


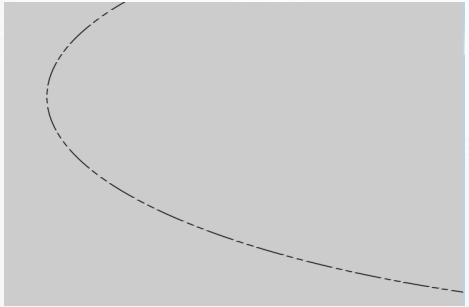


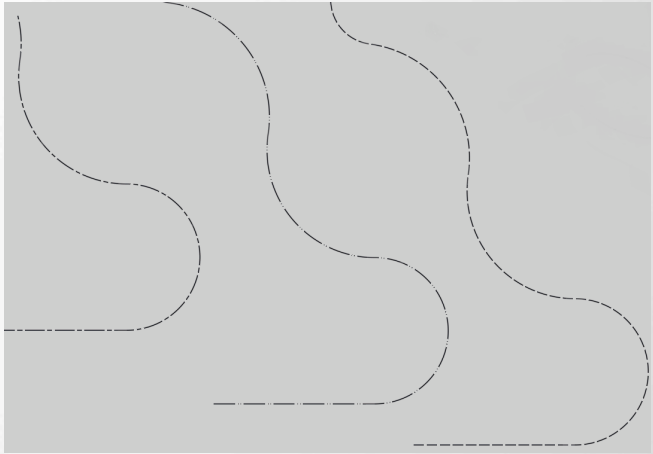
AUTODESK

Linetypes: Approach

- Texture: 2D as array of 1D textures (R,G)
- Use (cumulative) length as parameterization
- Handle end conditions via Phase Shift/Clamping



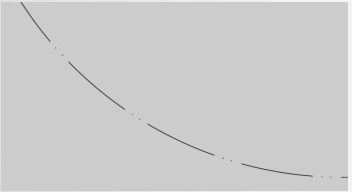


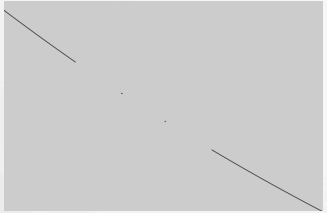


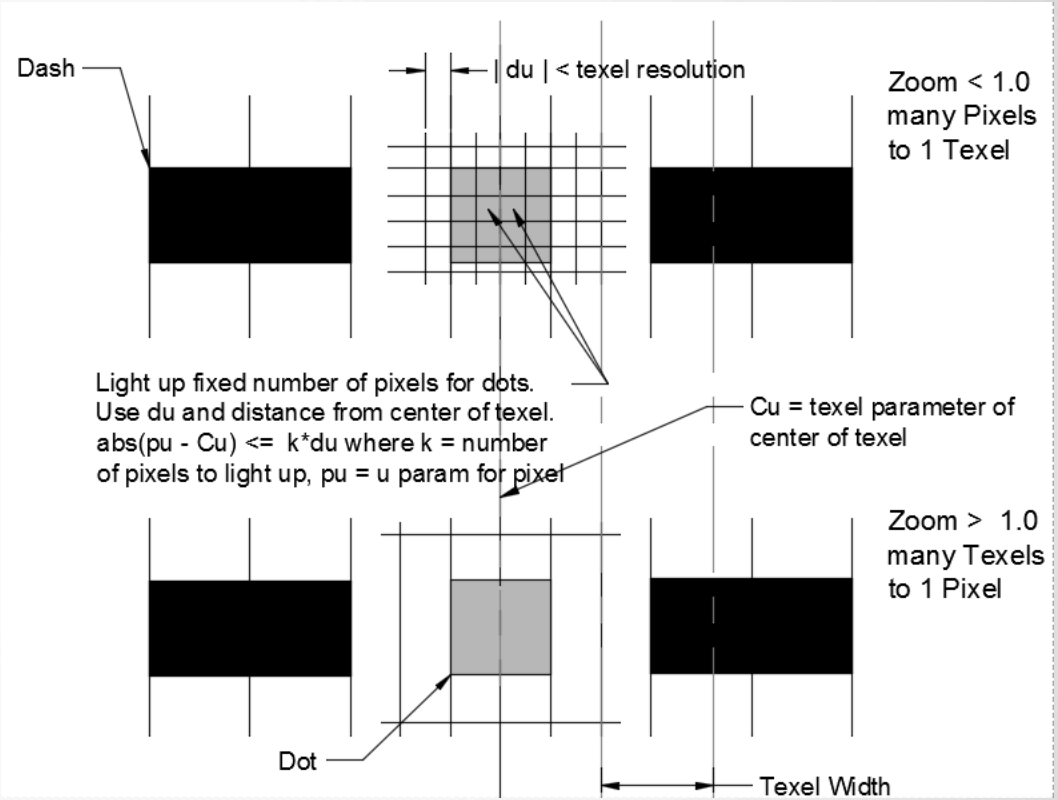
AUTODESK

Linetypes: Dots

- Texel Center u: ‘G’
- Dot or Dash : ‘R’
- Test pixel u distance from Cu in the shader







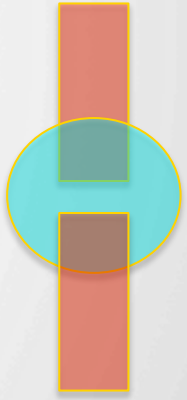
AUTODESK

Demo

AUTODESK

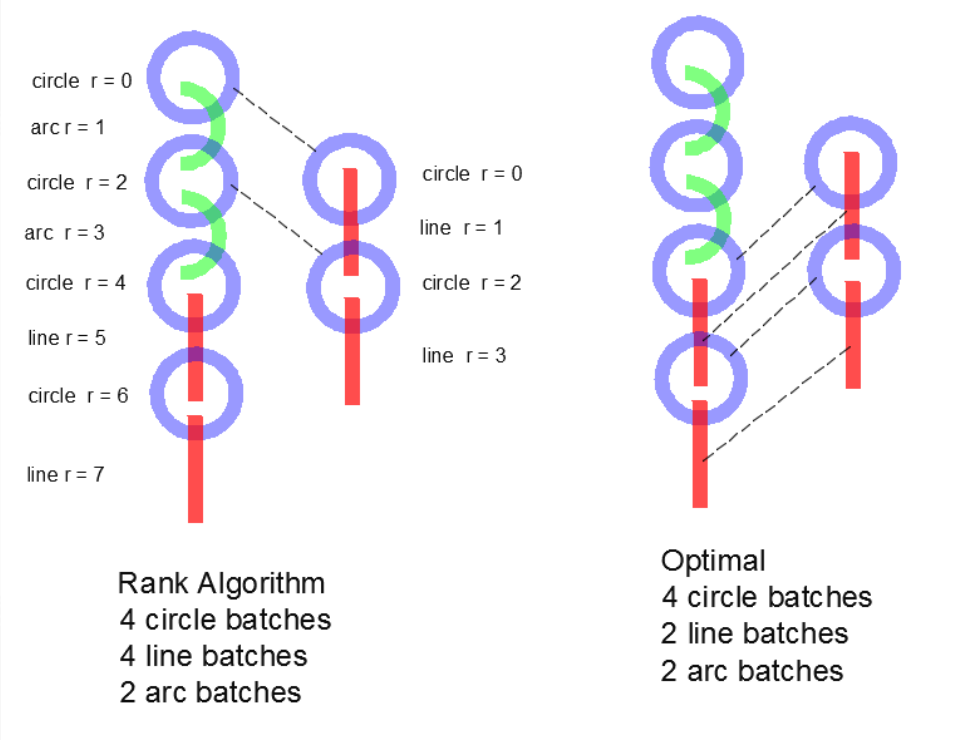
Draw Order and Transparency

- Curves are transparent: need to draw back to front for correctness
- Instancing: cannot interleave drawing different geometry types
- CPU Based: view as a graph problem: Rank Approach
  - Consider the DAG based on overlap and order
  - Topological sort
- GPU Based: No broad support.
  - Consideration of OIT Dx11 SM 5.5
- Static scenarios should consider preprocessing v/s every frame



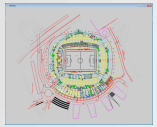
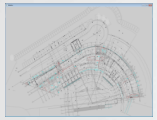
Rank Algorithm:

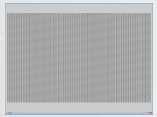
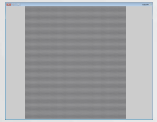
- Topological Sort
- Node numbering based on depth
- Batch by type and node numbering
- Bounded by N*OPT where N = number of distinct batch types, OPT = optimal batch count



AUTODESK

Samples and Performance : Pan full scene

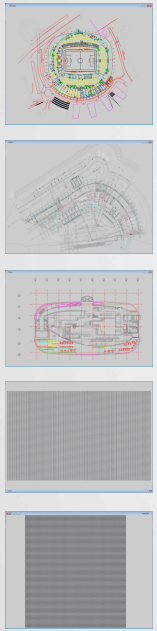



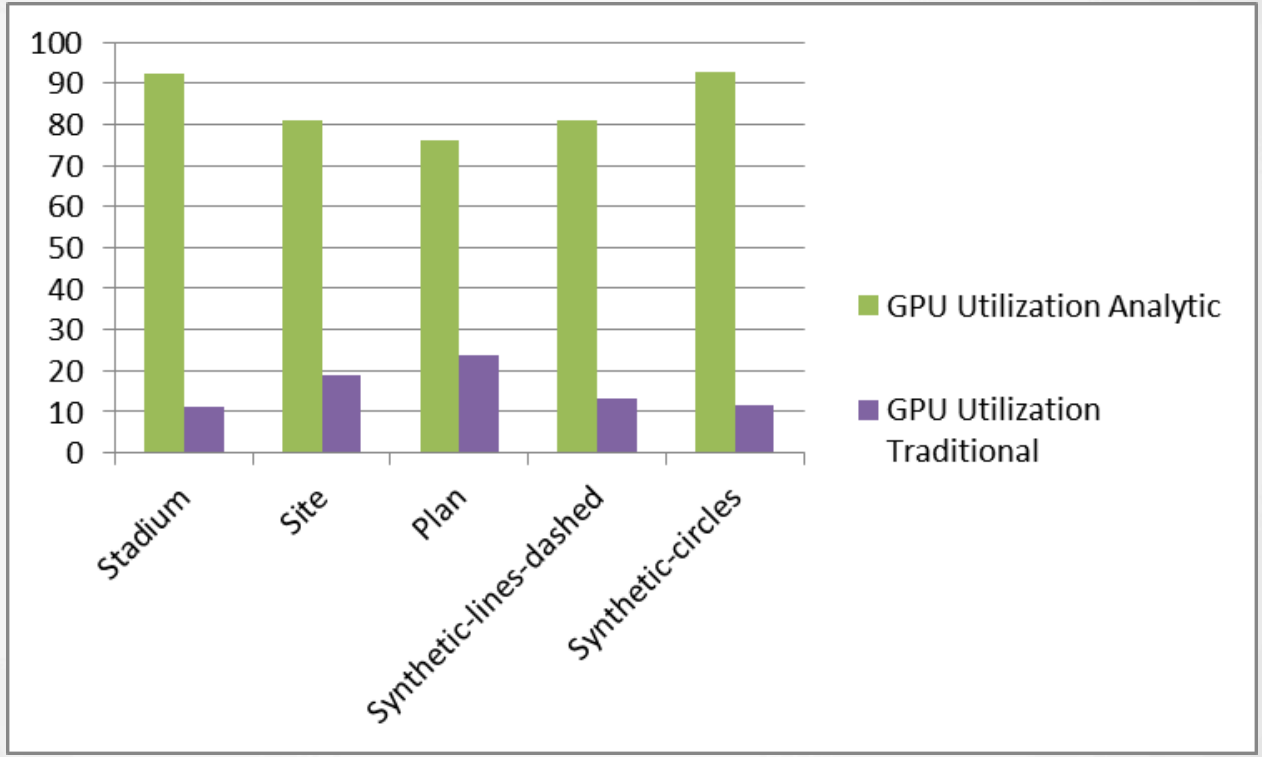



DataSet	Application	#Lines	#Elliptical Arcs	#Circles	# CircleArcs	GPU%	Average ms/ Frame
Stadium	Traditional	301378	529	1265	8350	11.2	65
Stadium	Analytic	301378	529	1265	8350	92.5	32
Site	Traditional	140755	533	1956	7699	18.9	45
Site	Analytic	140755	533	1956	7699	81	19
Plan	Traditional	104786	540	144	1866	23.6	25
Plan	Analytic	104786	540	144		76.2	11
Synthetic-lines-dashed	Traditional	125000				13.4	68
Synthetic-lines-dashed	Analytic	125000				81.1	4
Synthetic-circles	Traditional			100000		11.4	35
Synthetic-circles	Analytic			100000		92.7	9

AUTODESK

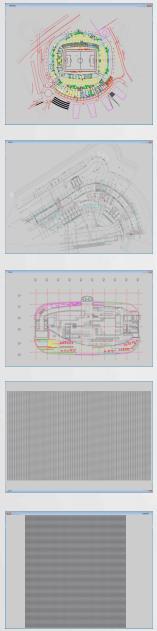
GPU Utilization

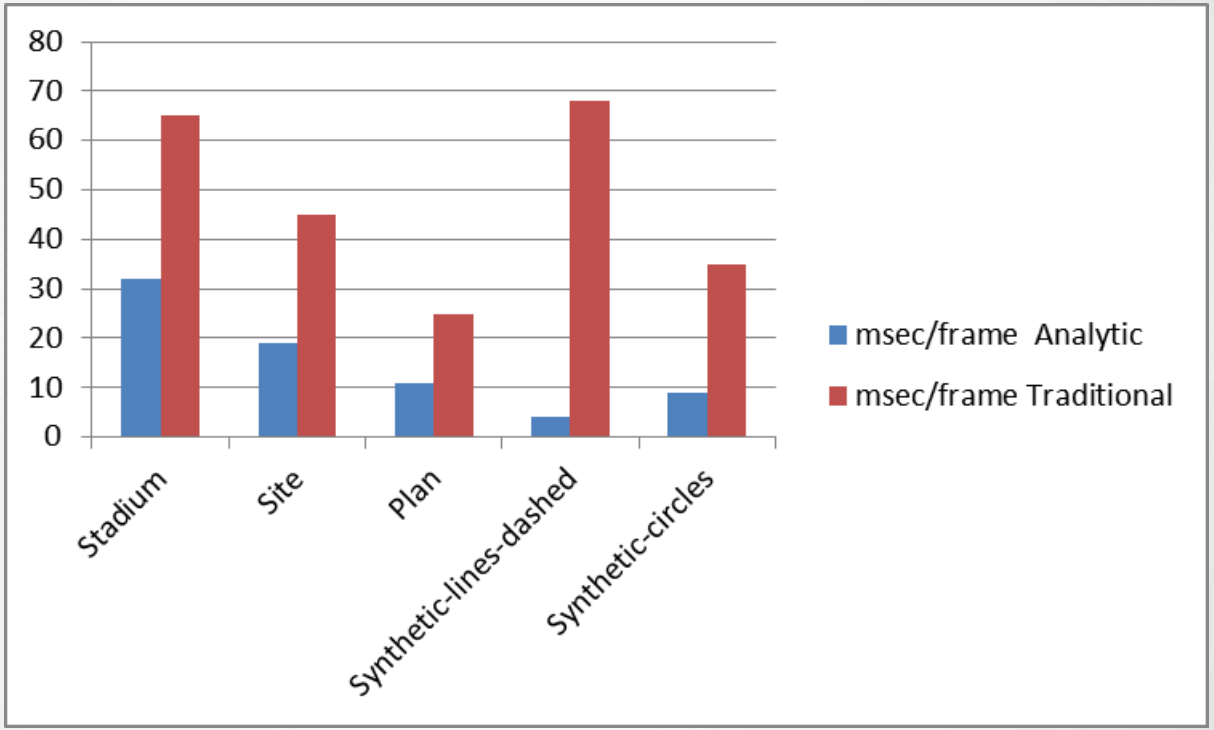




AUTODESK

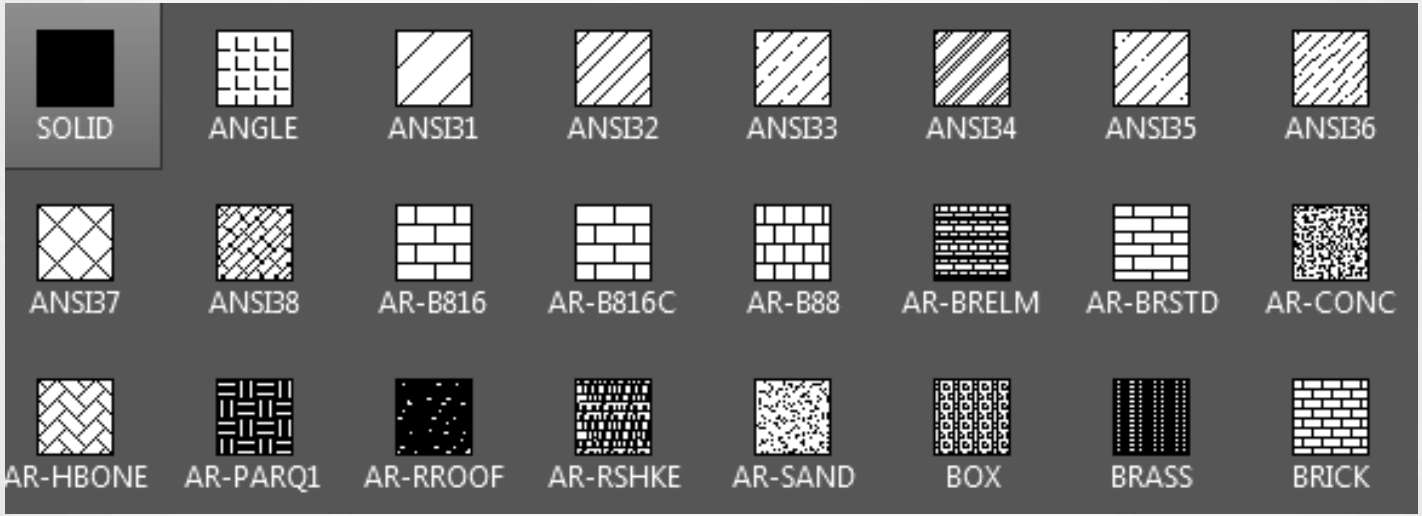
Milliseconds per frame





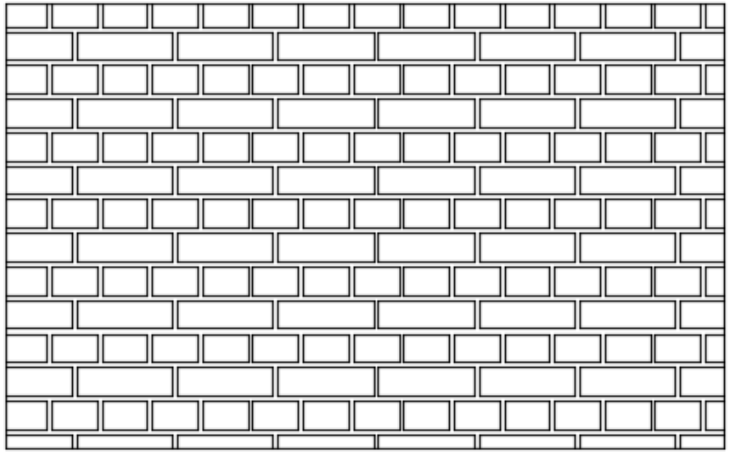
AUTODESK

Extension of linetypes to hatch pattern



AUTODESK

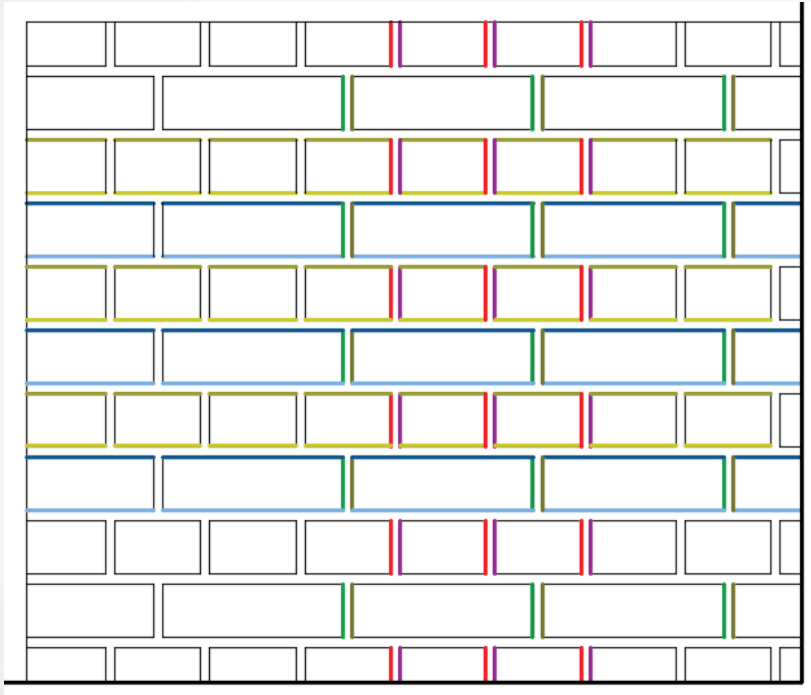
Hatch Pattern Definition



```
*AR-B88, 8x8 Block elevation stretcher bond
0,        0,0,       0,8
90,       0,0,       8,4,                          8,-8
*AR-BRELM, Standard brick elevation english bond with mortar joints
0,       0,0,        0,5.334,                      7.625,-.375
0,       0,2.25,     0,5.334,                      7.625,-.375
0,       2,2.667,    0,5.334,                      3.625,-.375
0,       2,4.917,    0,5.334,                      3.625,-.375
90,      0,0,        0,8,                          2.25,-3.084
90,     -0.375,0,    0,8,                          2.25,-3.084
90,      2,2.667,    0,4,                          2.25,-3.084
90,      1.625,2.667,  0,4,                        2.25,-3.084
*AR-BRSTD, Standard brick elevation stretcher bond
0,        0,0,       0,2.667
90,       0,0,       2.667,4,                      2.667,-2.667
*AR-CONC, Random dot and stone pattern
```

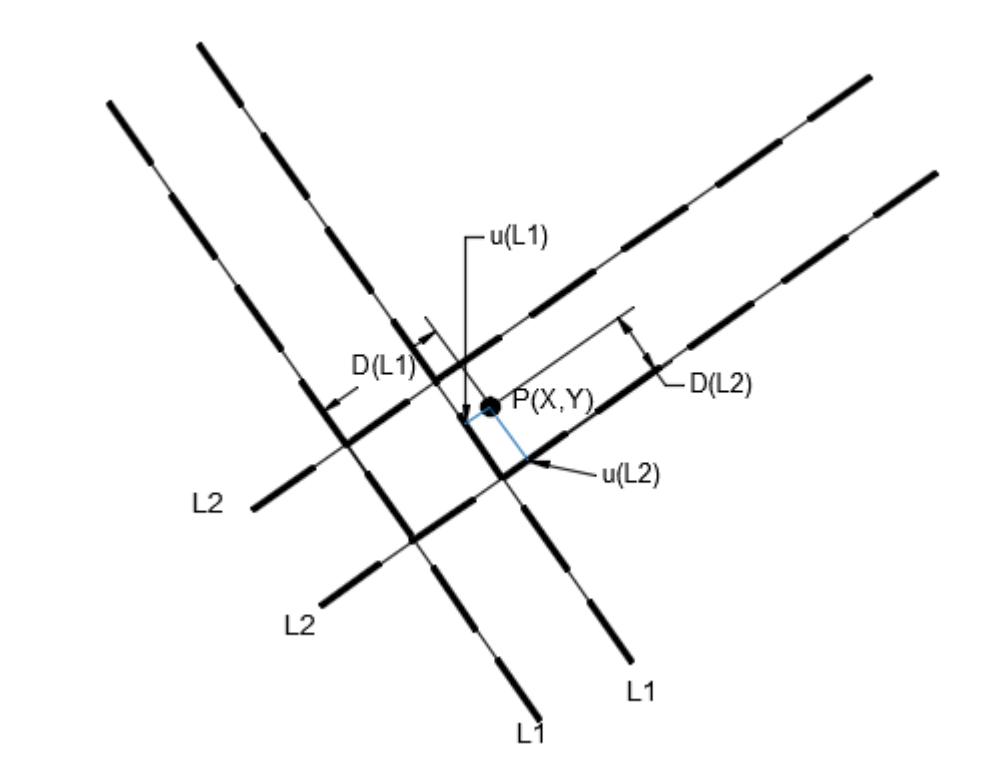

AUTODESK

Generated as lines with linetypes



AUTODESK

Pixel Shader: Identify closest line and linetype



AUTODESK

Conclusions

- High quality with performance and capacity gains
- Non traditional behavior for stylization
- Can be implemented with lower shader model requirements

Questions?

AUTODESK