RenderMan Nuts & Bolts

- Mark Hammel Tal Lancaster
  - March 2007
  - Part 1 of 2

Disclaimer

The contents of these documents were created for a class that Mark Hammel and Tal Lancaster gave at Disney Animation Studios in March 2007.

They are just two people's experiences with PRMan over the years. They are not official statements by or for Disney or Pixar. Mark and I do wish to thank Disney for letting us share these with the RenderMan Community.

Also at the time we only had PRMan-13. There have been many improvements since then.
-- Tal Lancaster March 11, 2008

What Is RenderMan?

RenderMan Interface

- Scene description specification
- Device/implementation independent
- Two forms:
  - C language binding (RI API)

```
RiSphere(RtFloat radius, RtFloat zmin, RtFloat zmax,
RtFloat thetamax, ...parameterlist...);
```

  - RenderMan Interface Bytestream protocol

```
Sphere radius zmin zmax thetamax ...parameterlist...
```

RenderMan Interface

- Basically abandoned, although not officially
  - No longer updated
  - Last published 2005

RenderMan Interface Bytestream

- RIB
- File format for a scene description following the Interface
- Binary or ASCII (compressed)
- Generally requires external files (shaders, textures, other RIB files – *RIB archives*)

Implementations

- PhotoRealistic/Pixar's RenderMan (PRMan)
- Others:
  - 3Delight (dna research)
  - Aqsis (open source)
  - BMRT (freeware/Exluna, discontinued)
  - Entropy (Exluna, discontinued)
  - RenderDotC (Dot C Software)

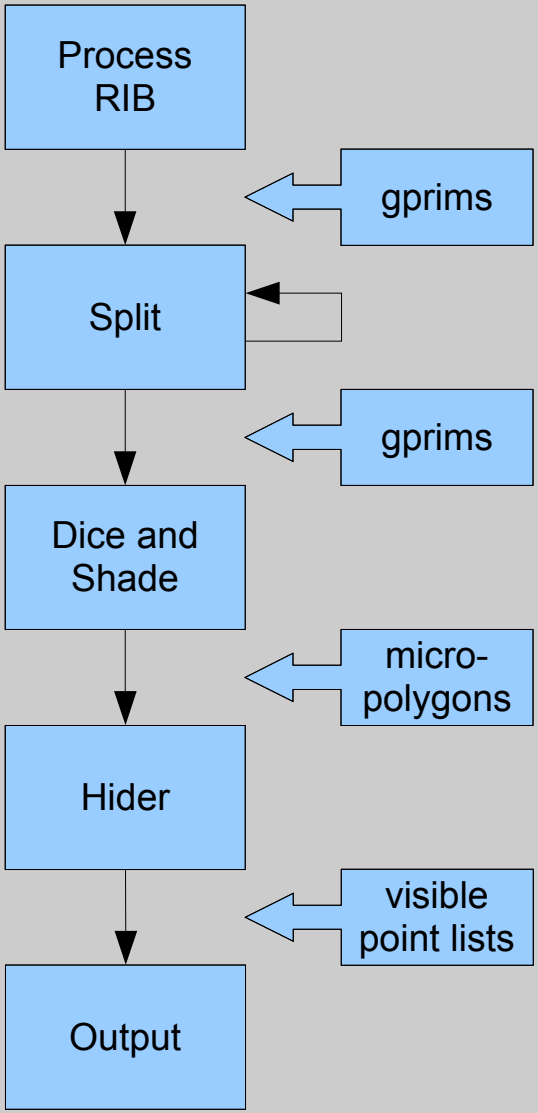
PRMan Geometric Pipeline

PRMan Geometric Pipeline

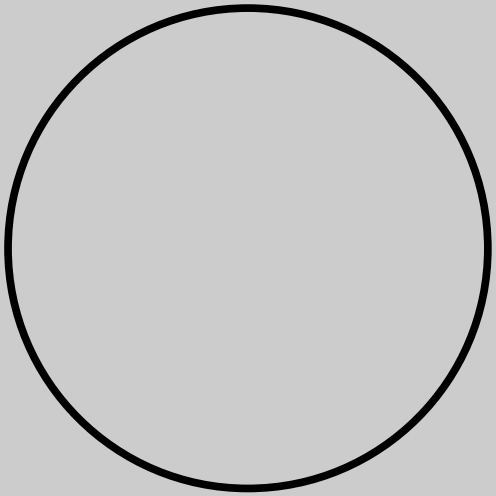
- Reyes algorthim
  - Lucasfilm (Pixar), mid-1980s
  - First published in Siggraph 1987
  - *Advanced RenderMan* (Apodaca, Gritz), 2000

PRMan Geometric Pipeline

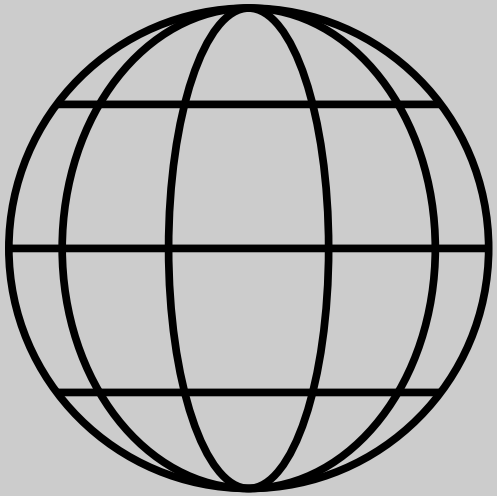
- Divide and conquer
  - Process RIB
  - Split primitives until manageable
  - Dice primitives into *grids*
    - Rectangular mesh of *micropolygons*
  - Shade micropolygons
  - Sample for output



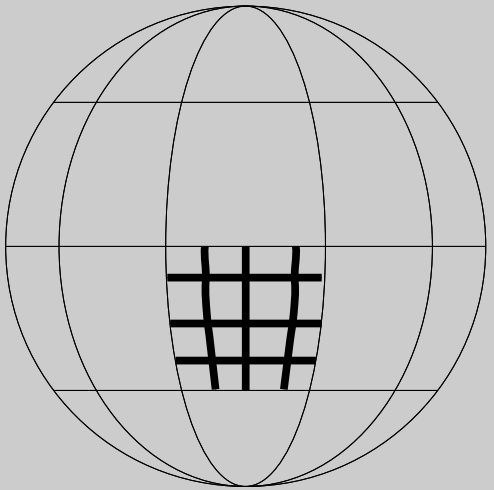
PRMan Geometric Pipeline Basics



Geometric primitive



Grids

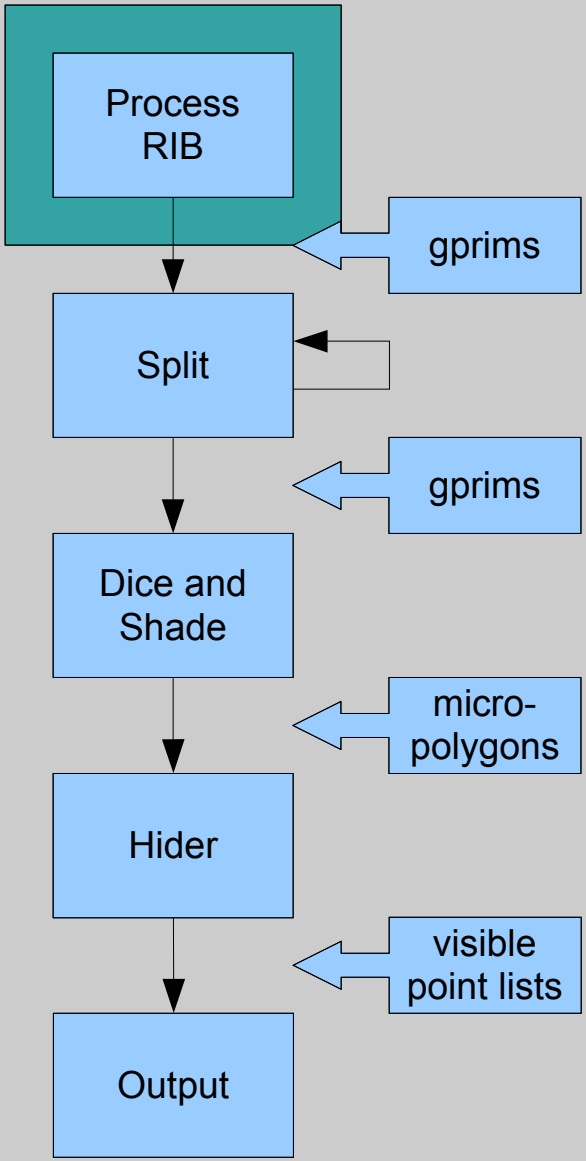


Micropolygons

PRMan Geometric Pipeline Basics

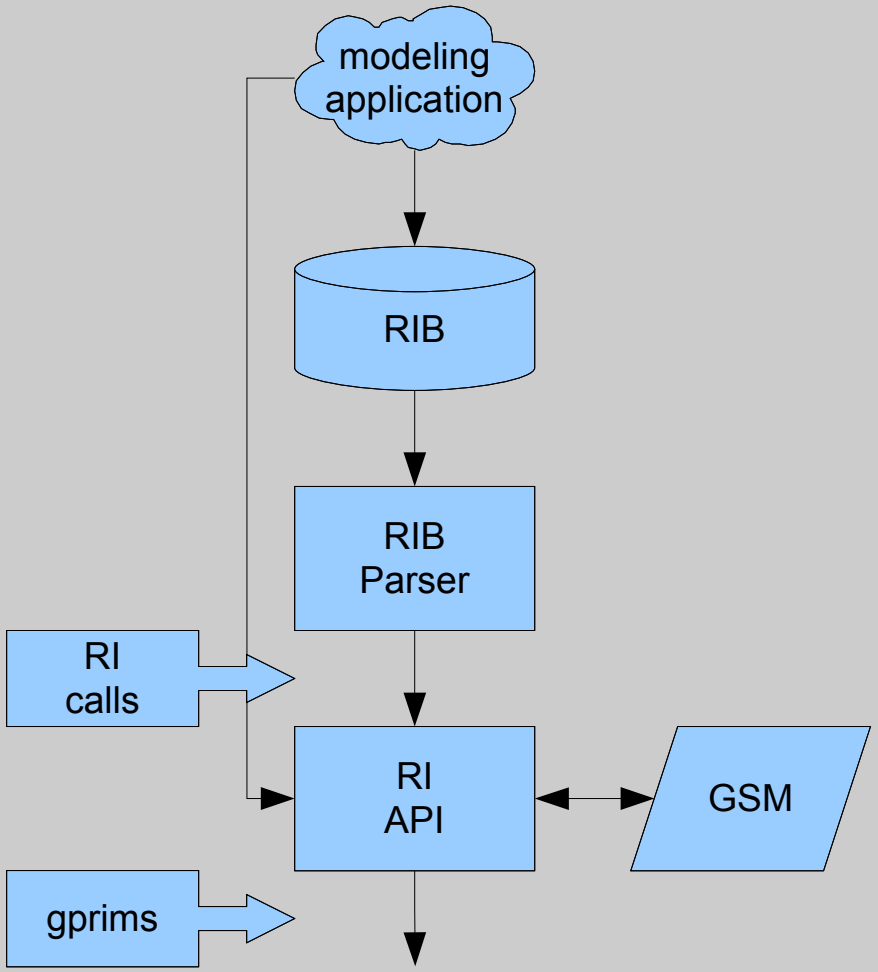
- Efficiency from:
  - Identical format to all primitives
  - Only processing necessary data

PRMan Geometric Pipeline



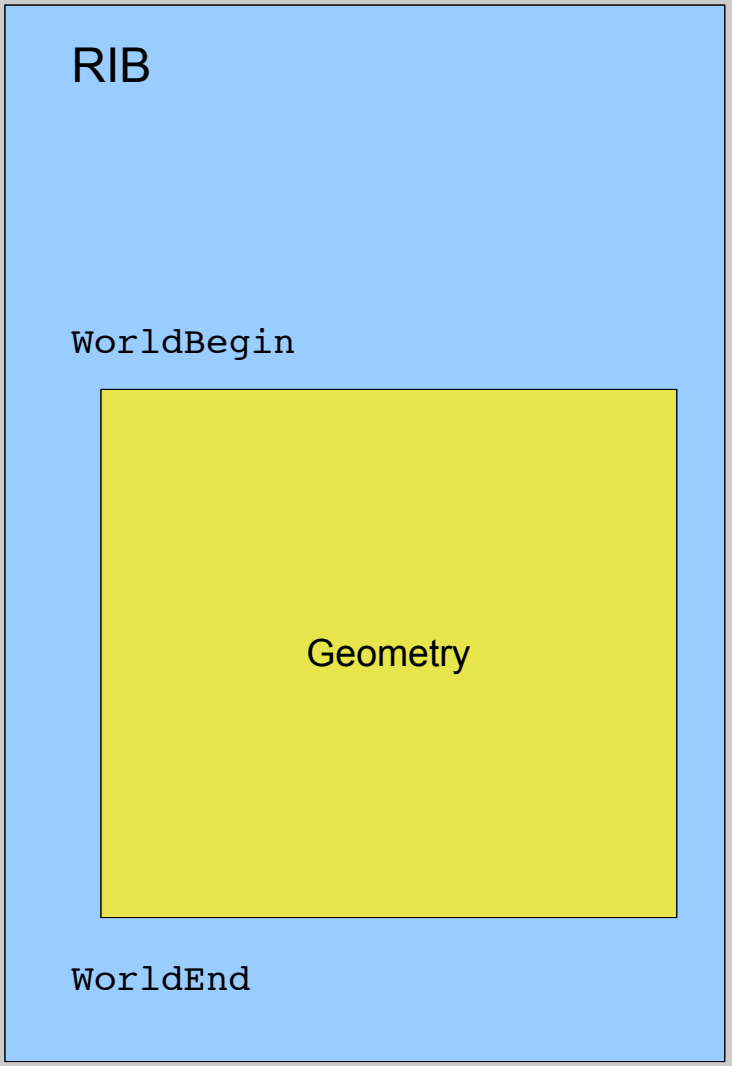
PRMan Geometric Pipeline

- RIB parsing
- RenderMan Interface processing
  - Two classes of RI calls
    - Geometric primitives
    - Graphics state
      - Information necessary to render geometric primitives



Geometric Primitives

- Between `WorldBegin/ WorldEnd` block of RIB
- Examples:
  - Quadrics (Sphere, Cone)
  - Subdivision Surfaces
  - Points and Curves
  - Procedural primitives



Geometric Primitives

- RIB example:

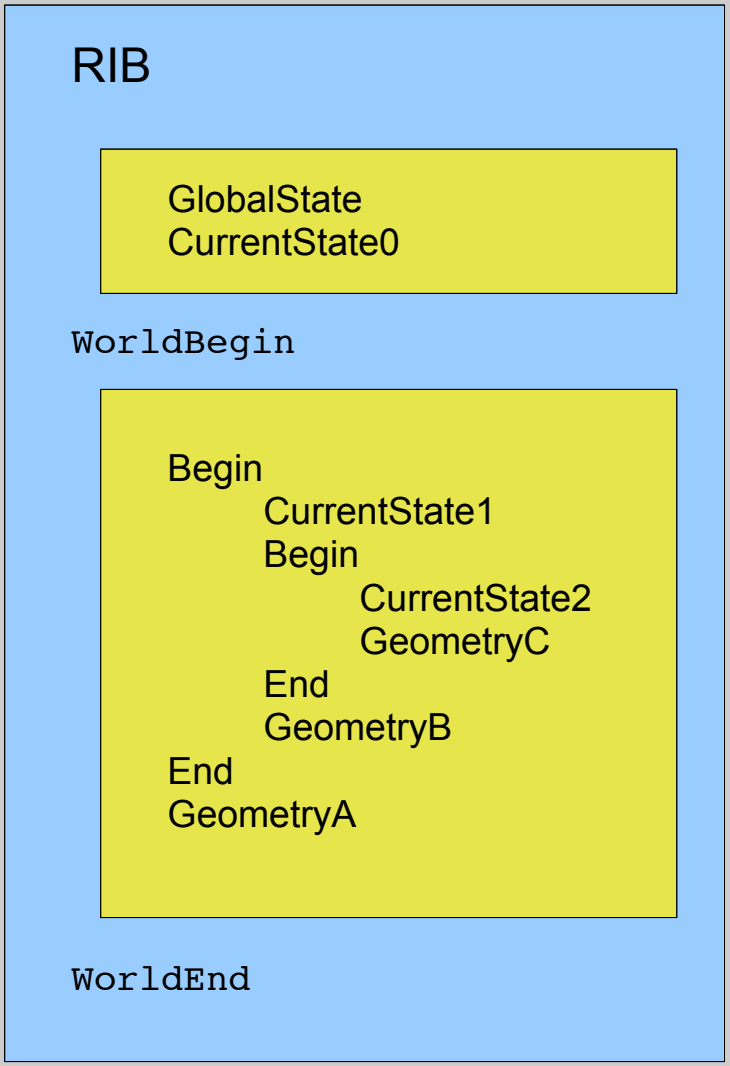
Standard parameters

```
SubdivisionMesh "catmull-clark" [4 4 ...]
[3 2 1 0 ...] ["stitch" "stitch" ...] [3 0 ...]
[1032 1 2 ...] []
"P" [8.6 15.2 17.6 ...] "st" [0 0.9 ...]
"vertex point __Pref" [0.54 0.72 0.96 ...]
"uniform float objectID" [ 3 ]
```

Primitive variables

Graphics State Machine

- Manages graphics state
- Two parts
  - Global state
    - Constant across all geometric primitives
  - Current state
    - Can change from primitive to primitive
    - Stack-based



Global State

- Options
  - Search Paths
  - Bucket Size
  - Texture Memory
  - Pixel Filter
  - Camera

Current State

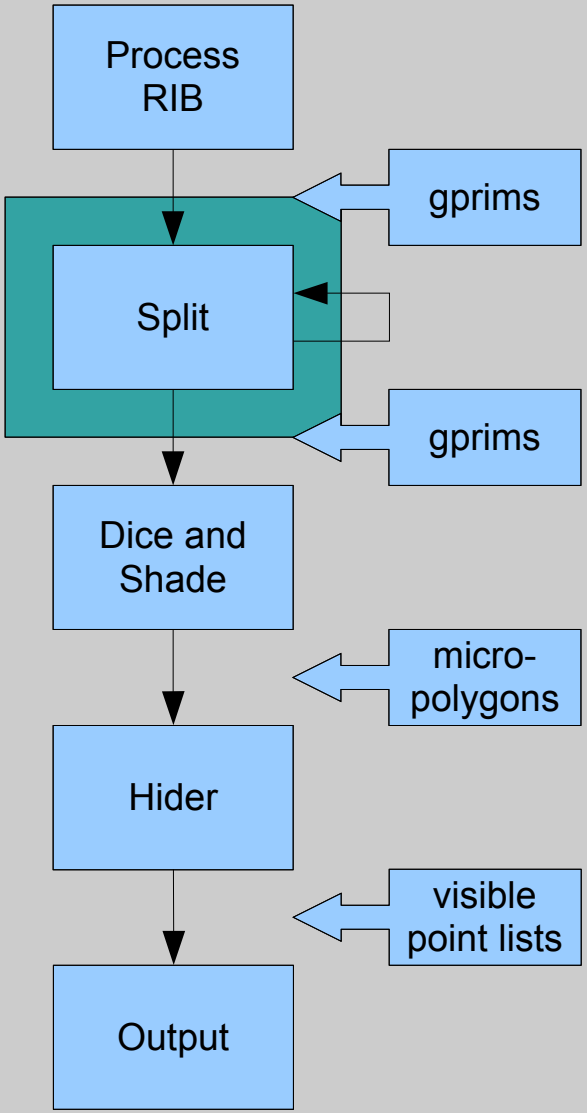
- Attributes
  - Displacement Bounds
  - Shading Rate
  - Shaders
  - Transforms
- Begin/End Blocks
  - `AttributeBegin/AttributeEnd`
  - `TransformBegin/TransformEnd`

User Specified Attributes

- Arbitrarily defined token/value pairs
- Have no effect on geometric pipeline
- Can be queried in shaders
- RIB example:

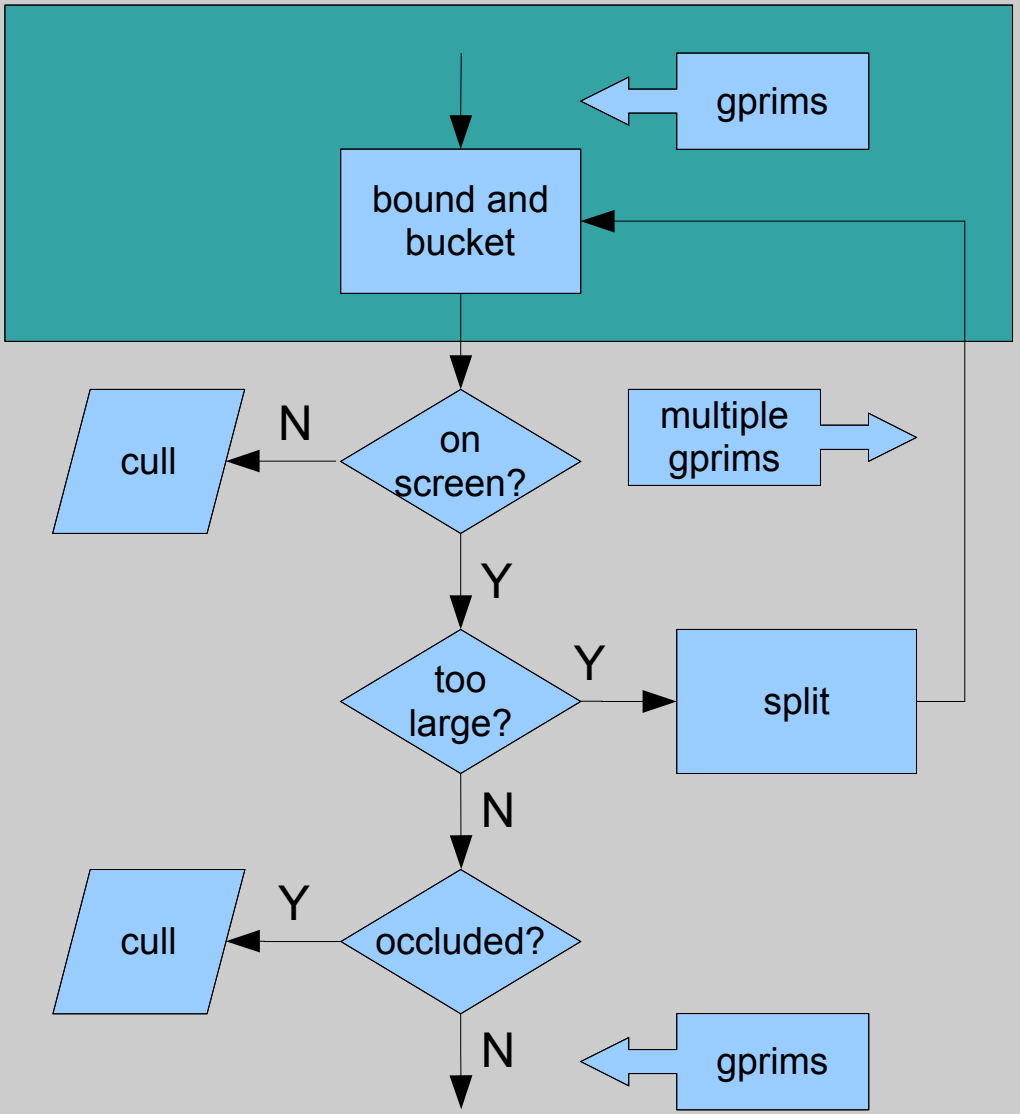
```
Attribute "user"
"uniform color rgbScale" [0.5 0.9 0.4]
```

PRMan Geometric Pipeline



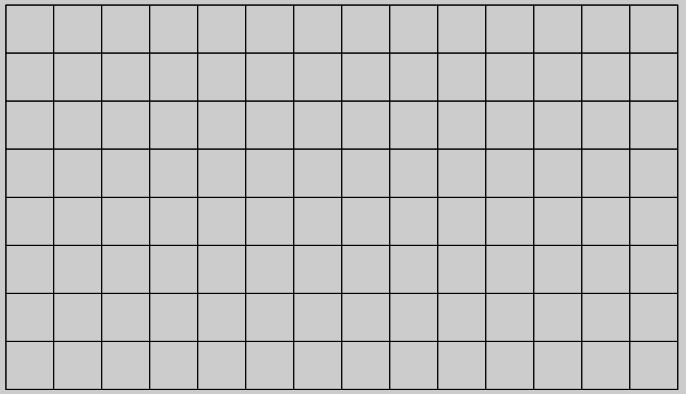
PRMan Geometric Pipeline (cont)

- Camera-space axis-aligned bounding box calculated
- Depth-sorted into buckets



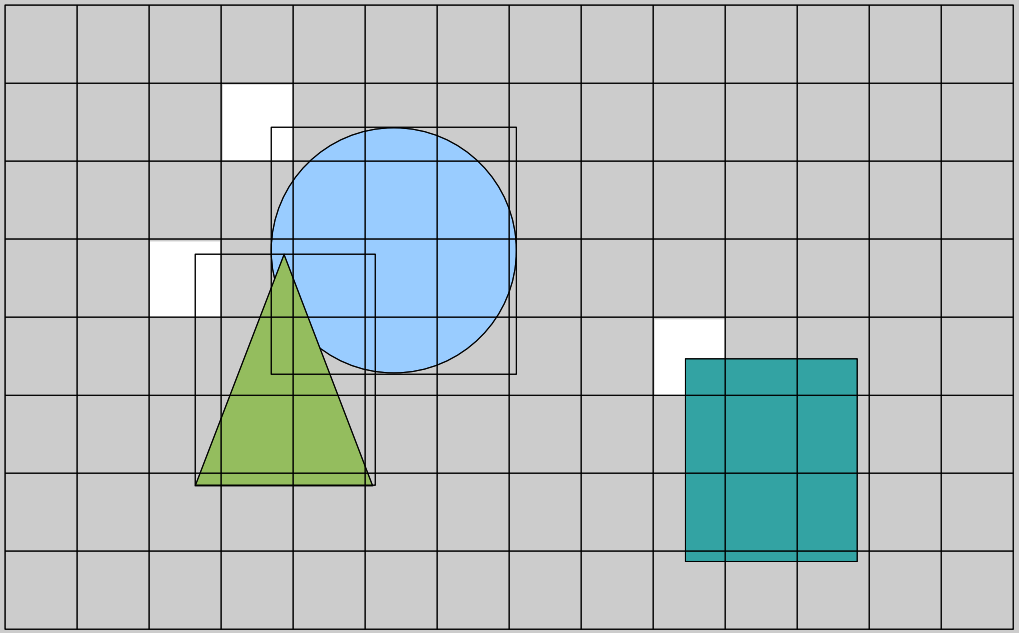
Buckets

- Image plane is divided into rectangular pixel regions – *buckets*
- RIB example:



```
Option "limits" "bucketsize" [16 16]
```

Bucketing

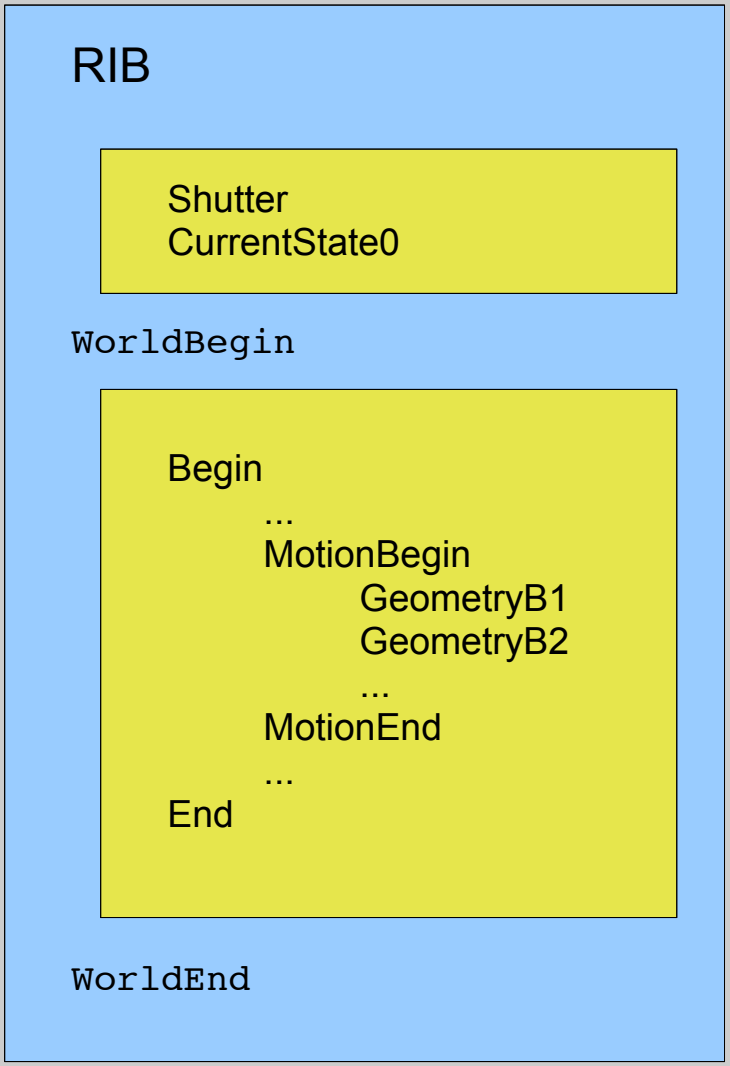


Motion Blur

- Transformation or deformation
- Moving geometric primitive, with time-based positions along motion path
- Bounding box includes all positions

Motion Blur

- RIB components
  - Shutter option
    - Specifies open and close times for camera shutter
  - Motion block
    - Encloses only transforms or primitives at different times
    - Definitions must be identical in form, but not value



Motion Blur

- RIB example:

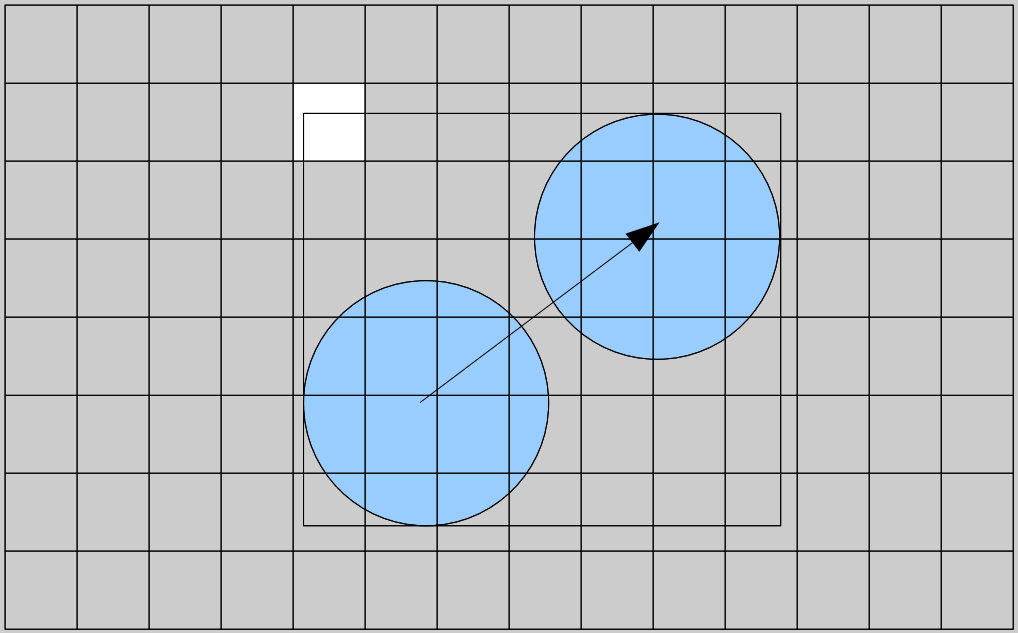
```
Shutter 0 0.5
MotionBegin 0 1
   Translate x x x
   Translate y y y
MotionEnd
MotionBegin 0 1
   Sphere x x x x
   Sphere y y y y
MotionEnd
```

Multi-segment Motion Blur

- RIB example:

```
Shutter 0 0.75
MotionBegin 0 0.5 1
   Translate x x x
   Translate y y y
   Translate z z z
MotionEnd
```

Motion Blur



Clamping Shutter Motion

- RIB example:

```
Shutter 0 0.5
MotionBegin [0 1]
  Translate 0 0 1
  Translate 0 0 2
MotionEnd
MotionBegin [0 1]
  Rotate  0 1 0 0
  Rotate 90 1 0 0
MotionEnd
```

Clamping Shutter Motion

- Standard method – interpolate, then concatenate:

```
Translate 0 0 1.5
Rotate 45 1 0 0
```

Clamping Shutter Motion

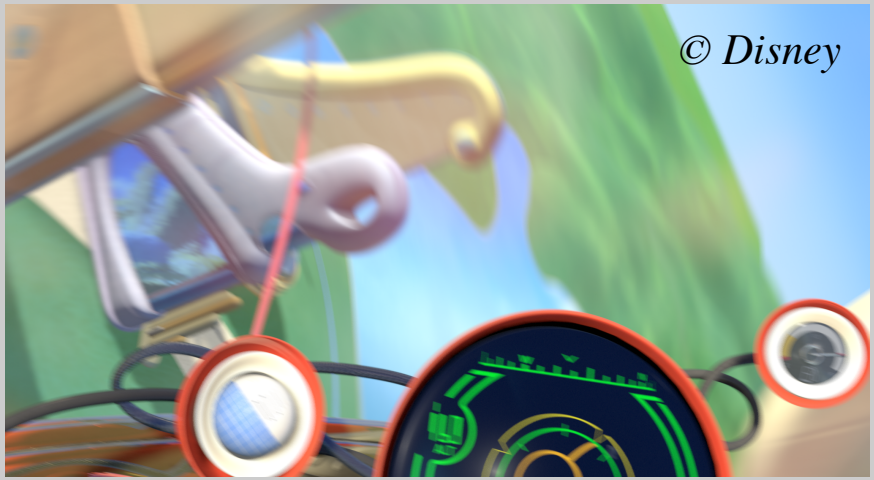
- Standard method – interpolate, then concatenate:

```
Translate 0 0 1.5
Rotate 45 1 0 0
```

- New method – concatenate, then interpolate:

```
concat(Translate 0 0 1, Rotate 0 1 0 0)
concat(Translate 0 0 2, Rotate 90 1 0 0)
```

Clamping Shutter Motion



Images are Copyrighted by Disney, from the picture “Meet The Robinsons”.

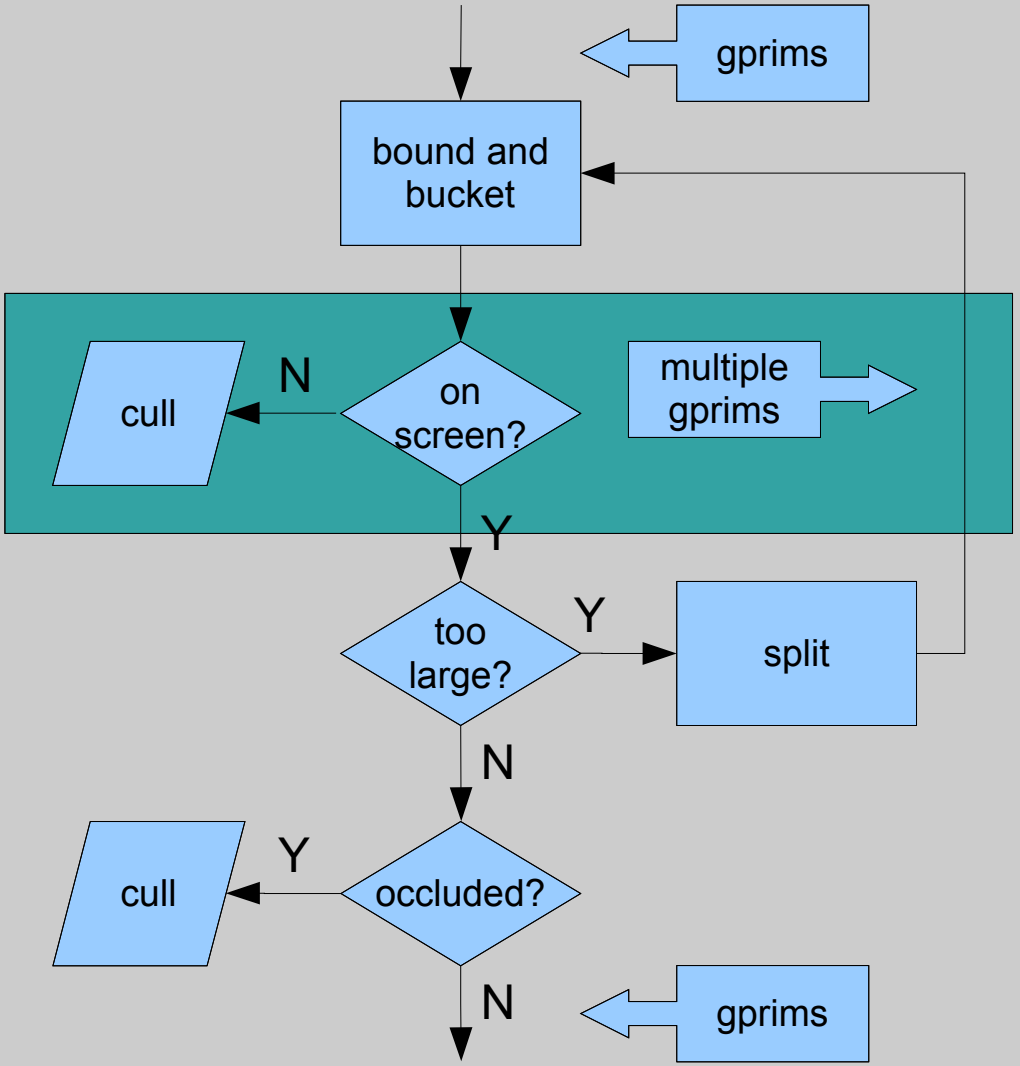
Clamping Shutter Motion

- RIB example:

```
Option "shutter" "int clampmotion" [0]
```

PRMan Geometric Pipeline (cont)

- Screen visibility tested
  - Camera volume
  - Backfacing



Sidedness and Orientation

- Objects can be single or double-sided
- RIB example:

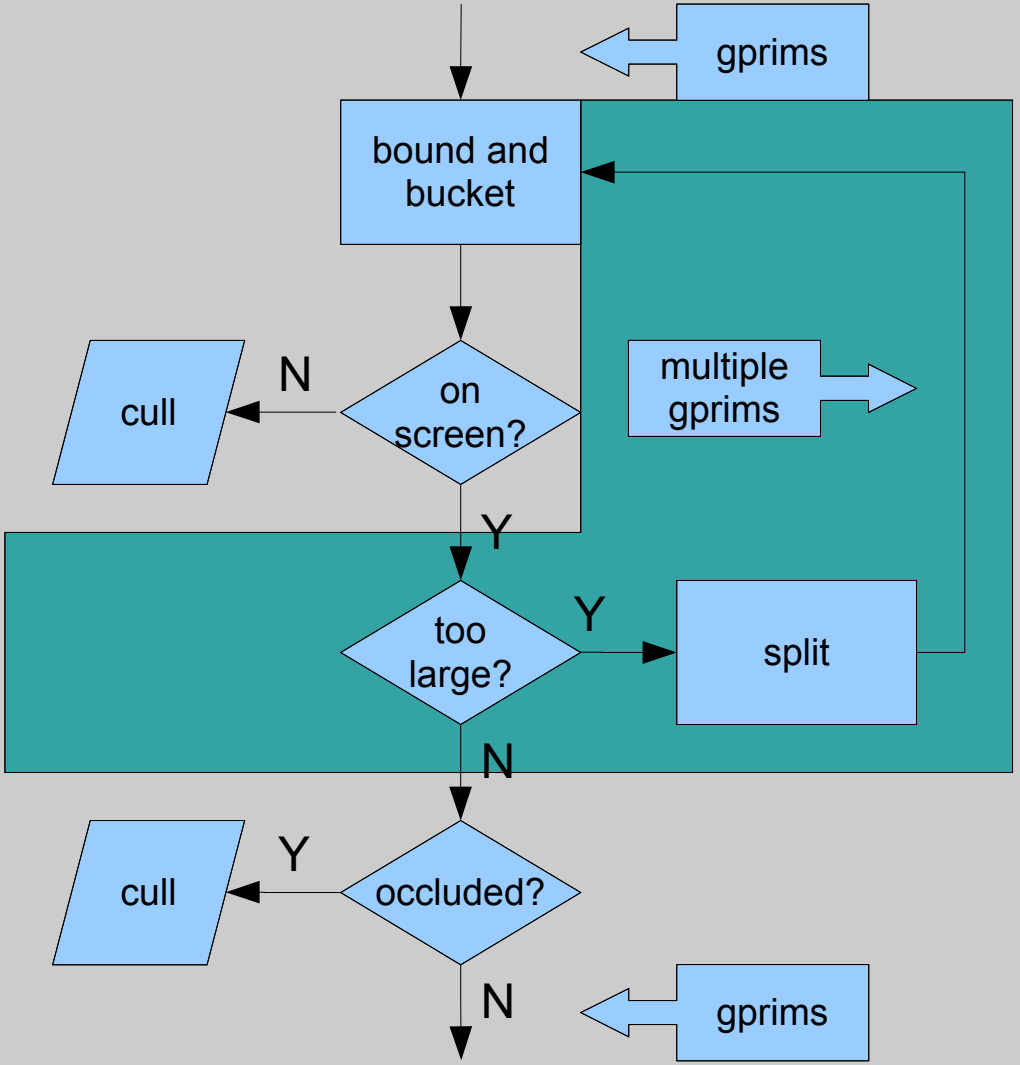
```
Sides 1
```

- The orientation (geometric normal) can be reversed
- RIB example:

```
ReverseOrientation
```

PRMan Geometric Pipeline (cont)

- Size tested
  - User controlled with *grid size*
    - Maximum number of micropolygons in grid
  - Ideal – primitive fits in bucket



Grid Size

- Geometric primitives will be turned into grids
- Grid is rectangular mesh of micropolygons
- Size of mesh determined by *shading rate*
  - Area of a micropolygon in pixels
  - Side length is *sqrt(shading rate)*
- Grid size
  - Maximum number of micropolygons allowed in a grid

Grid Size

- Standard grid size formula
  - *grid size =*
    *(bucket size X * bucket size Y) / shading rate*
  - Calculates the number of micropolygons in a bucket
    - Shading rate is area in pixels, so *1 / shading rate* is the number of micropolygons in a pixel
      - Shading rate 0.25 = 4 micropolygons
- If grid size is set bigger, then more micropolygons will created than will be processed in a single bucket

Grid Size

- Grid size is classical *time vs. space* trade-off
  - Larger grid size results in faster renders
    - Gain efficiency from shading more points at once
  - Larger grid size results in more memory usage
    - More micropolygons kept in memory
    - More memory for shading samples

Shading Rate and Grid Size

- RIB example:

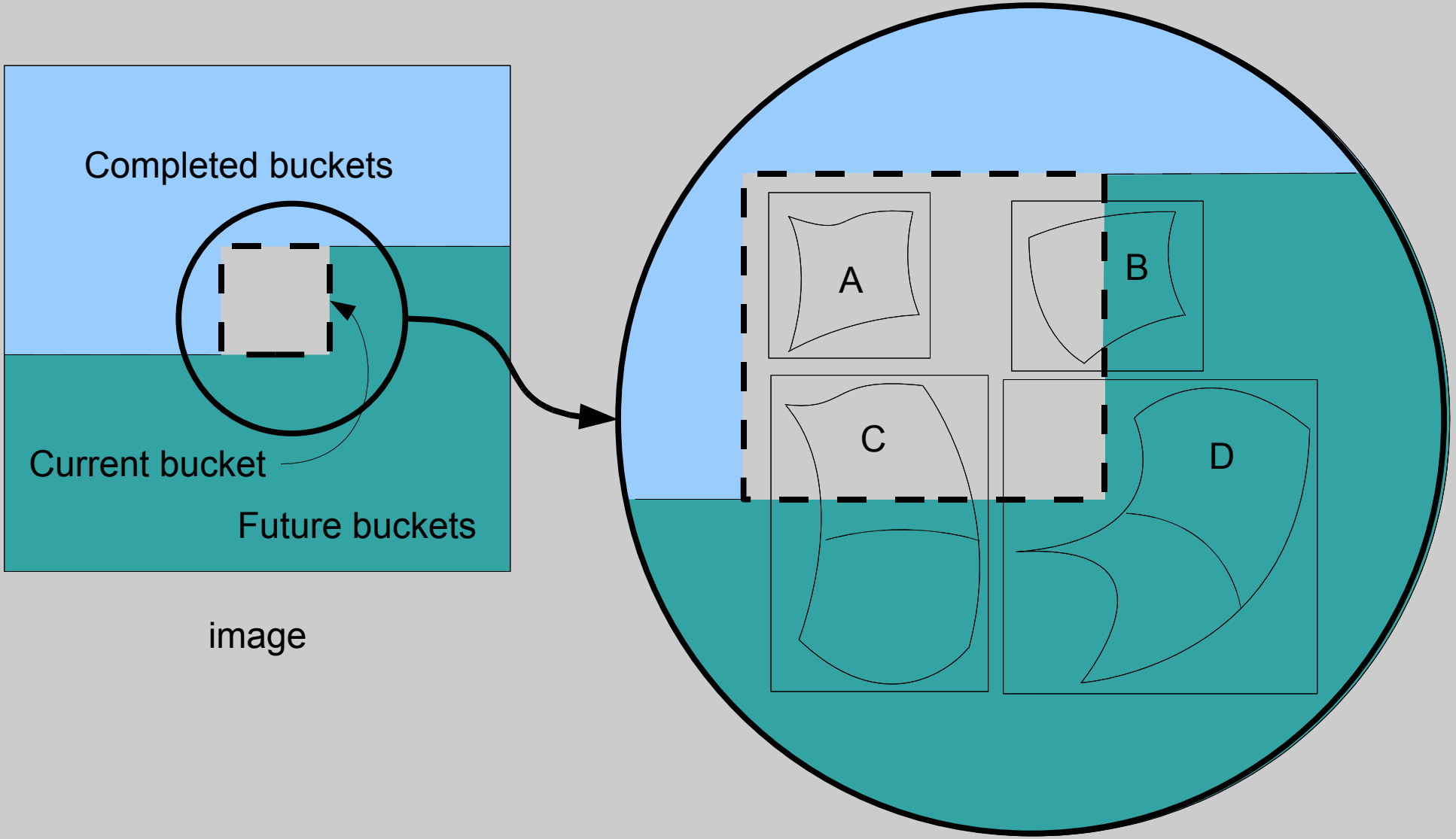
```
ShadingRate 1

Option "limits" "gridsize" [256]
```

Motion Factor

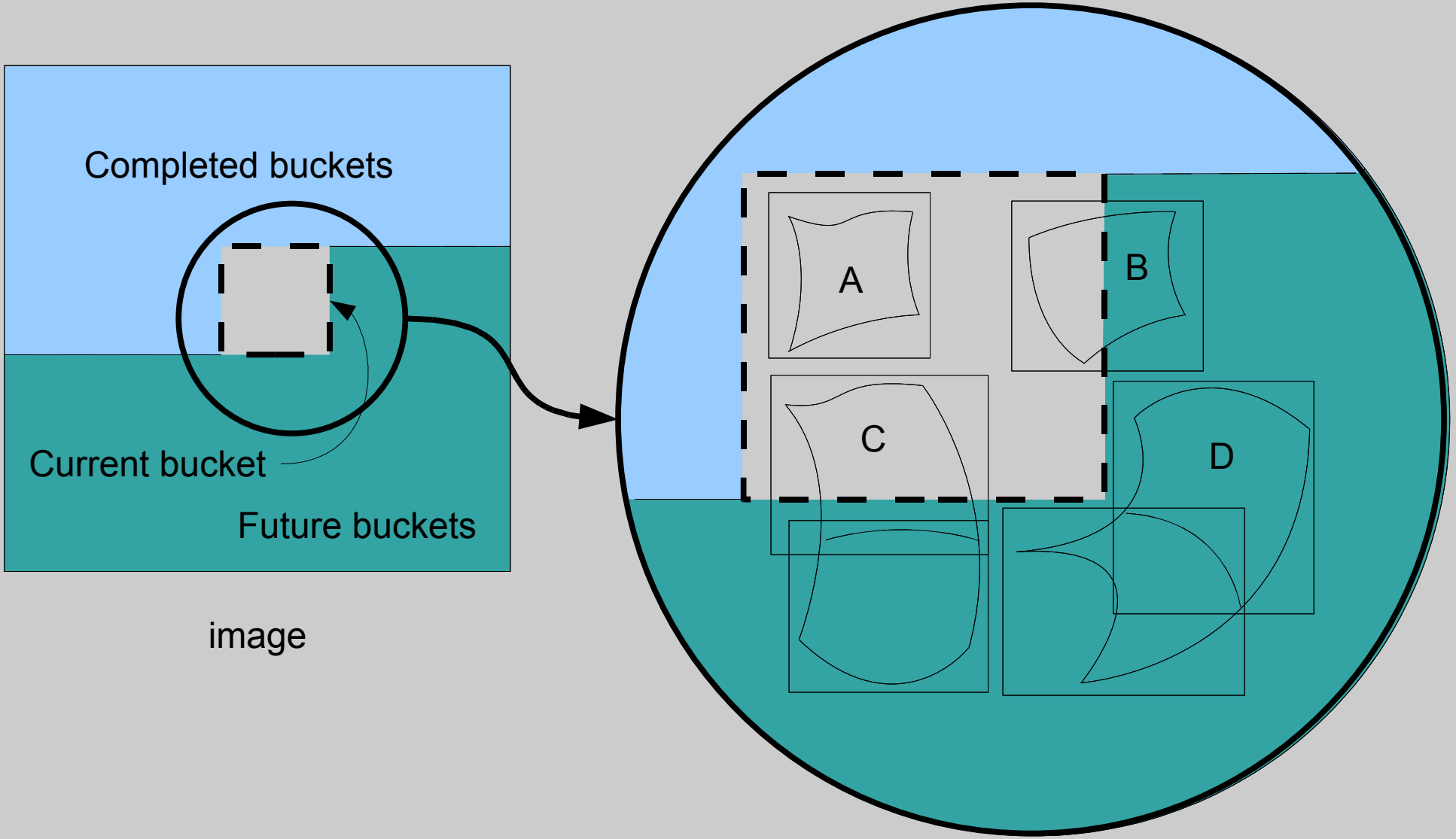
- Increases shading rate based on length of primitive blur in screen space
- RIB example:

```
GeometricApproximation "motionfactor" 1
```

Splitting


Completed buckets
Current bucket
Future buckets
image
A
B
C
D

Splitting



image

Procedural Primitives

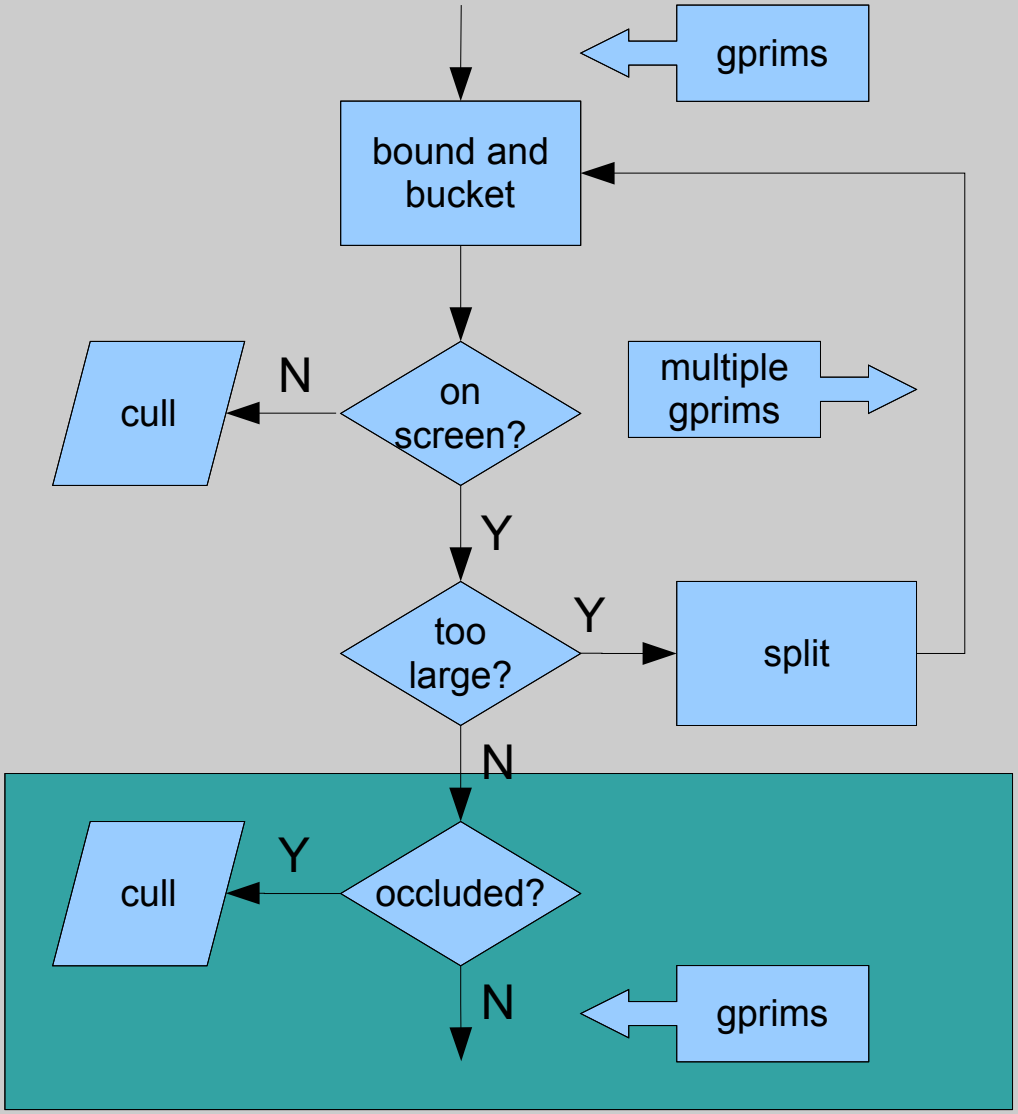
- Treated like every other geometric primitive
- Must implement own splitting
  - Split into non-procedural primitives
  - Split into other procedural primitives
    - Implement splitting as recursive routine which ends with non-procedural primitives

Eyesplits

- Prior to PRMan 12.0, primitives crossing the camera plane might not be split effectively
  - Perspective projection only works for primitives in front of camera
  - Primitives needed to be split into portions in front of and behind camera
  - Potential infinite splitting in halves
- In PRMan 12.0, eyesplits problems were “eliminated”
  - But there is still a cost

PRMan Geometric Pipeline (cont)

- Occluded primitives culled
  - Depth sorting
  - Bucket coverage tracked



Bucket Coverage

- Opacity Culling
  - Accumulated opacity considered to be completely opaque

```
Option "limits" "othreshold" [0.996 0.996
0.996]
```

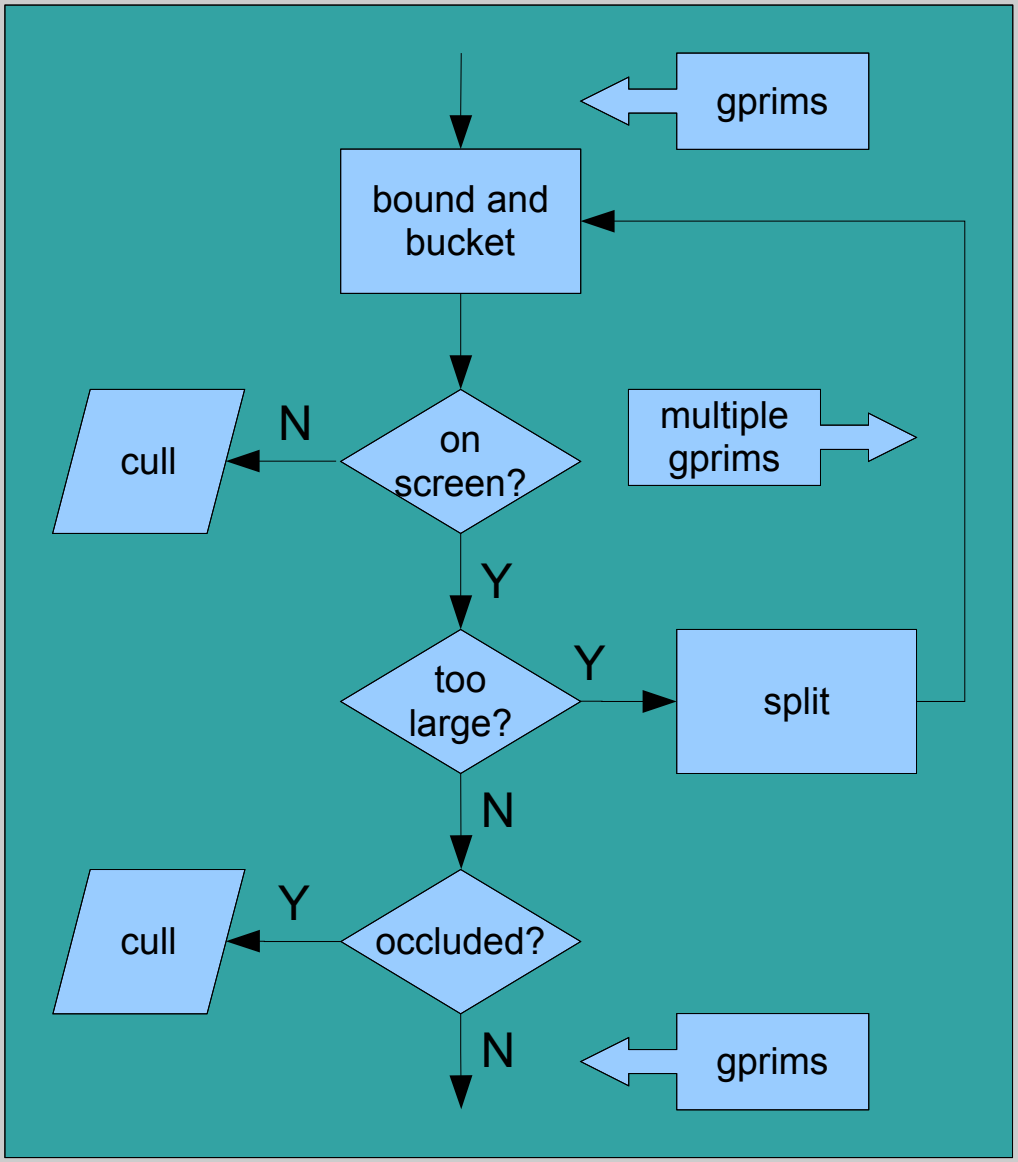
Not Bucket Coverage

- Opacity Threshold
  - Primitives with opacity less than opacity threshold will be excluded from standard shadows

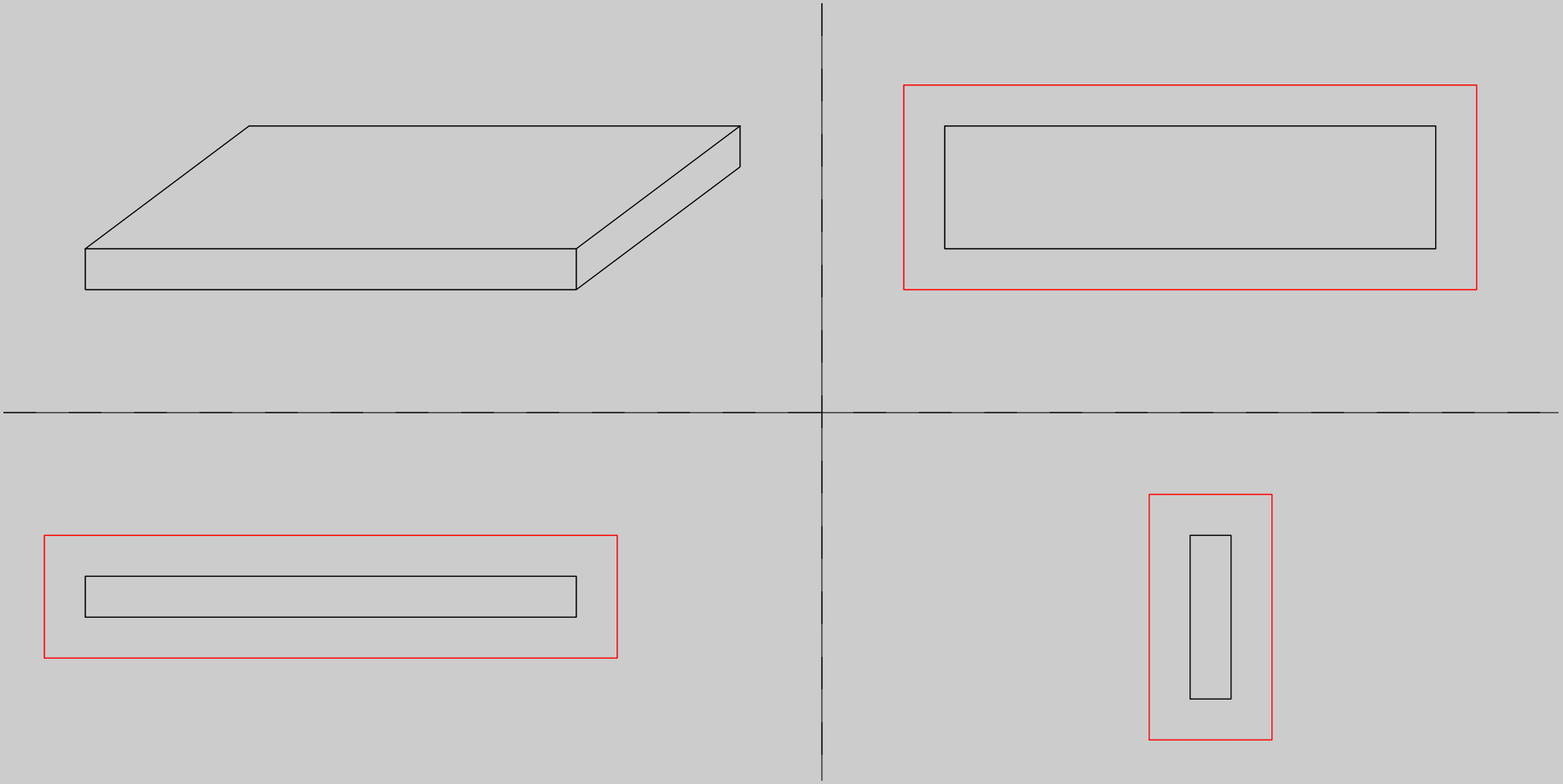
```
Option "limits" "zthreshold" [0.5 0.5 0.5]
```

PRMan Geometric Pipeline (cont)

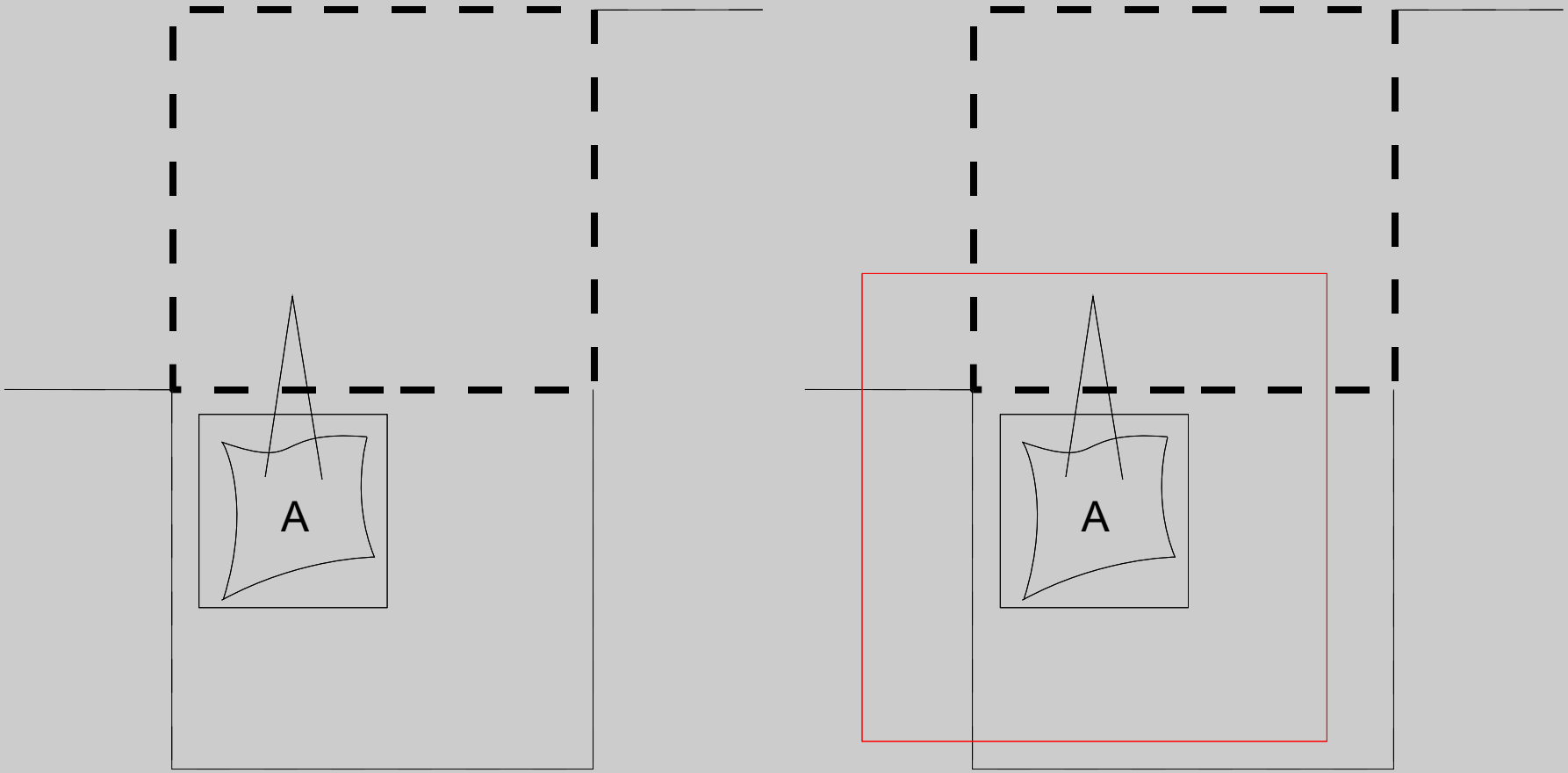
- Displacement Bounds
  - Addition to bounding box to account for yet-to-be-calculated displacements



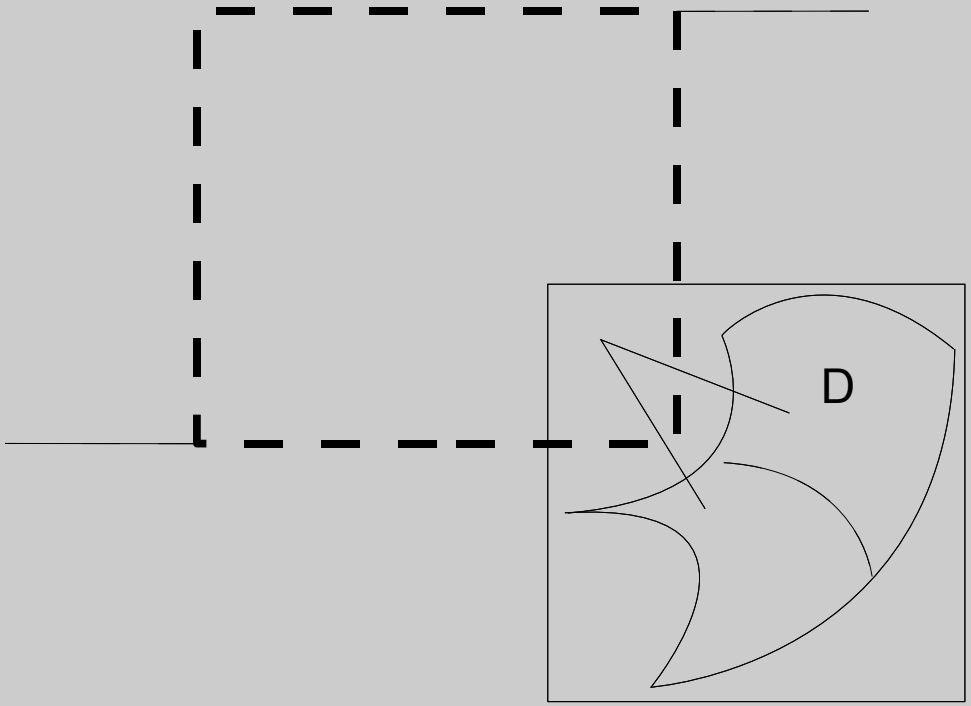
Displacement Bounds



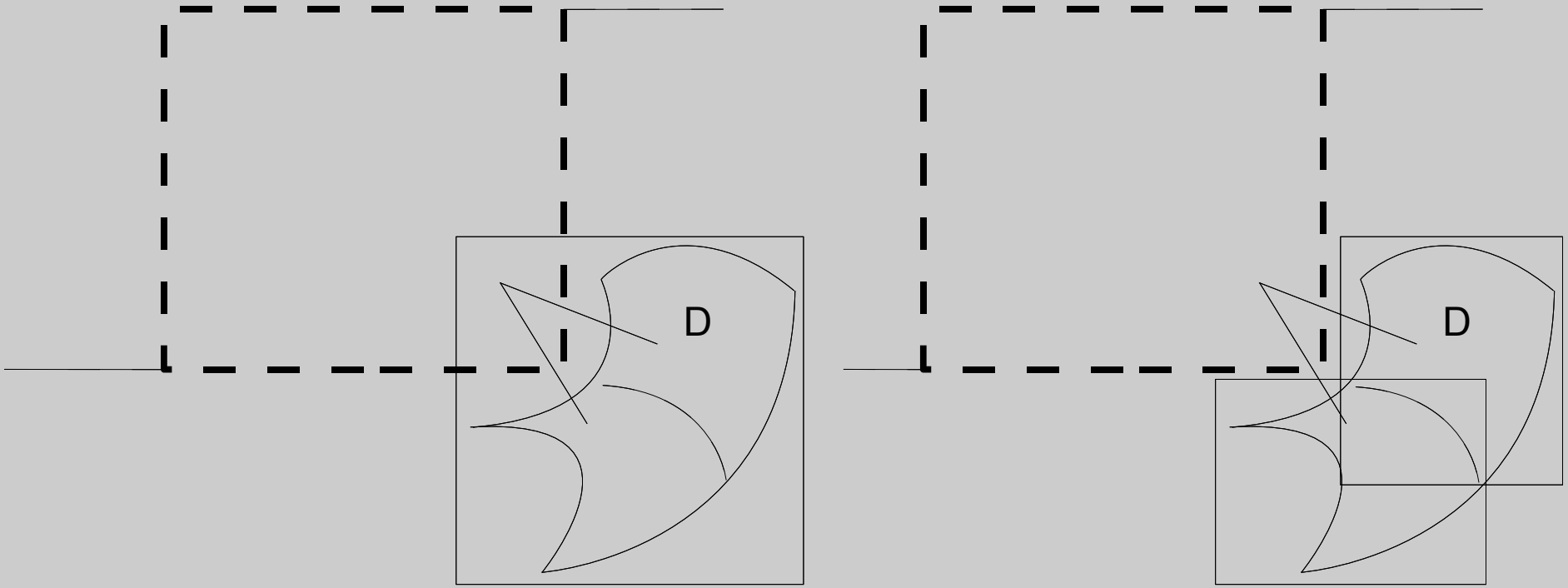
Displacement Bounds – Bucketing



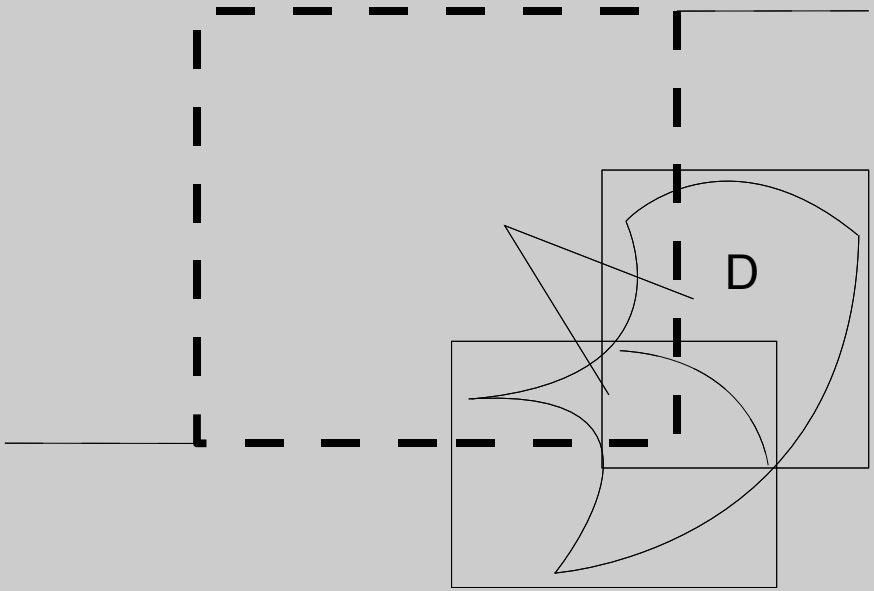
Displacement Bounds – Bucketing



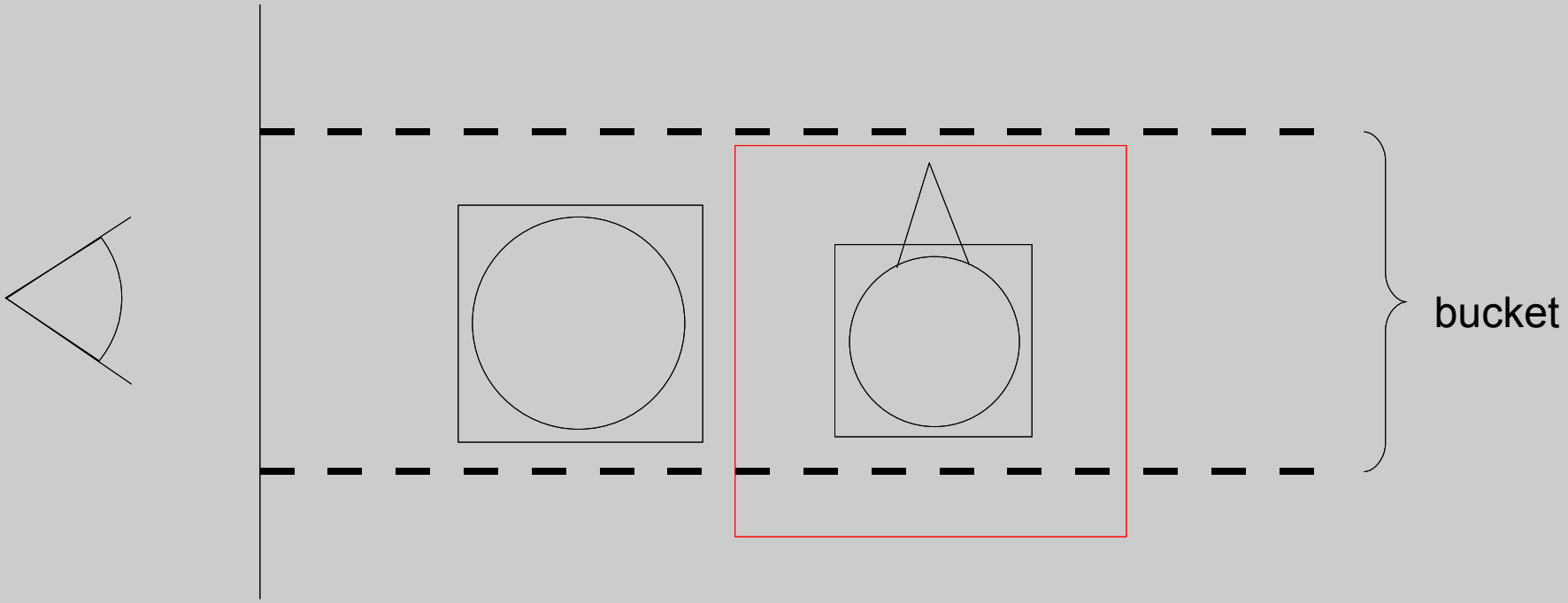
Displacement Bounds – Bucketing



Displacement Bounds – Bucketing



Displacement Bounds – Occlusion

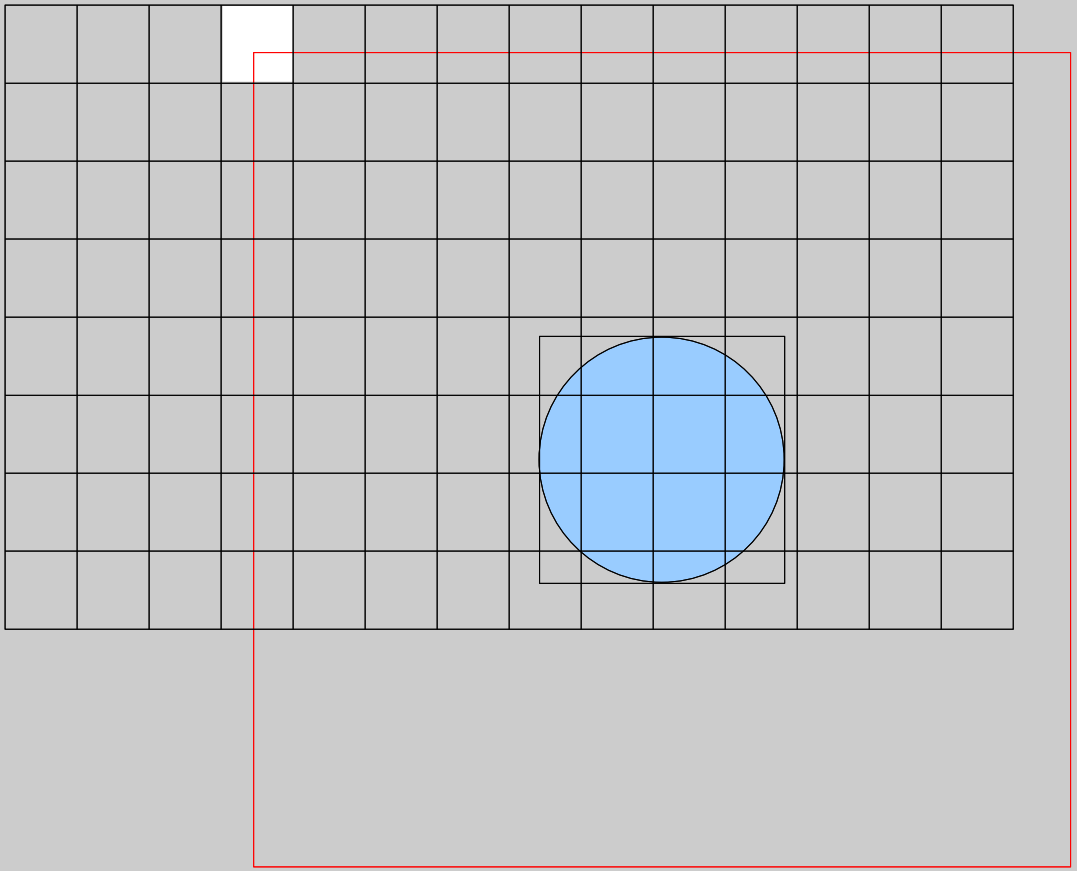


Displacement Bounds

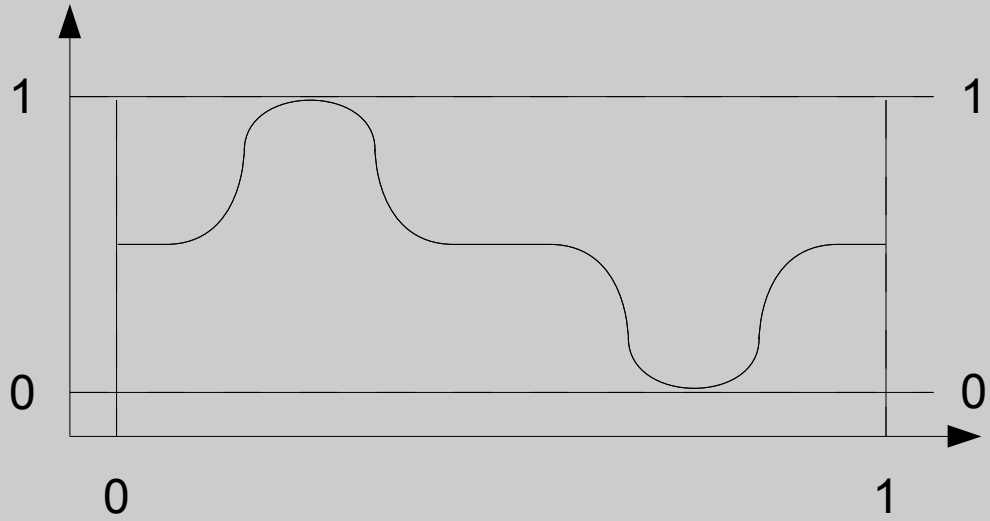
- Keep as small as possible
  - Reduces memory
    - Primitives not processed until absolutely necessary
  - Increases speed
    - Ensures effective occlusion culling
- RIB example:

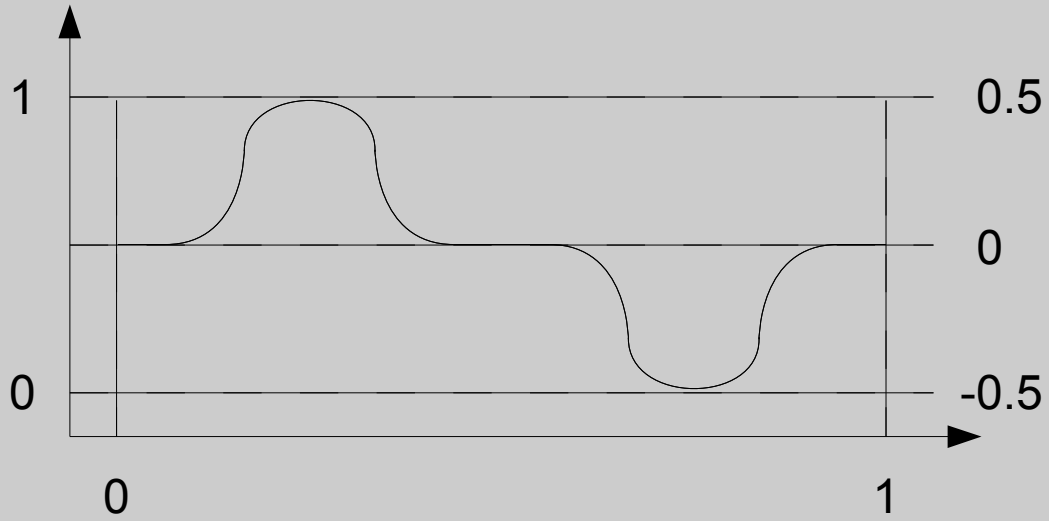
```
Attribute "displacementbound" "sphere" [2]
"coordinatesystem" ["object"]
```

Displacement Bounds



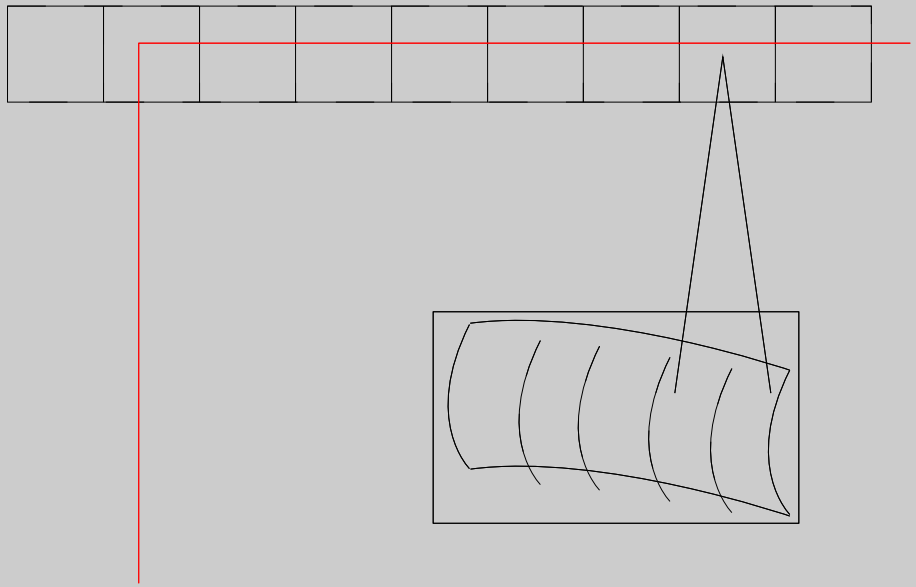
Displacement Bounds





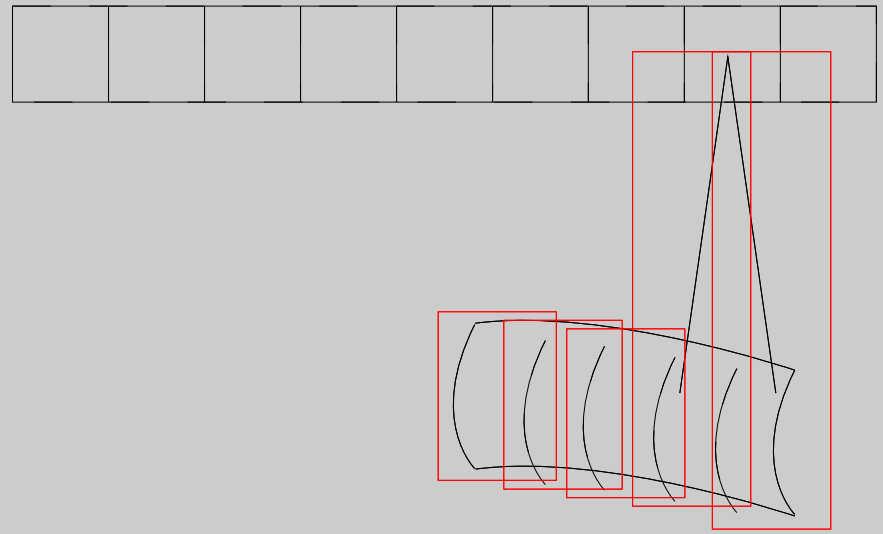
Extreme Displacement

- Runs shaders to calculate exact displacement bound for primitives



Extreme Displacement

- Costs extra evaluation time to save memory



Extreme Displacement

- RIB example:

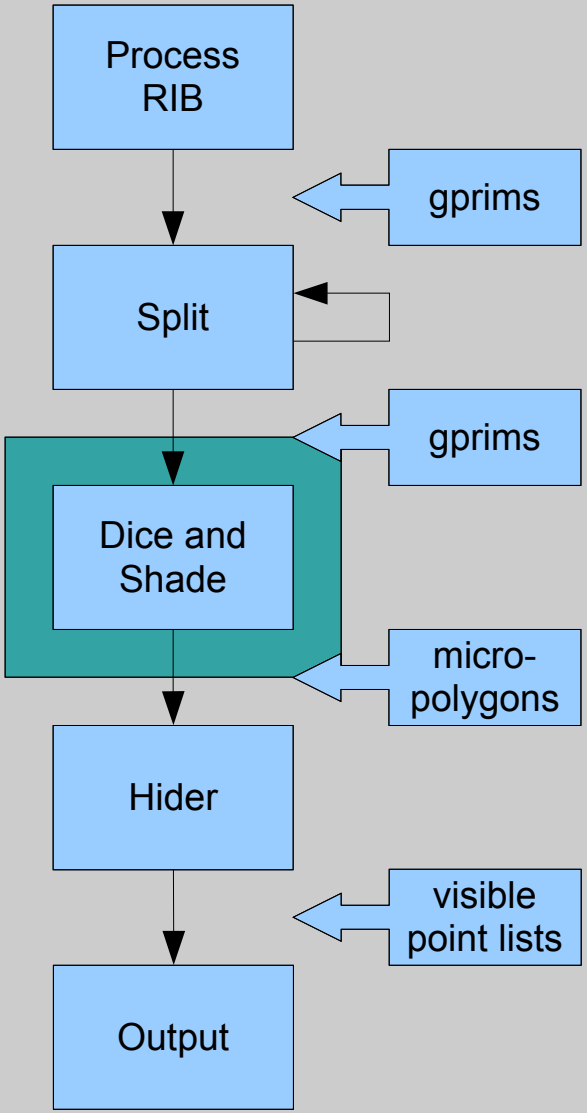
```
Option "limits" "extremedisplacement" [24]
```

- If displacement bound exceeds value in pixels, shaders are applied
- Default: 32

Extreme Displacement

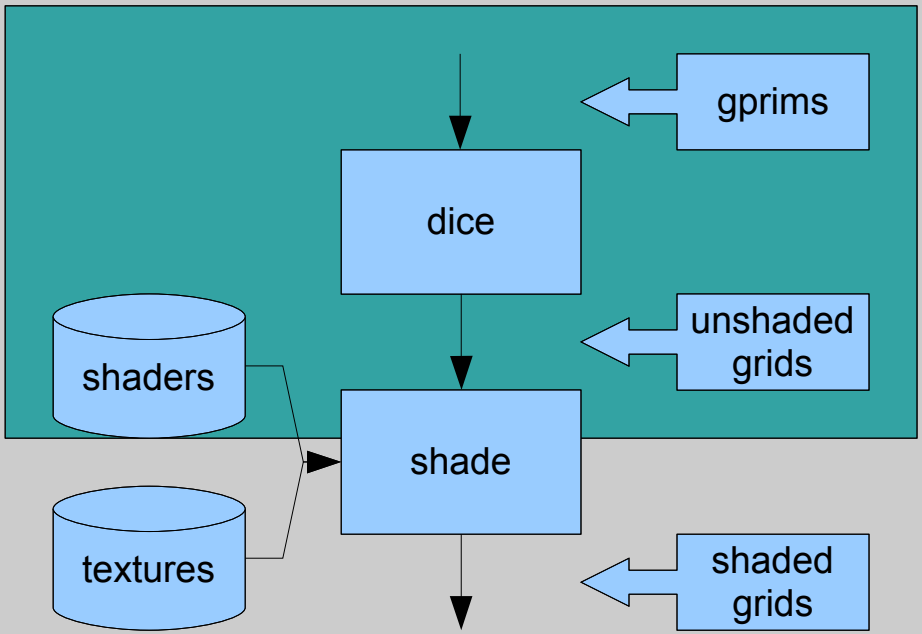
- “Gone in 13.5”
  - Presumably still a cost for large displacements

PRMan Geometric Pipeline



PRMan Geometric Pipeline (cont)

- Dicing: Tessellation into micropolygon grids
- All geometric primitives now have same representation
- Attributes associated with grid, per-vertex primitive variables interpolated



Internal Cracks

- Prior to PRMan 3.9, differences in dicing of adjacent grids in a split primitive could introduce cracks
- In PRMan 3.9, internal cracks were eliminated

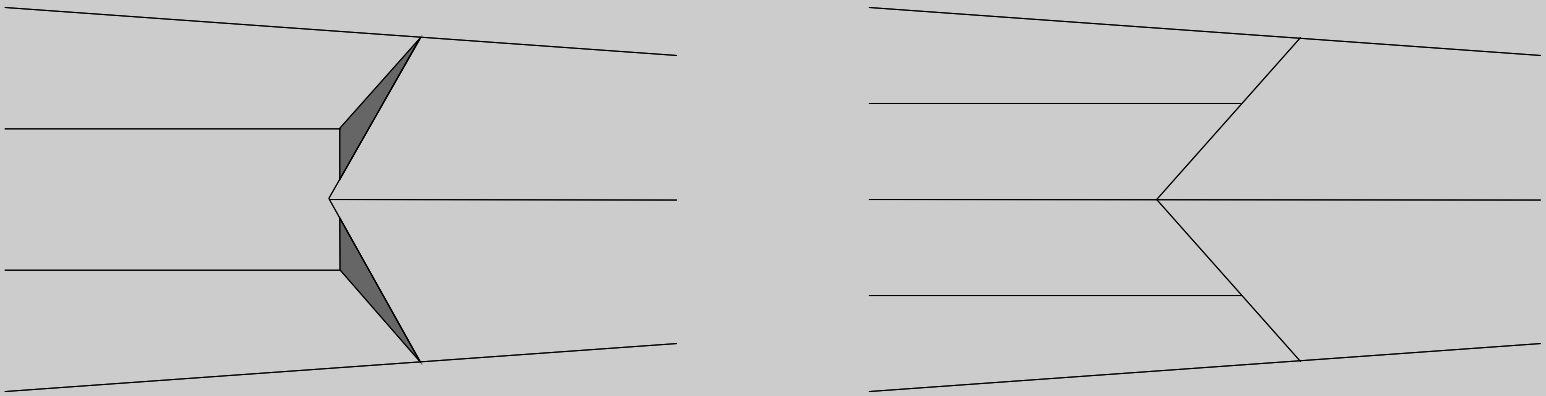
Patch Cracks

- Cracks between abutting geometric primitives
  - Due to:
    - Different displacement evaluation along abutted edge
    - Unchecked deformations
  - Fixed for subdivision with stitch tags
  - Still a potential problem for NURBS, poly meshes
    - Non-deforming less susceptible
    - Binary dicing helps

Binary Dicing

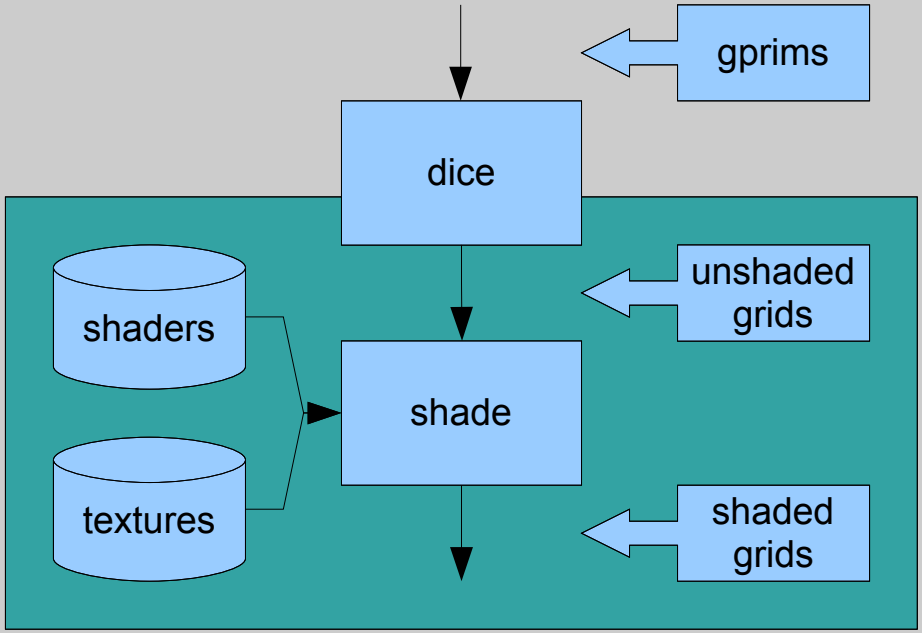
- Ensures compatible griding on abutting primitives
- RIB example:

```
Attribute "dice" "binary" [1]
```



PRMan Geometric Pipeline (cont)

- Shaders
  - ASCII `.sl` files compiled into platform-independent byte-code `.slo` files
- Shading order
  - Displacement
  - Surface
    - Light (cached)
  - Atmosphere



Shaders

- Evaluated at grid vertices
  - Shading samples
  - Shading rate
    - Frequency of shading
- Two basic types of parameters
  - Uniform
  - Varying

Shading

- Single instruction, multiple data (SIMD) interpreter
  - Each instruction applied to all shading samples at once
  - Benefits
    - Shader overhead per-grid, not per-sample
    - Uniform calculations performed once per grid
    - Have derivatives of any shader parameter
  - Conditionals require *active/inactive* run flags
  - Not depth first

Shading

- To get greater detail, increase number of shading samples
  - Smaller Shading Rate
  - Increases render time and memory
    - More shading samples to calculate
    - More shading samples to keep in memory

Double-sided Shading

- Duplicates grids with orientation reversed
- Each is shaded independently
- Mixed results in actual use
- RIB example:

```
Attribute "sides" "doubleshaded"
```

Combined Shading

- Warning:

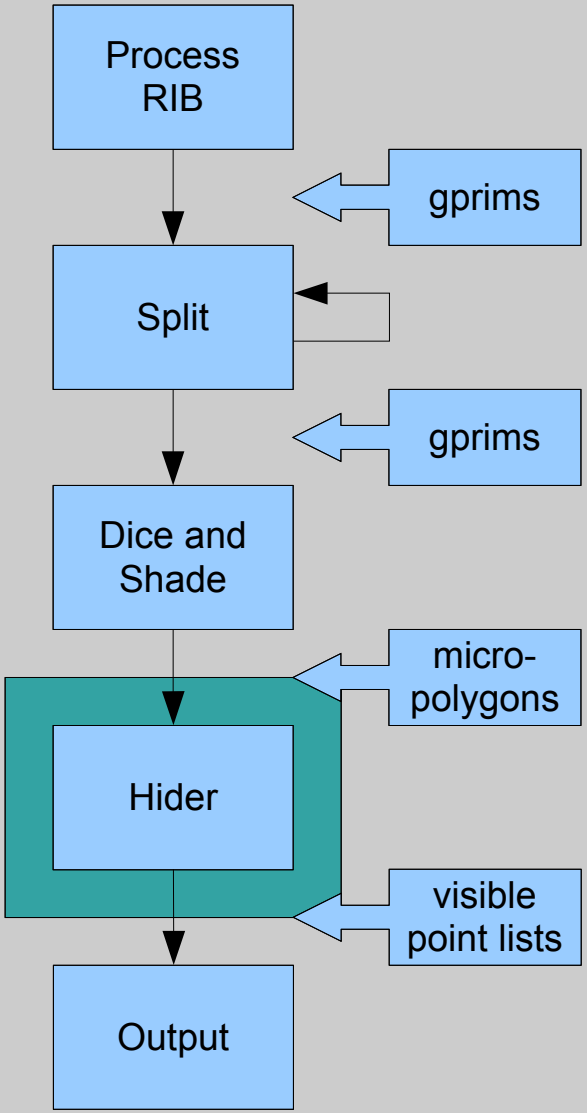
```
Small average grid size (# points). It
may be possible to speed up shading by a
factor of #x; see the multi-gprim
statistics for details. (PERFORMANCE
WARNING)
```

- Combined shading allows multiple grids to be shaded as one

Combined Shading

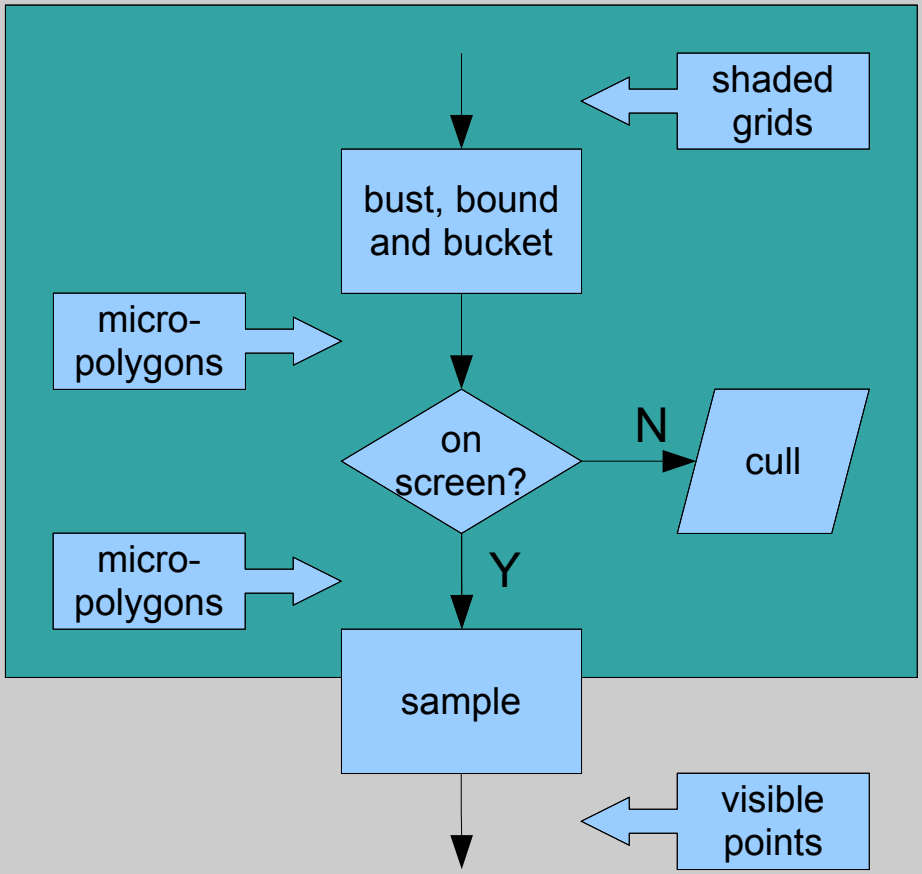
- Shaders need to fit SIMD interpreter
  - Shaders must be the same
  - Uniform variables must have identical values across all grids
  - Uniform variables which differ need to be changed to varying

PRMan Geometric Pipeline



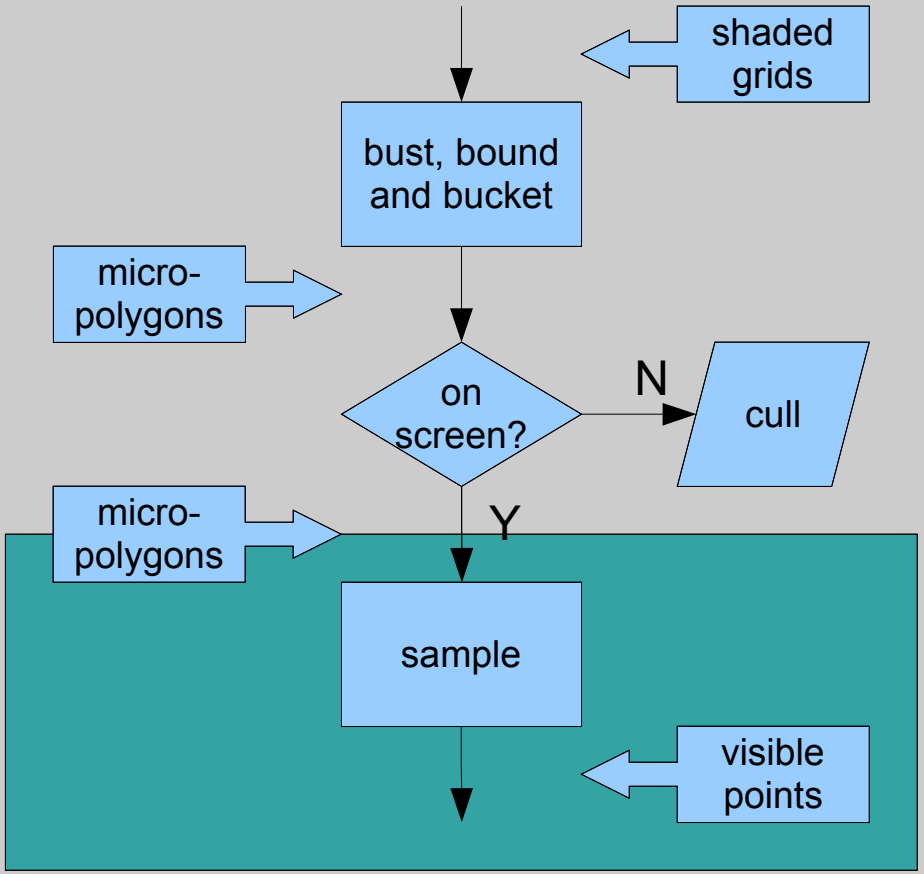
PRMan Geometric Pipeline (cont)

- Grid is sent through miniature version of splitting loop
  - *busted* into individual micro-polygons,
  - *bound*,
  - *bucketed*,
  - and possibly *culled*

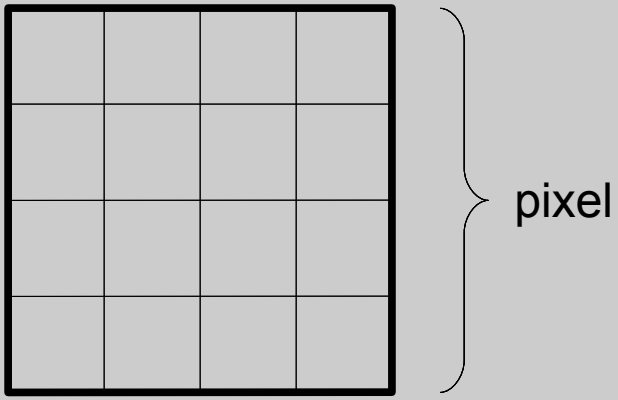


PRMan Geometric Pipeline (cont)

- Sample micropolygons
  - Pixel divided into a number of sub-pixels determined by *pixel samples*

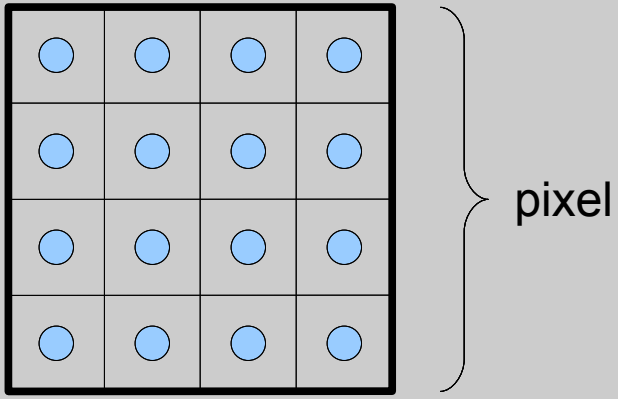


Pixel Samples



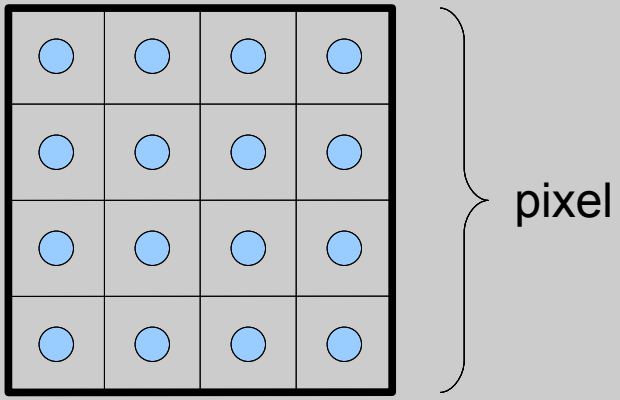
4x4 pixel samples

Pixel Samples

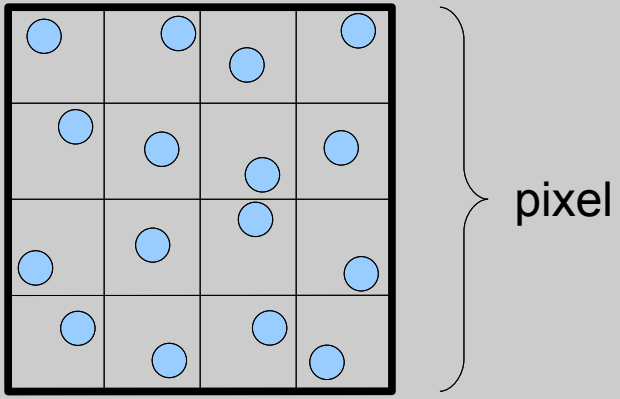


4x4 pixel samples

Pixel Samples



Jitter off



Jitter on

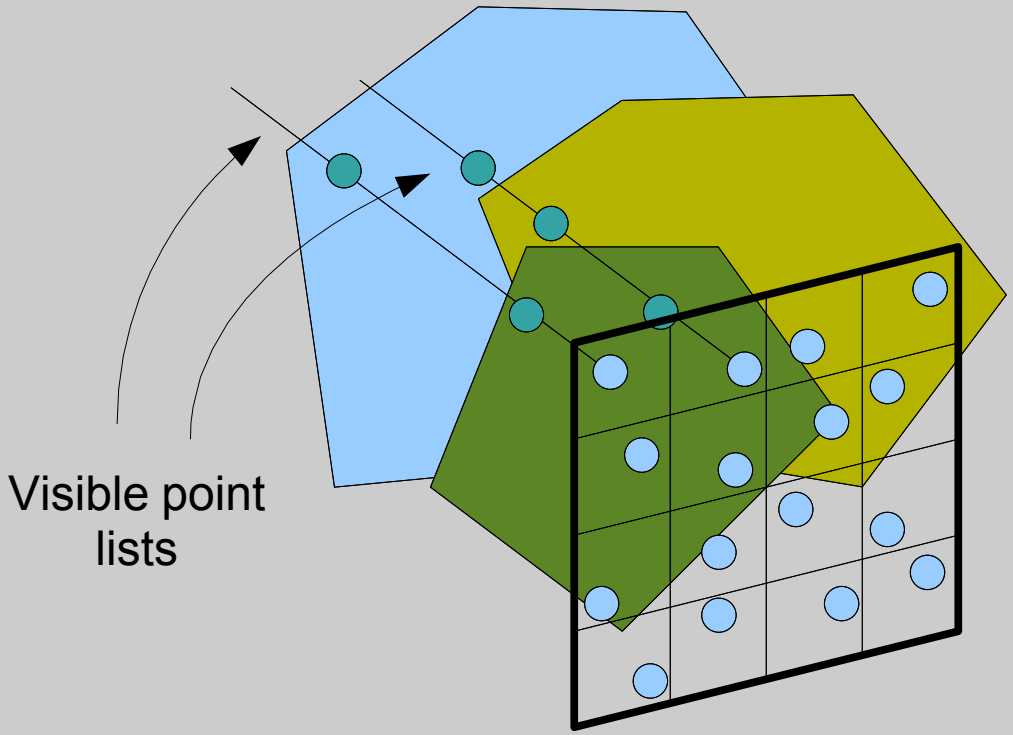
Pixel Samples

- RIB examples:

```
PixelSamples xsamples ysamples
PixelSamples 8 8

Hider "hidden" "jitter" [1]
```

Pixel Sampling

- Color, opacity, and depth of micropolygons recorded in *visible point lists*
- Bucket coverage updated

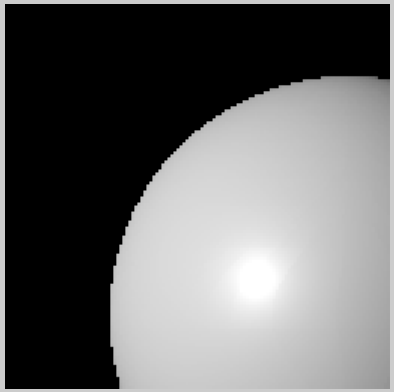


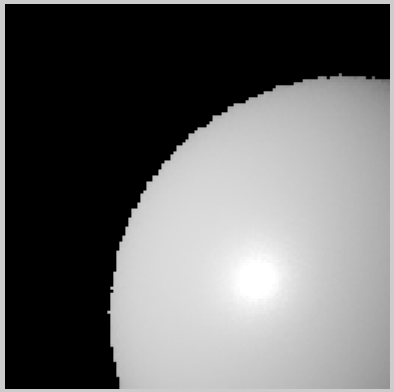
Jitter

Jitter off

Jitter on

Pixel Samples 1x1



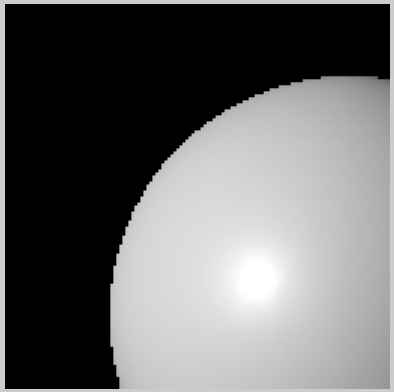


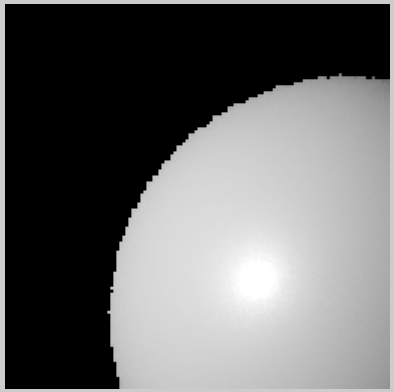
Jitter

Jitter off

Jitter on

Pixel Samples 1x1





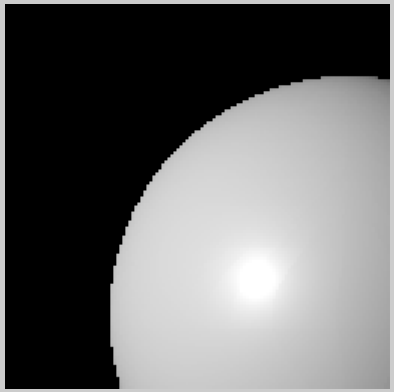
Pixel Samples 8x8

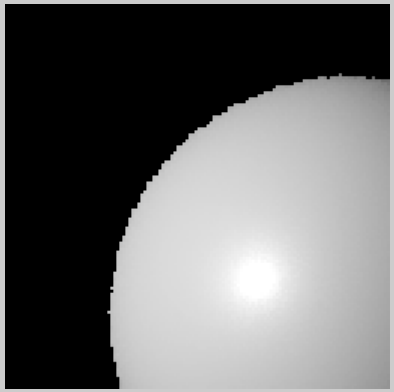
Jitter

Jitter off

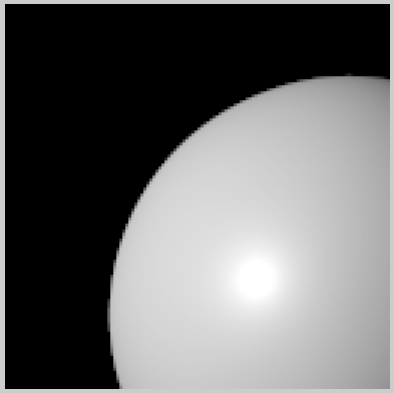
Jitter on

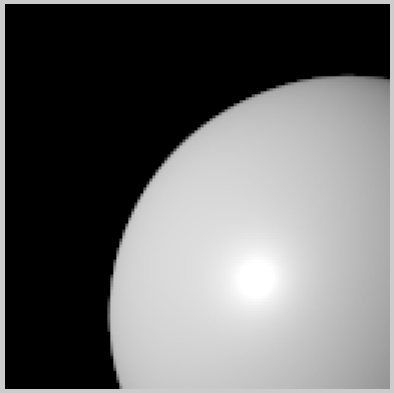
Pixel Samples 1x1



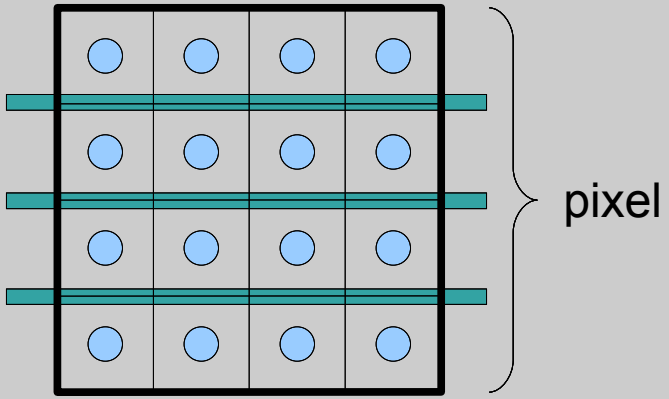


Pixel Samples 8x8

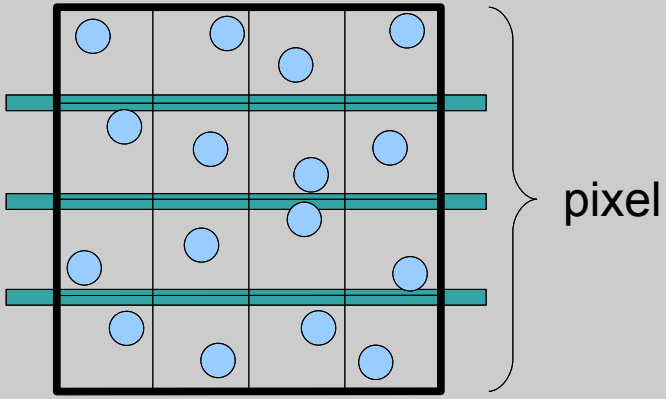




Jitter

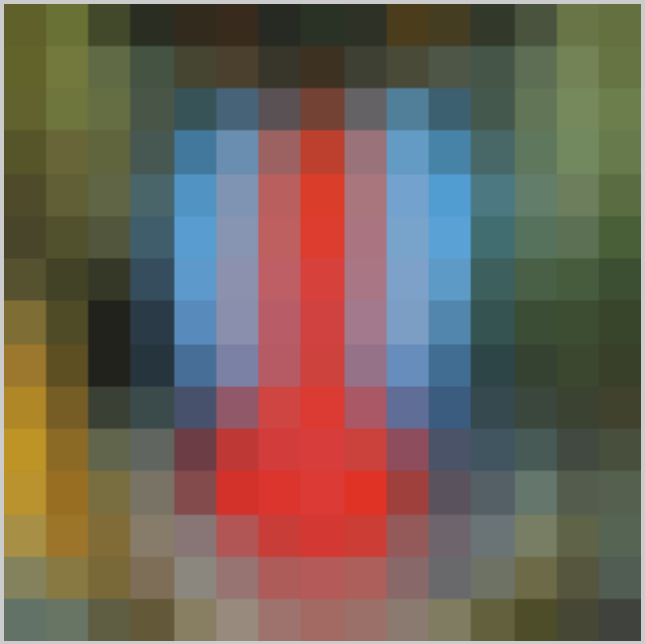


Jitter off



Jitter on

Shading Interpolation



constant

smooth

- Smooth shading requires pixel samples:

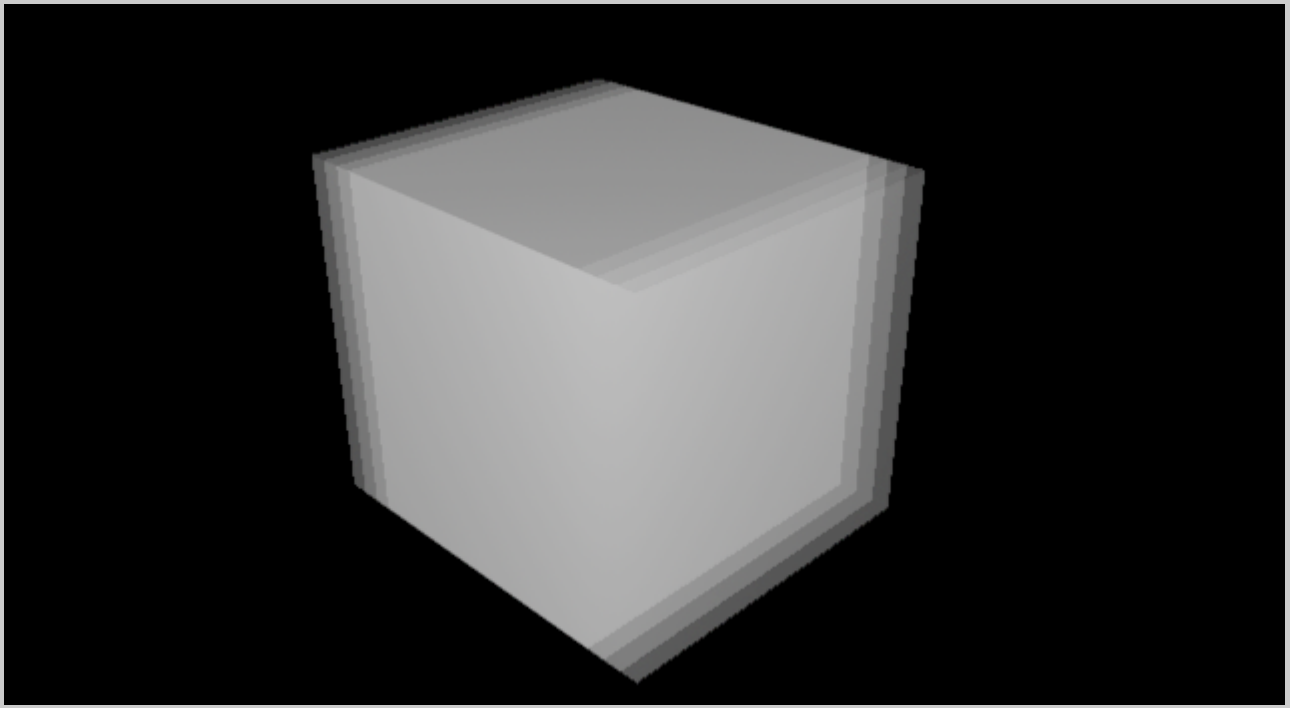
  2 / sqrt(shading rate)

Pixel Sampling

- Doesn’t significantly affect render time or memory
  - Sampling already existing shaded points
  - No additional dicing or shading performed
- Determines quality
  - Geometric aliasing
  - Motion blur

Motion Blur

- Each sample point is assigned a unique sample time

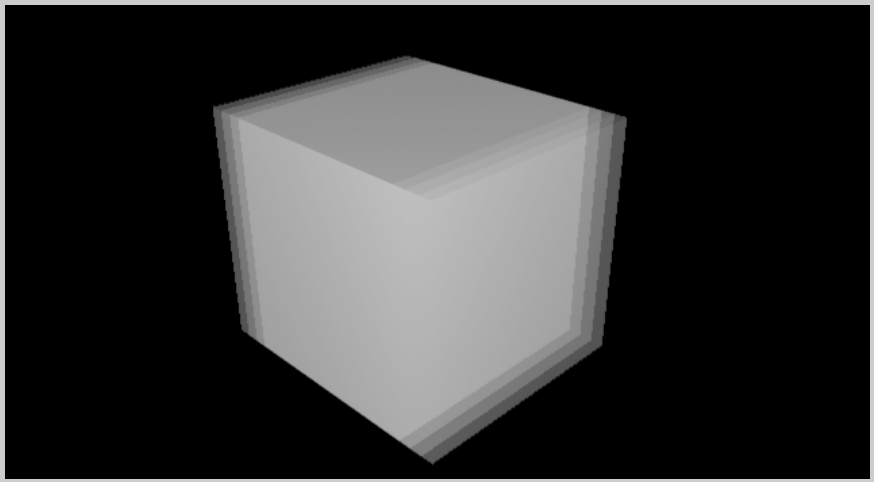


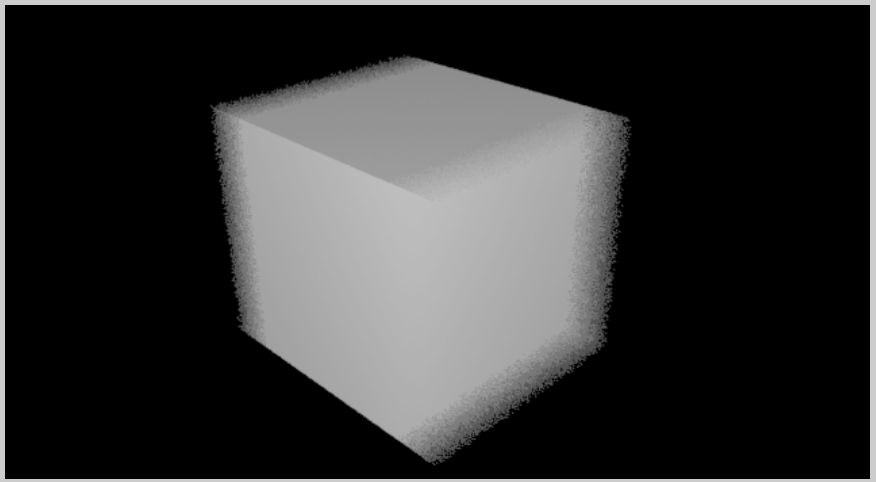
Motion Blur

Jitter off

Jitter on

Pixel Samples 2x2



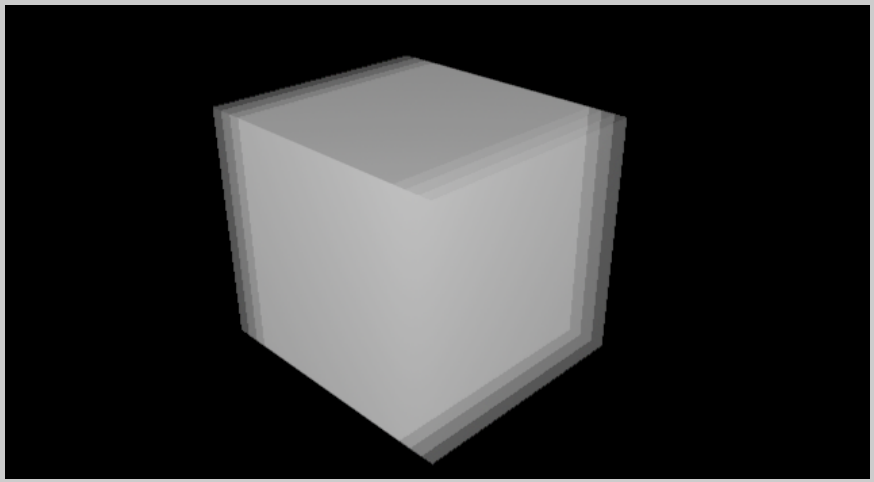


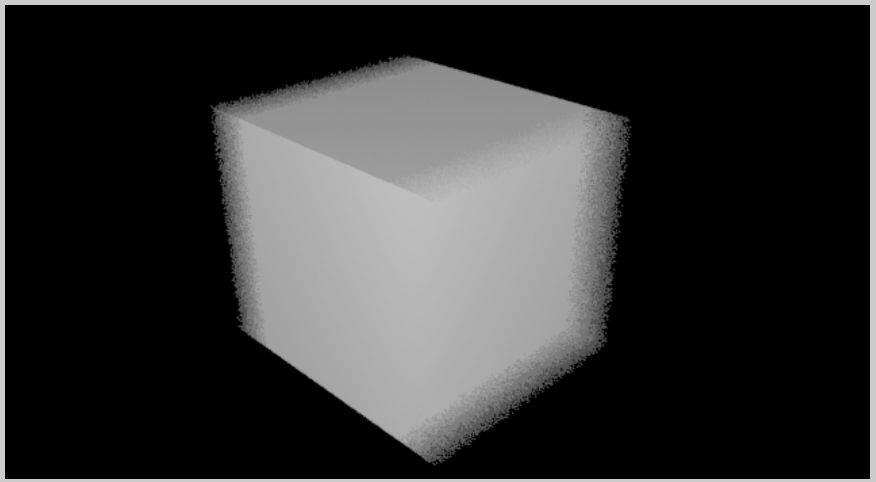
Motion Blur

Jitter off

Jitter on

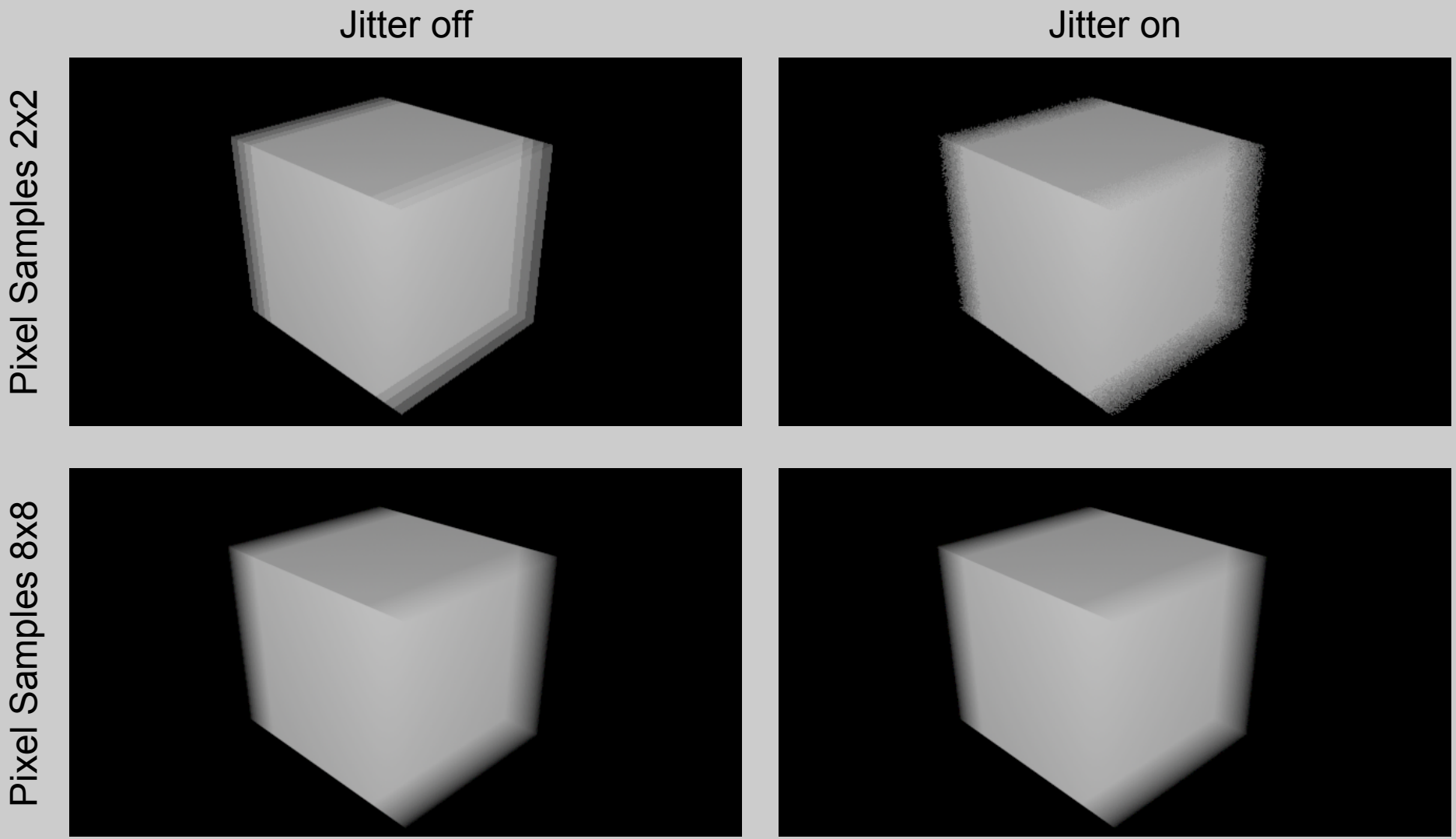
Pixel Samples 2x2



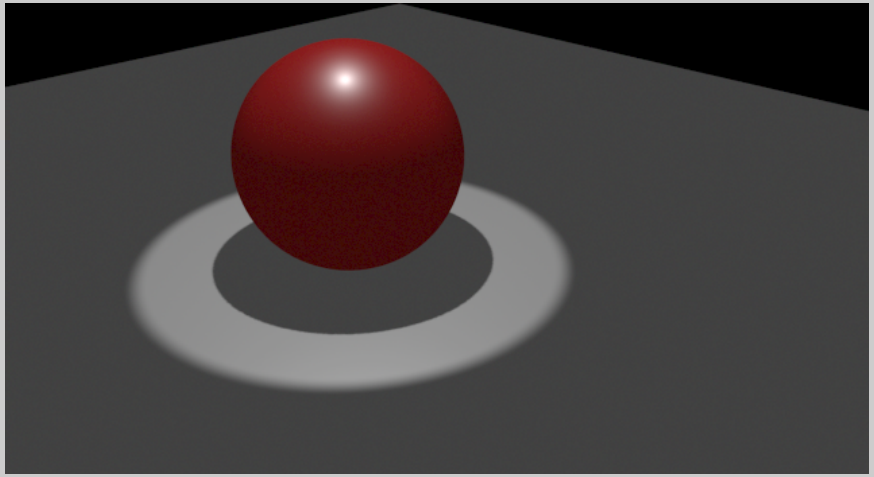


Pixel Samples 8x8

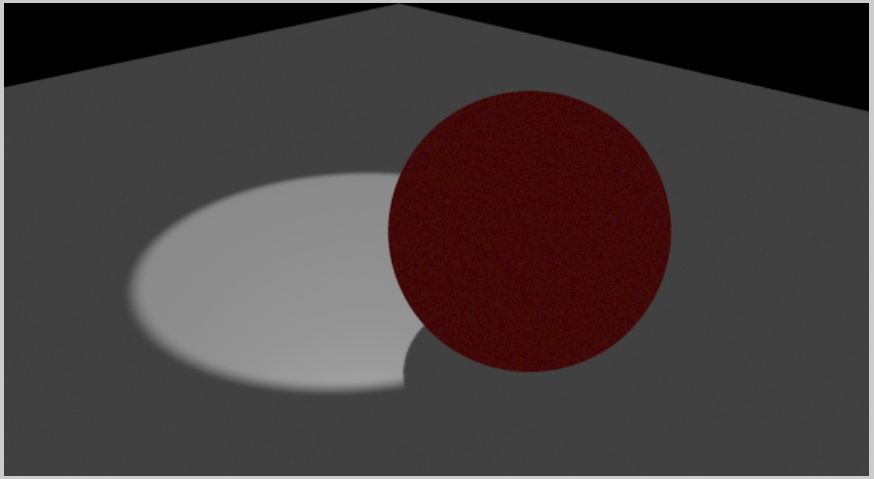
Motion Blur



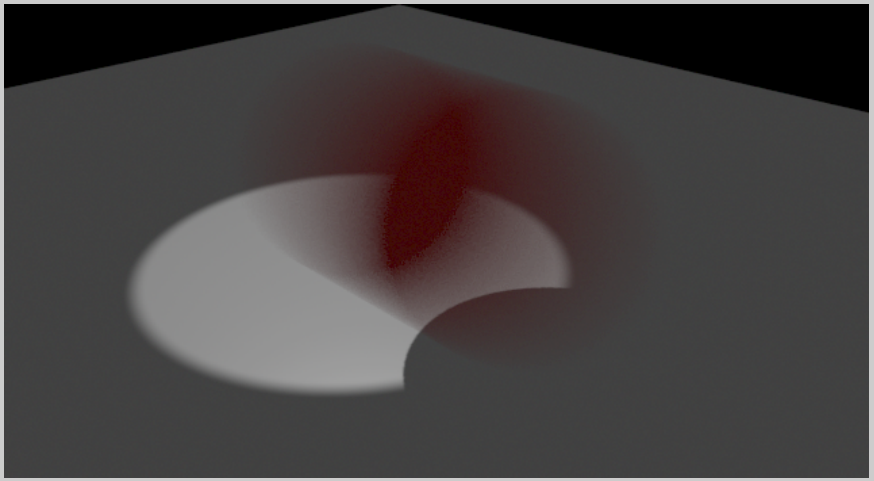
Motion Blur



Frame 1, no blur

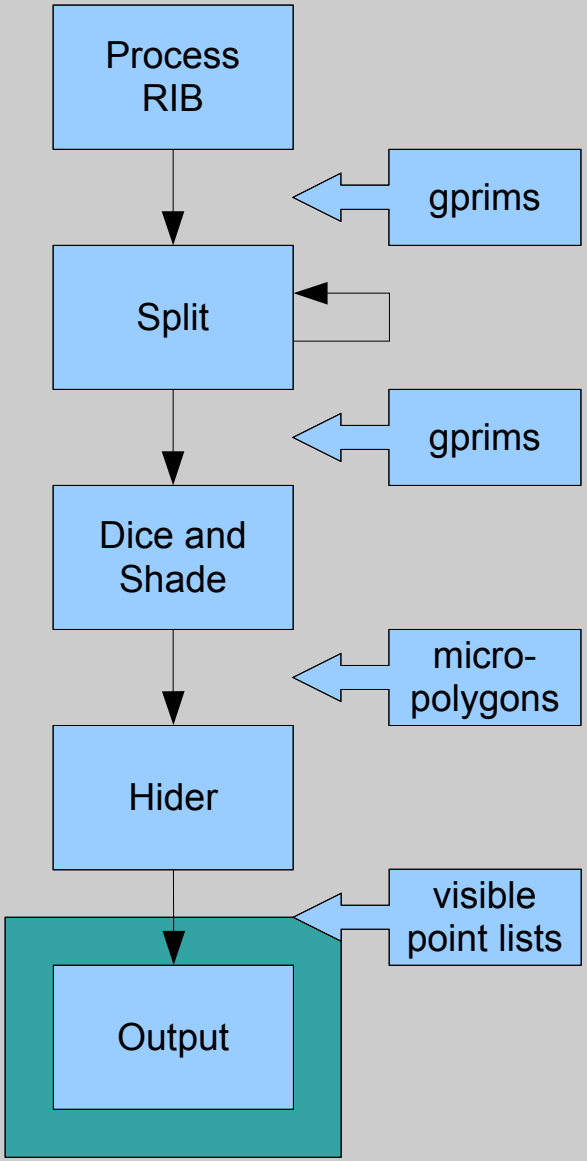


Frame 2, no blur



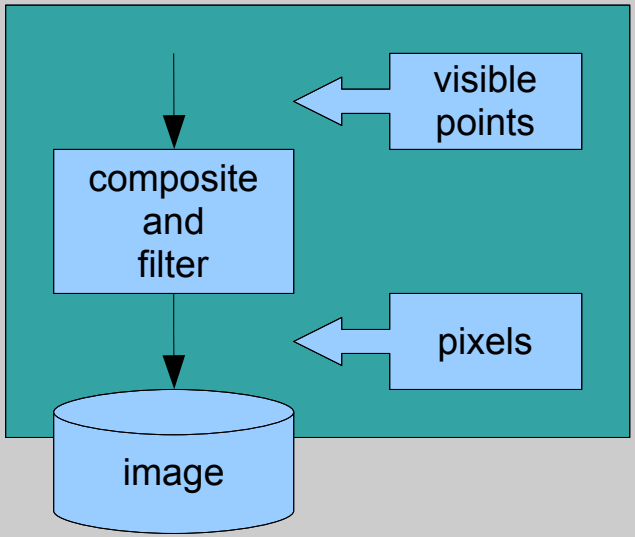
Frame 2, backwards blur

PRMan Geometric Pipeline



PRMan Geometric Pipeline (cont)

- Visible point lists composited together
- Sample points blended together with a *filter*
  - 2D operation on subpixels
  - Generally spans multiple pixels
  - Same as *Shake*



Pixel Filter

- RIB example:

```
PixelFilter type xwidth ywidth
PixelFilter "mitchell" 4 4
```

PRMan Geometric Pipeline (cont)

- Results stored in output image

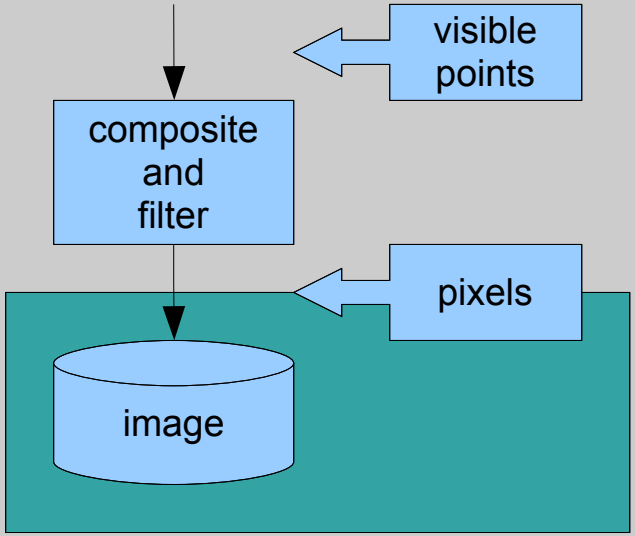


Image Output

- Quantization:

```
Quantize type one min max dither
Quantize "rgba" 255 0 255 0.5
```

- Output:

```
Display name type mode
Display "file.tif" "tiff" "rgba"
```

Conclusion

- Goal of pipeline to optimize in:
  - Memory
  - Speed

Memory Considerations

- Basic pipeline very memory efficient
  - Memory usage from:
    - Grids
    - Micropolygons
    - Visible point lists
  - Grids and micropolygons discarded as soon as possible (ideally, as soon as processed)
  - Visible point lists only kept long enough to process a bucket, not entire image

Memory Considerations

- Shading
  - Memory for code and variables in shaders
  - Transparency affects occlusion culling

Speed Considerations

- Geometric primitives
- Shading rate
- Transparency

The End

- Of Part 1 of 2.