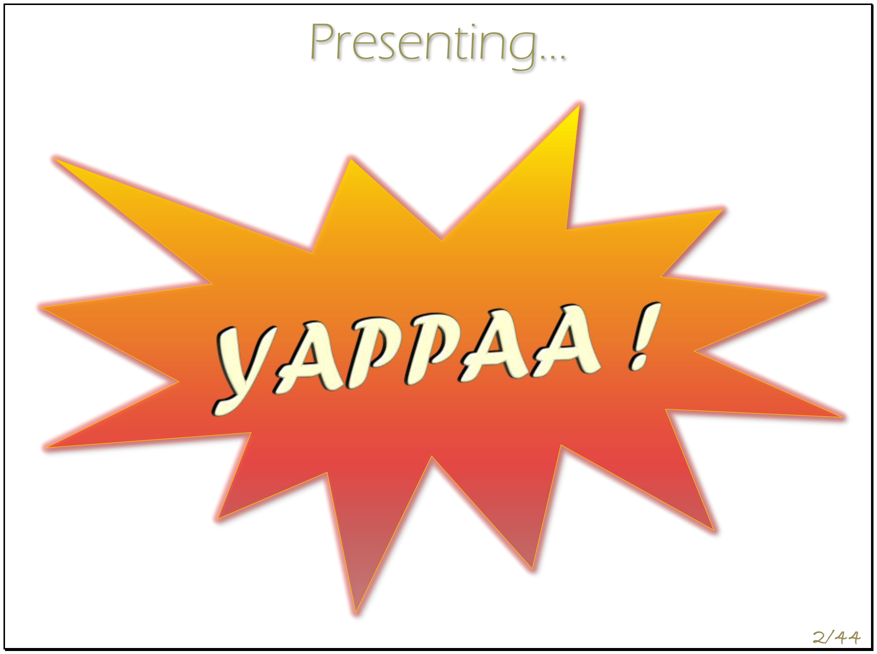
Reducing Aliasing Artifacts through Resampling

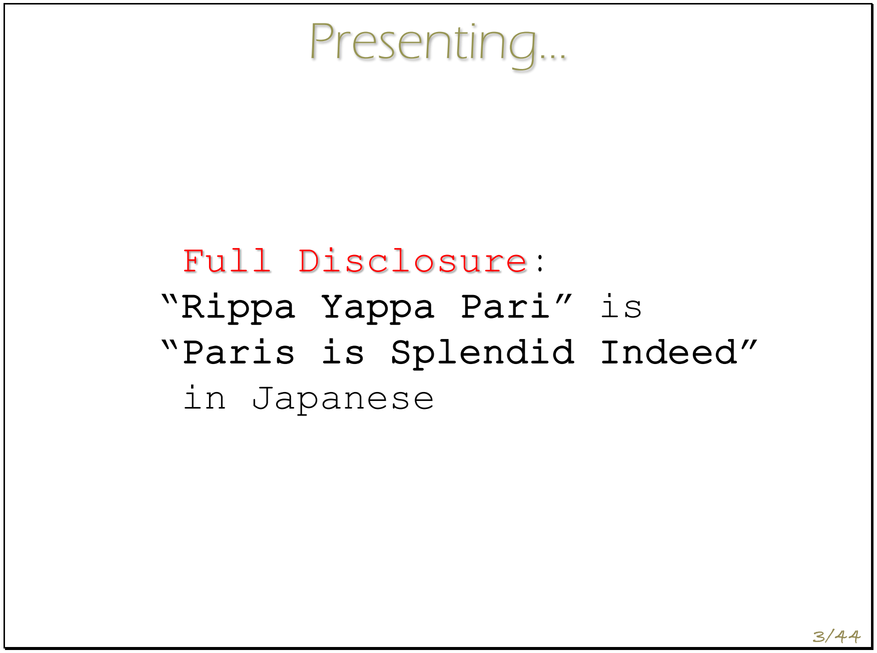
Alexander Reshetov

Slide 2


Presenting…

I will be presenting… Yet Another Post-Processing Antialiasing Algorithm.
A lot of such algorithms were proposed in recent years, so not only I will have to explain what I did, but also is why.

Slide 3



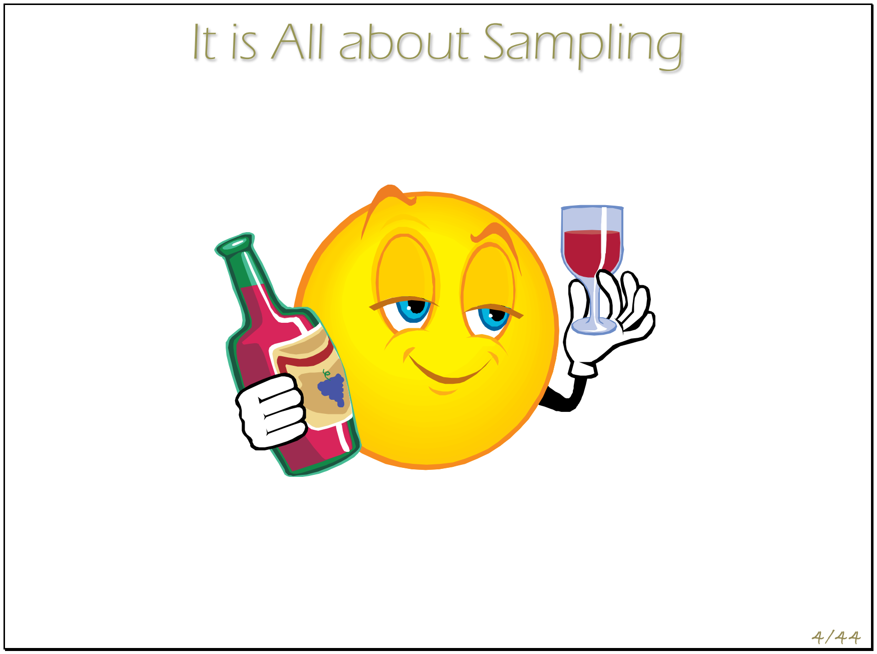
Presenting…

Full Disclosure:
“**Rippa Yappa Pari**” is
“**Paris is Splendid Indeed**”
in Japanese

3/44

In the interest of full disclosure, “Rippa Yappa Pari” is “Paris is Splendid Indeed” in Japanese and you could even find this song on YouTube.

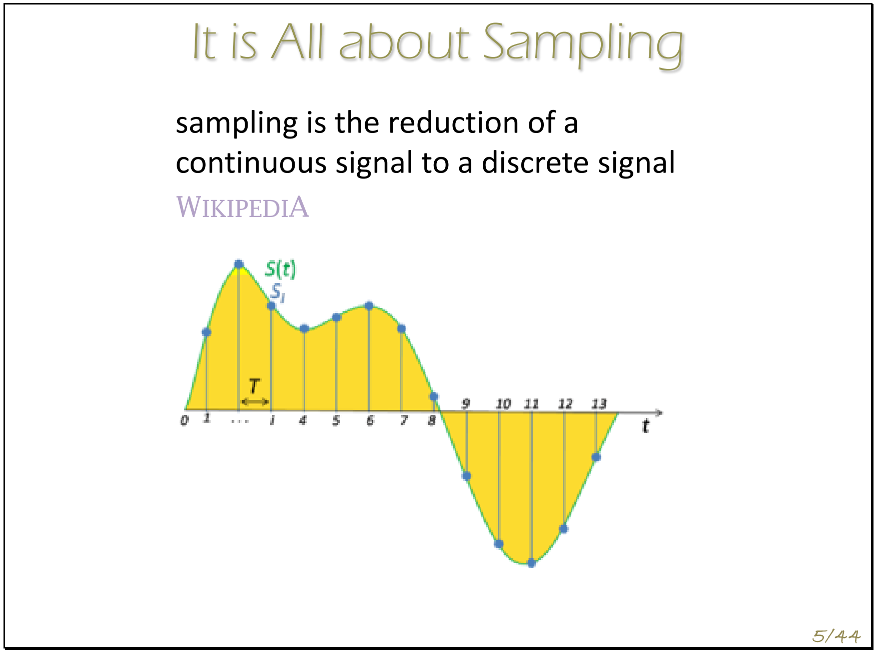
Slide 4



It is All about Sampling

Unless we're talking about some fancy analytical methods, rendering is all about sampling.
I am not talking about this one, but one you could read about in Wikipedia.

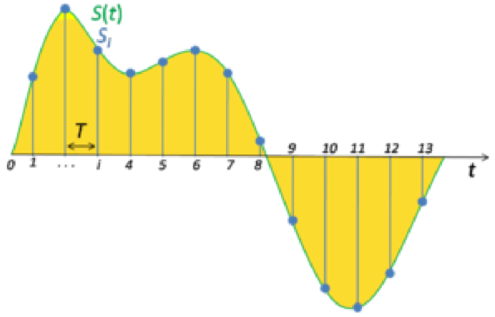
Slide 5



It is All about Sampling

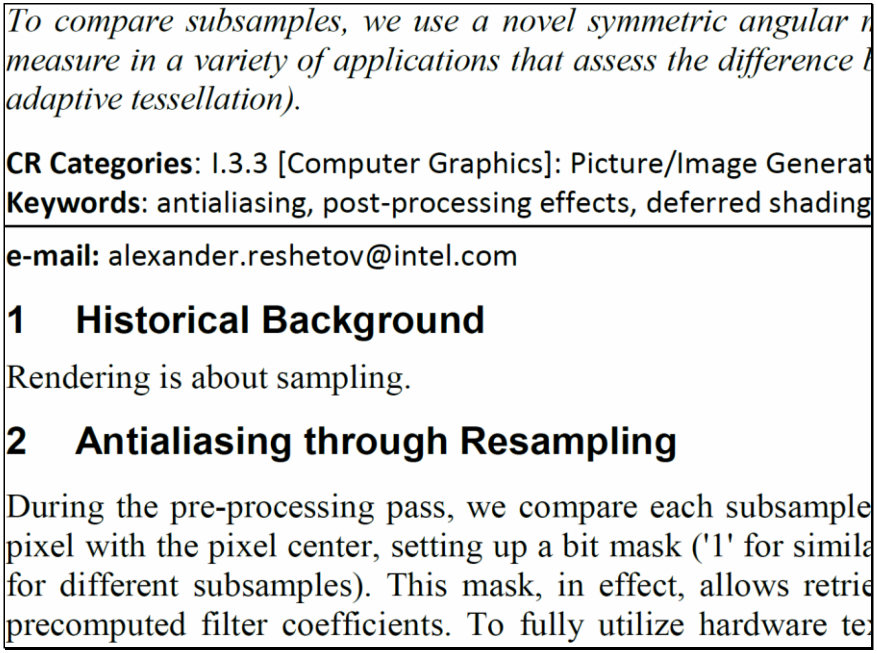
sampling is the reduction of a continuous signal to a discrete signal

WIKIPEDIA



There are 2 reasons for this:
- Target devices are discrete
- It just makes life easier (similar to wine tasting)

Slide 6


To compare subsamples, we use a novel symmetric angular n
measure in a variety of applications that assess the difference l
adaptive tessellation).

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generat
Keywords: antialiasing, post-processing effects, deferred shading

e-mail: alexander.reshetov@intel.com

1 Historical Background

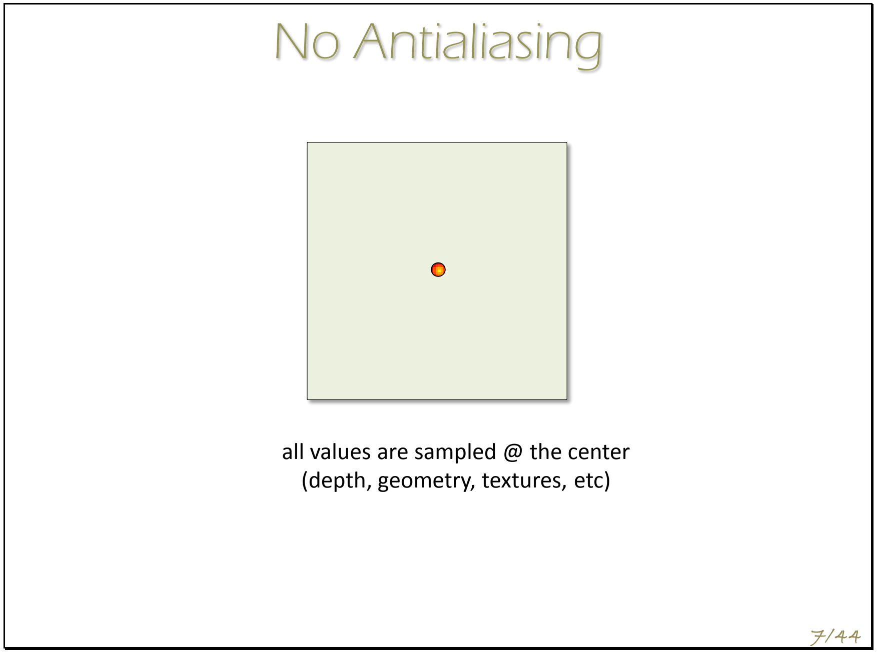
Rendering is about sampling.

2 Antialiasing through Resampling

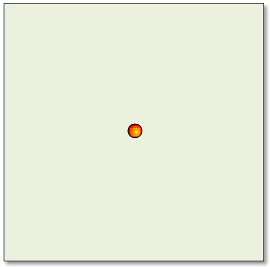
During the pre-processing pass, we compare each subsample
pixel with the pixel center, setting up a bit mask ('1' for simila
for different subsamples). This mask, in effect, allows retrie
precomputed filter coefficients. To fully utilize hardware te

So, my original idea for the paper was just to say it in the introduction, and then proceed with the main idea presentation.
But, probably, it would've been too extreme. So let's talk about prior art now.

Slide 7



No Antialiasing



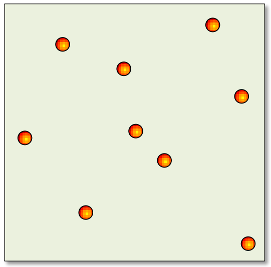
all values are sampled @ the center
(depth, geometry, textures, etc)

7/44

I will introduce key ideas of the new approach after briefly describing the existing techniques and their limitations.
Sampling everything once per pixel is, obviously, the fastest way, but quality will suffer.

Slide 8

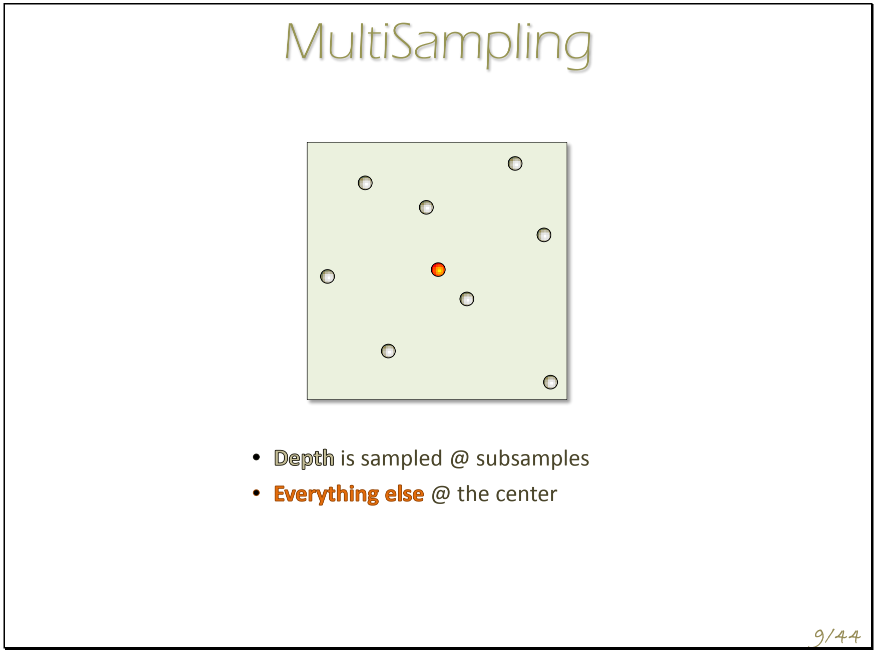
Supersampling



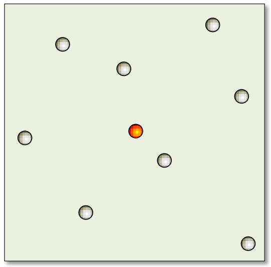
values are sampled @ multiple locations
(and resulting colors are averaged)

Au contraire, supersampling allows the best quality but rather expensive.

Slide 9



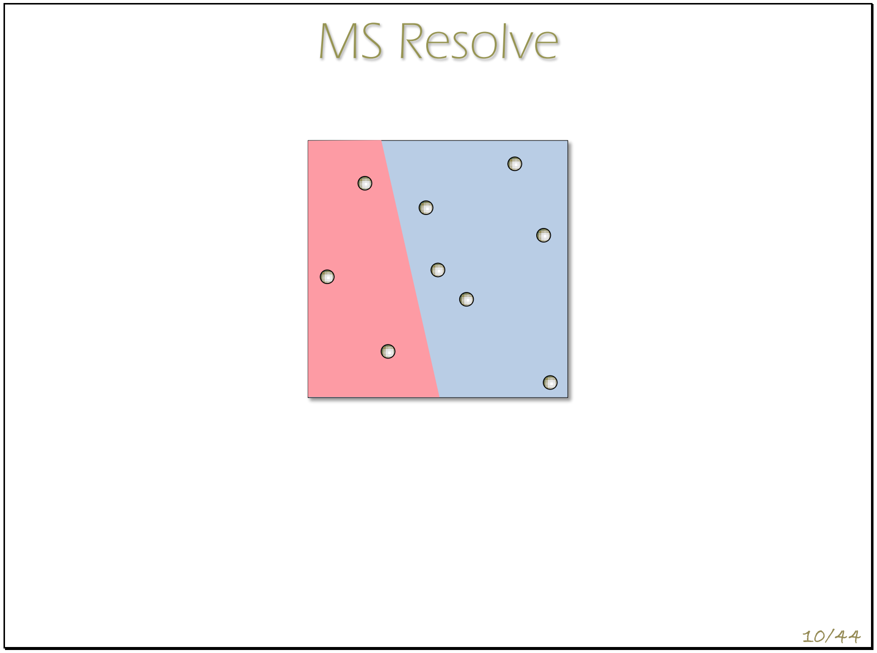
MultiSampling



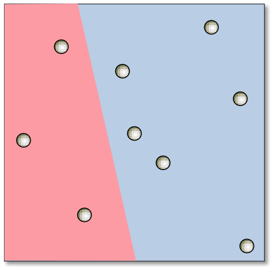
- Depth is sampled @ subsamples
- Everything else @ the center

Multisampling is an attempt to get all the advantages of supersampling at a fraction of the cost. It works nice for direct shading pipeline when the final pixel color is resolved through hardware acceleration.

Slide 10

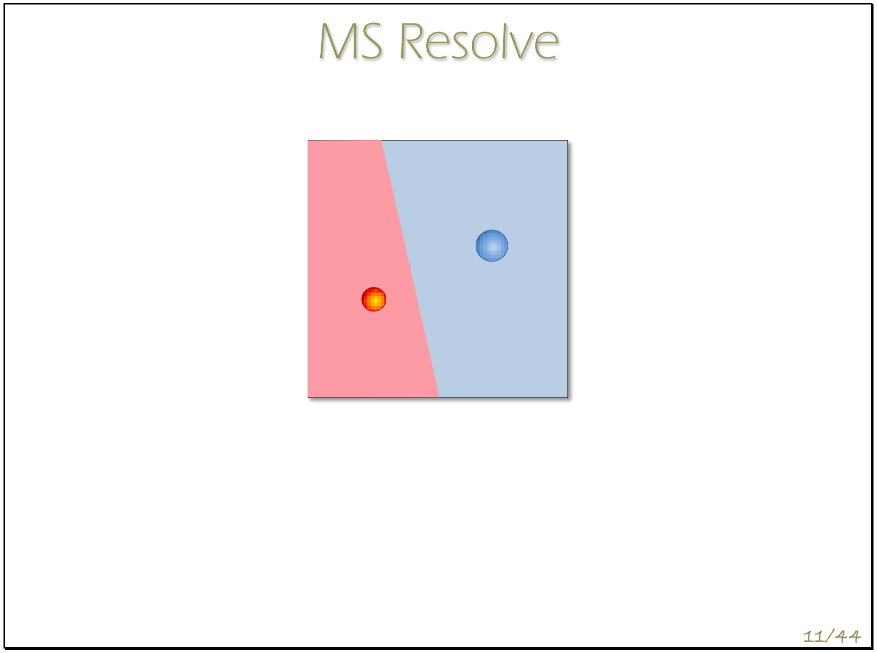


MS Resolve

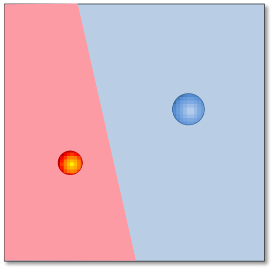


So if there are only 2 primitives overlapping a pixel, subsamples, obviously, will be split into 2 groups and

Slide 11

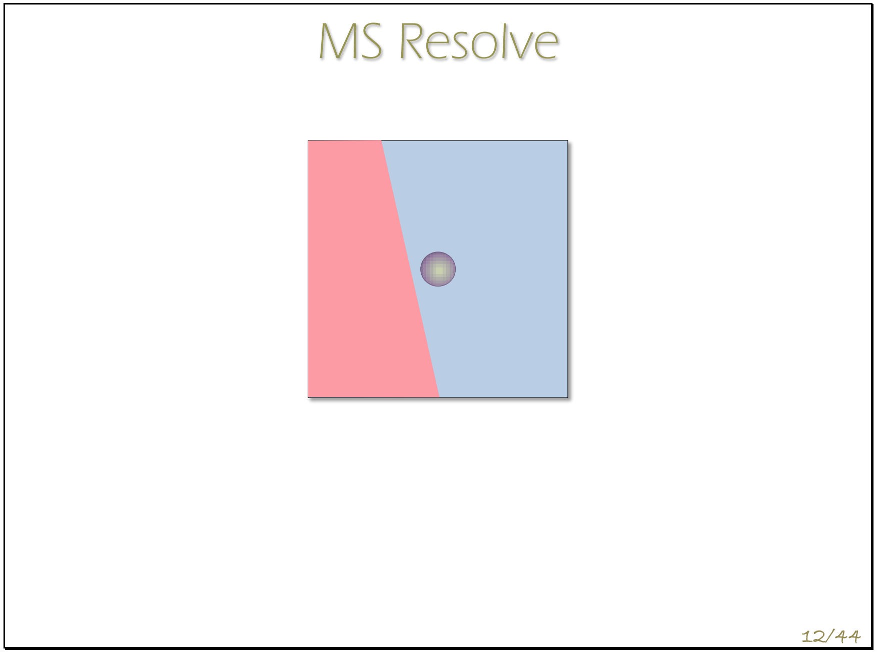


MS Resolve

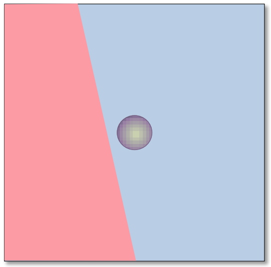


shading will be done only once per group.

Slide 12

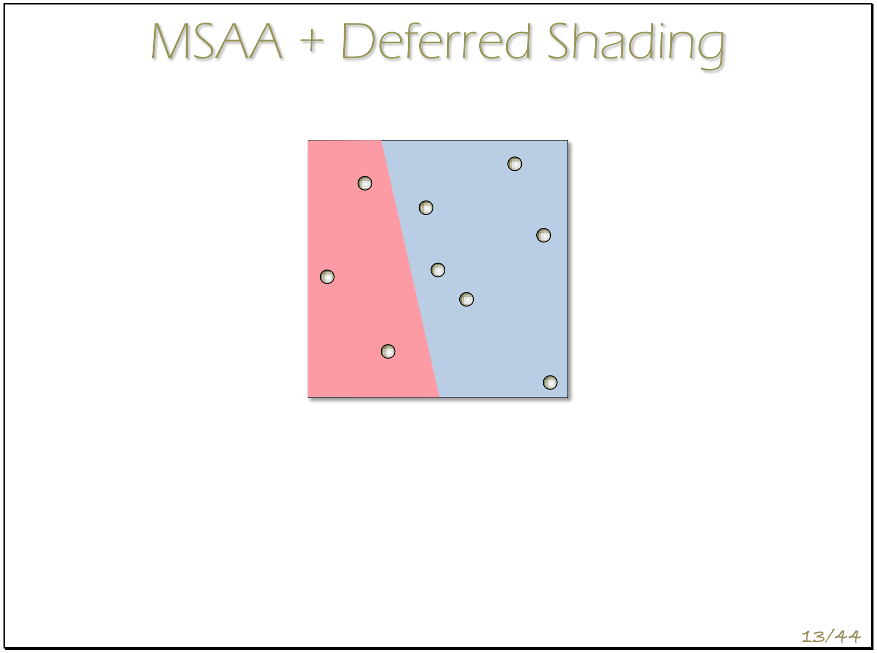


MS Resolve

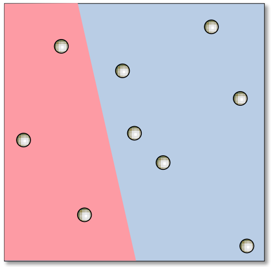


The final color will be computed by blending these 2 colors with weights proportional to coverage.

Slide 13

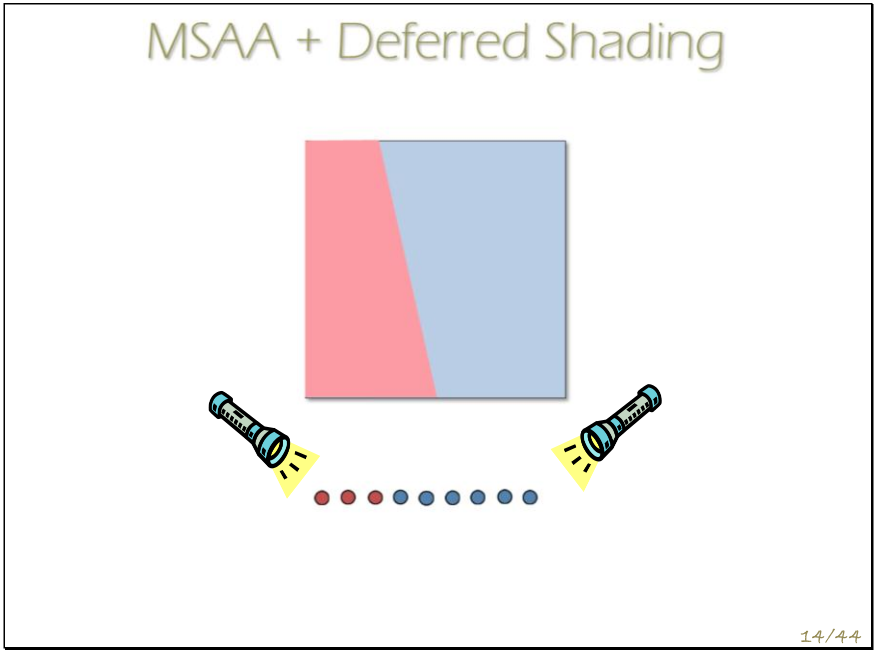


MSAA + Deferred Shading

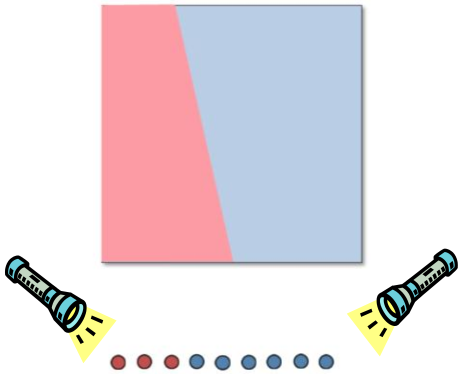


For deferred shading though all subsamples have to be explicitly written to output buffer for later processing, so it is not very efficient.

Slide 14

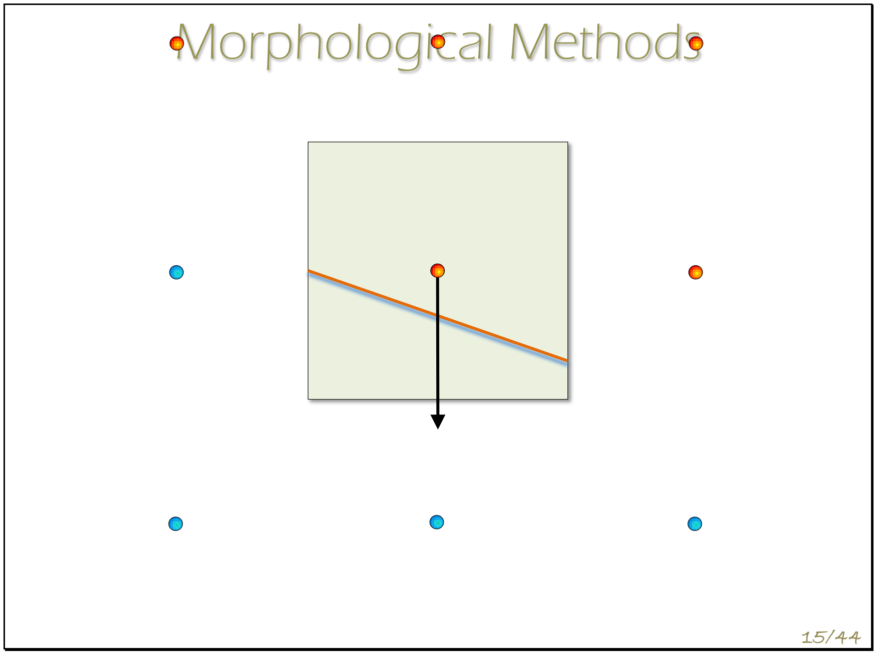


MSAA + Deferred Shading



The reason for this is that resolve and lighting computations are noncommutative.

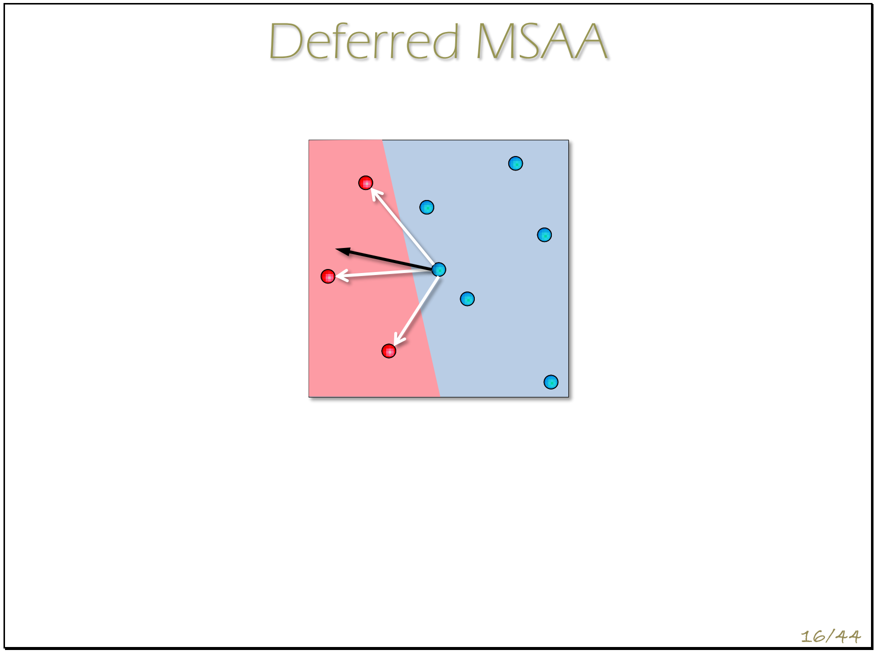
Slide 15



Morphological Methods

In morphological methods colors are computed once per pixel, and additional knowledge is obtained from analyzing non-local neighborhood, allowing to reconstruct plausible object silhouettes and then improve the quality through linear interpolation.
Accordingly, such methods are a good match for deferred shading, and, actually, for any other shading, since they are independent from the rest of the pipeline.
For this reason, such methods become popular in last few years. Who would knew?
Still, there are some inherent quality limitations, especially in temporal domain.

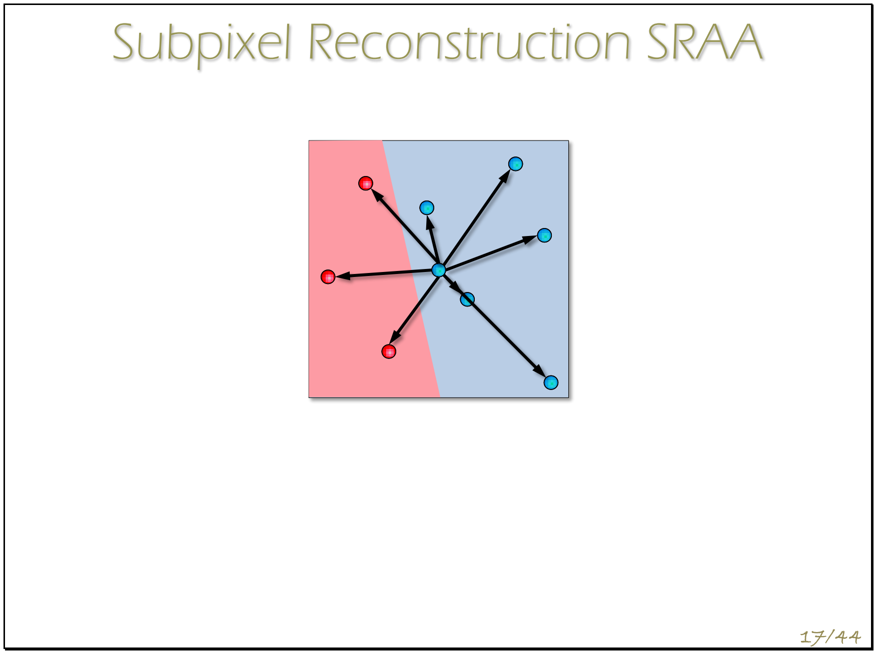
Slide 16



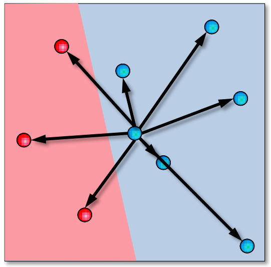
Deferred MSAA

It is possible to improve quality by considering additional geometric samples, while still executing expensive shading computations once per pixel.
The technique I describe today falls into this category.
Another approach to this effect is deferred MSAA, proposed by Matt Pettineo. It works by averaging all subsamples which depth is significantly different from the depth of the central sample and then bilinearily interpolating the image in the direction of the average vector.
It is a very simple approach, but it doesn't scale well with number of samples and it is possible to come with a better direction of the resampling vector.

Slide 17

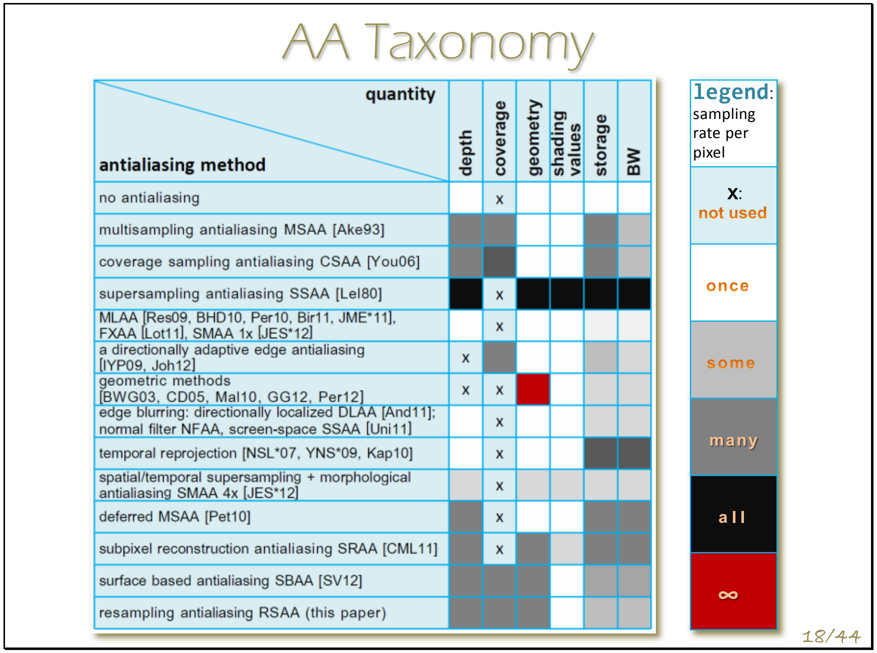


Subpixel Reconstruction SRAA



In sub-pixel reconstruction, all processing is done implicitly, applying cross-bilateral filtering to upsample per-pixel colors from geometric data.
It is a very interesting idea, but it substitutes a linear block filter with non-linear one in which coefficients are proportional to geometric difference between subsamples, expressed numerically.

Slide 18



AA Taxonomy

antialiasing method \ quantity	depth	coverage	geometry	shading values	storage	BW
no antialiasing	□	x	□	□	□	□
multisampling antialiasing MSAA [Ake93]	▓	▓	□	□	▓	▒
coverage sampling antialiasing CSAA [You06]	▓	▓	□	□	▓	▒
supersampling antialiasing SSAA [Lel80]	■	x	■	■	■	■
MLAA [Res09, BHD10, Per10, Bir11, JME*11], FXAA [Lot11], SMAA 1x [JES*12]	□	x	□	□	░	░
a directionally adaptive edge antialiasing [IYP09, Joh12]	x	▓	□	□	▒	░
geometric methods [BWG03, CD05, Mal10, GG12, Per12]	x	x	🟥	□	░	░
edge blurring: directionally localized DLAA [And11]; normal filter NFAA, screen-space SSAA [Uni11]	□	x	□	□	░	░
temporal reprojection [NSL*07, YNS*09, Kap10]	□	x	□	□	▓	▓
spatial/temporal supersampling + morphological antialiasing SMAA 4x [JES*12]	░	x	░	░	░	░
deferred MSAA [Pet10]	▓	x	□	□	▓	▓
subpixel reconstruction antialiasing SRAA [CML11]	▓	x	▓	░	▓	▓
surface based antialiasing SBAA [SV12]	▓	▓	▓	□	▒	▒
resampling antialiasing RSAA (this paper)	▓	▓	▓	□	▒	▒

legend: sampling rate per pixel

x:	not used
□	once
▒	some
▓	many
■	all
🟥	∞

18/44

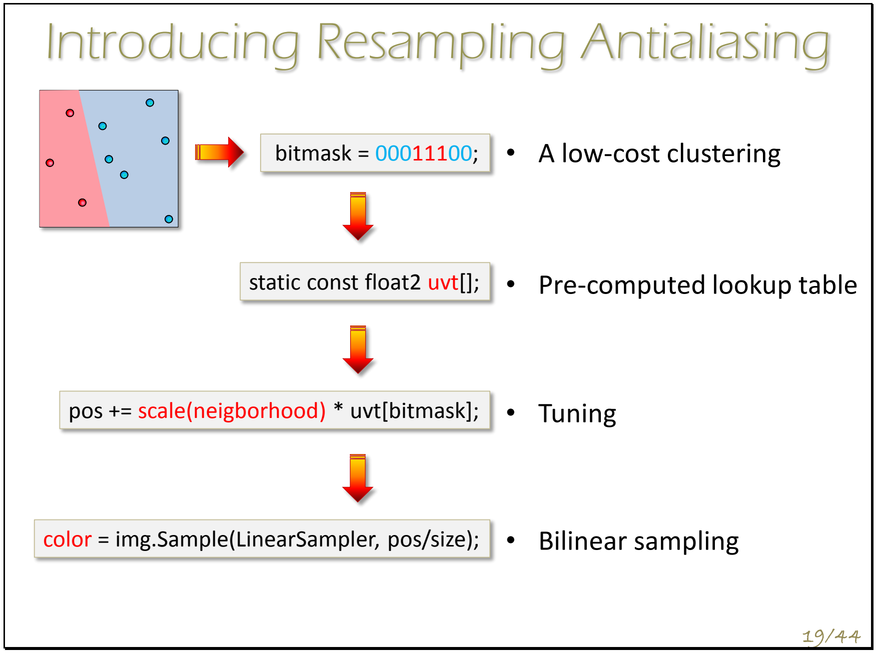
You could read more about various post-processing antialiasing methods in the last year's Siggraph course. I have also included all those algorithms, and a few new ones, into this convoluted table. To my surprise, reviewers did not object to this table excessively, so it is now official.

Please don't try to read too much into this table: it is more like an art rather than a science.

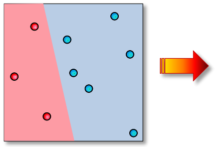
Basically, I am using shades of gray to indicate typical sampling rates for a number of entities required for antialiasing computa-tions.

A white color stands for a single sample per pixel and then it scales up to a black color for many subsamples.
It is not quite 50 shades of grey, but a lot.

This table just illustrates the fact that all of these values can be sampled at different rates. And it also improves the Hirsch citation index for all these papers.



Introducing Resampling Antialiasing



```
bitmask = 00011100;
```
- A low-cost clustering

```
static const float2 uvt[];
```
- Pre-computed lookup table

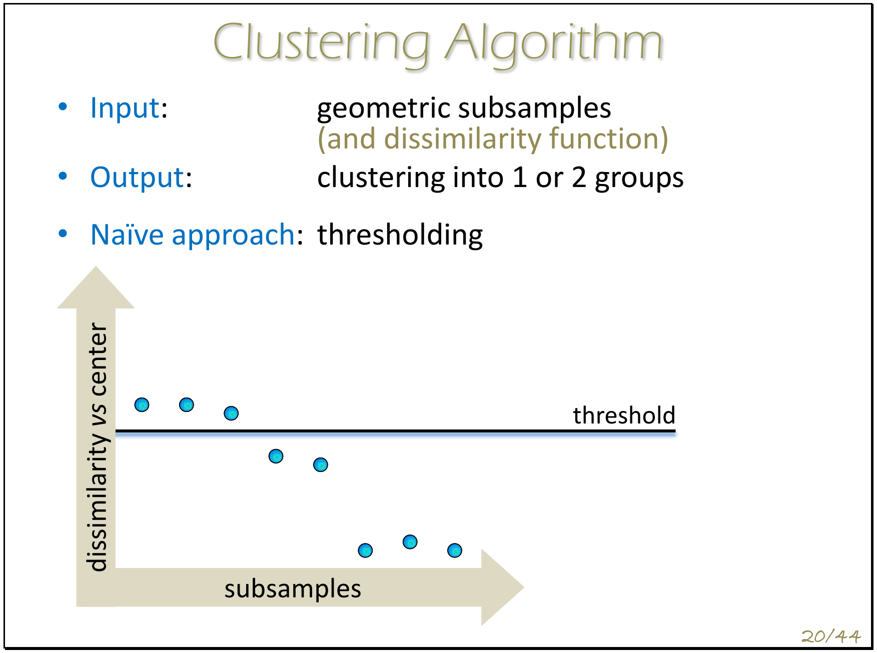
```
pos += scale(neigborhood) * uvt[bitmask];
```
- Tuning

```
color = img.Sample(LinearSampler, pos/size);
```
- Bilinear sampling

The algorithm I present today requires splitting all subsamples in pixel into 1 or 2 groups. The resulting binary mask is then used to retrieve filtering coefficients from the precomputed lookup table.
All this processing is done inside the pixel. There is also an optional tuning step, adjusting the coefficients by considering local neighborhood.
The last step of the algorithm is just bilinear interpolation.

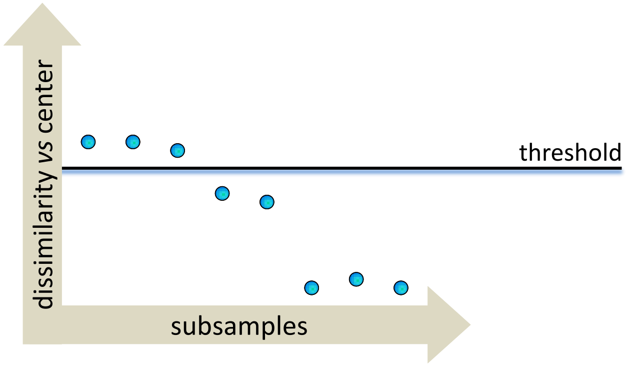
Except this, all other steps are original and I will describe them now.

Slide 20



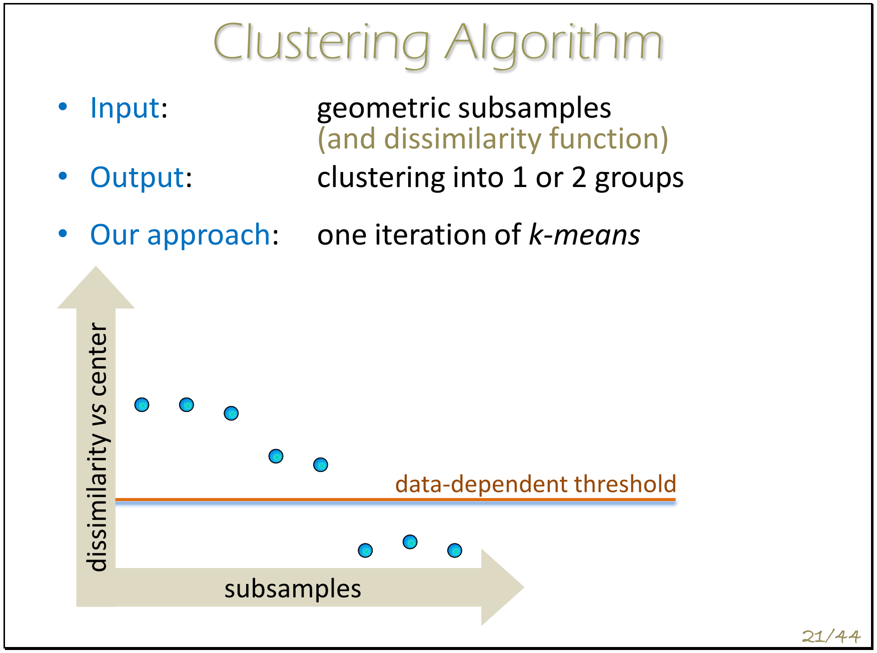
Clustering Algorithm

- Input: geometric subsamples (and dissimilarity function)
- Output: clustering into 1 or 2 groups
- Naïve approach: thresholding



The clustering is based on considering the dissimilarity between all subsamples and the center of the pixel, expressed numerically.
Naïve approach would be to just split everything by comparing these values with a predefined threshold.
It is not the best possible approach, as illustrated on this figure.

Slide 21



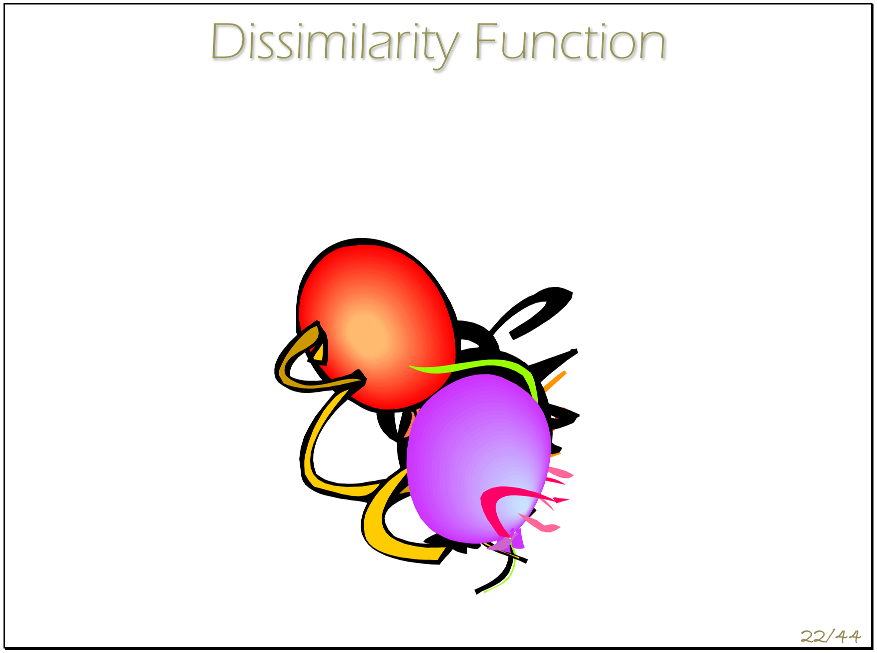
Clustering Algorithm

- Input: geometric subsamples (and dissimilarity function)
- Output: clustering into 1 or 2 groups
- Our approach: one iteration of *k-means*

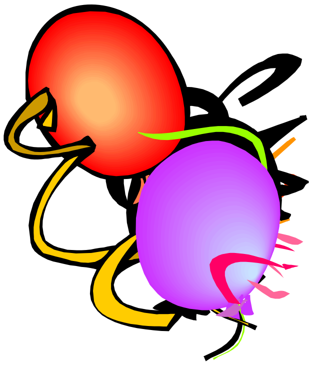
dissimilarity *vs* center
data-dependent threshold
subsamples

This is one of the most typical problems in science, and there are multiple ways to do it. One of the most widely used is *k-means*. It is good, but expensive, requiring multiple iterations.
So what we do is restrict ourselves just to one iteration.

Slide 22

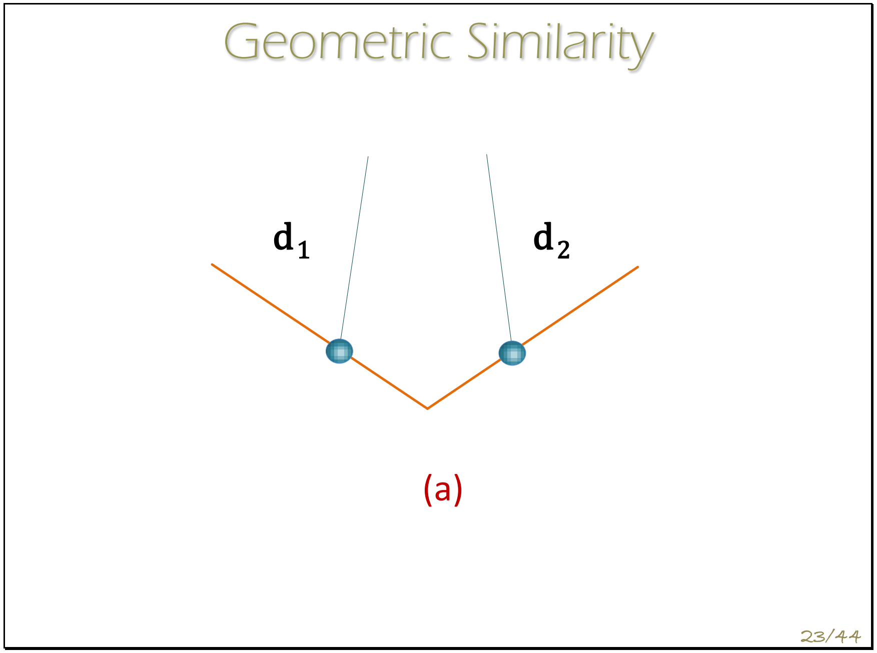


Dissimilarity Function

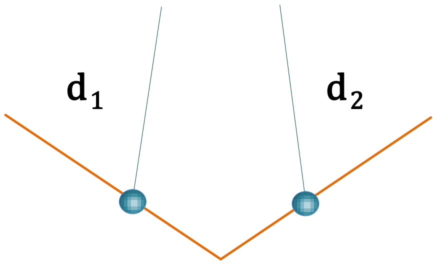


To cluster subsamples, we need a mechanism to measure dissimilarity of 2 subsamples… without using colors.

Slide 23

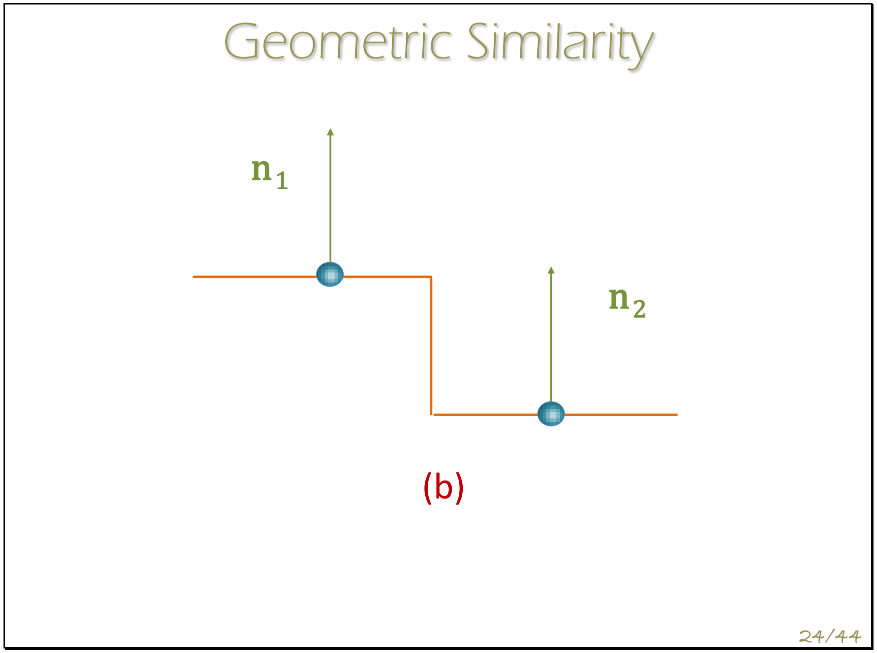


Geometric Similarity

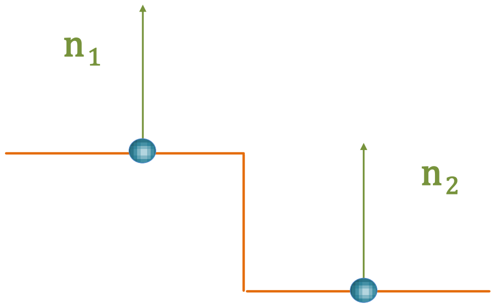


(a)

But we still can use geometry.
One popular approach is to compare depth of two samples, assuming that if depth is significantly different, subsamples belong to different objects. It is very simple, but may fail in corners, as shown on this image.



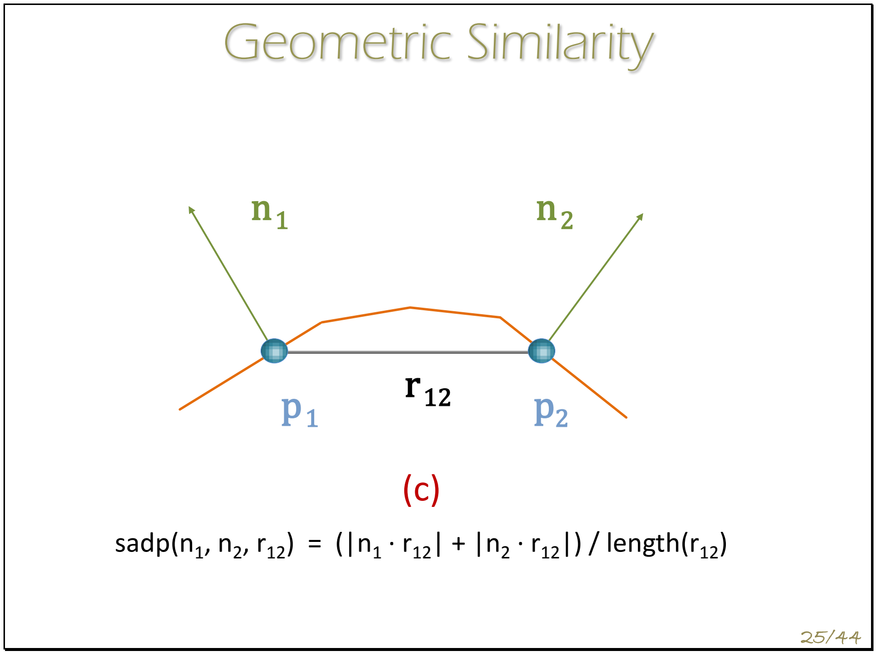
Geometric Similarity



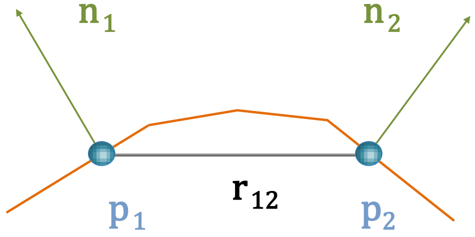
(b)

Corner detection could be resolved with a measure that uses subsample normals – for example, their dot product. However, this measure will not be able to distinguish spaced-out almost-parallel surfaces.

Slide 25



Geometric Similarity



(c)

$$\text{sadp}(n_1, n_2, r_{12}) = (|n_1 \cdot r_{12}| + |n_2 \cdot r_{12}|) / \text{length}(r_{12})$$

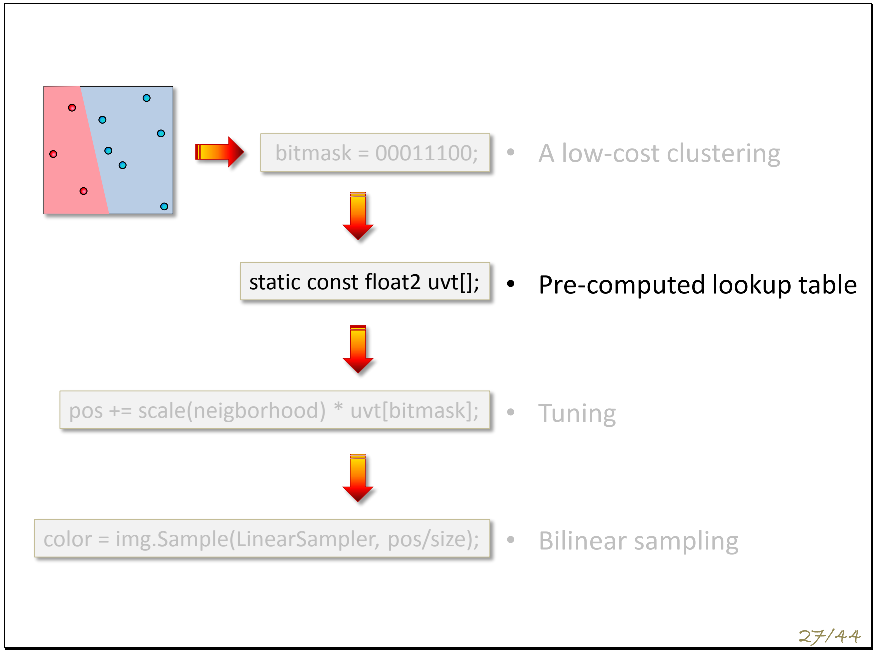
And finally a new measure which we propose. It is a sum of the absolute dot products of two normals and a normalized vector from one subsample to another. It addresses problems of other measures; it is symmetric and does not depend on camera position.

Slide 26

Sad P

We call it SADP and, since it can be used in other computer science disciplines, I created this clipart to help you remember it better.

Slide 27



```
bitmask = 00011100;
```
- A low-cost clustering

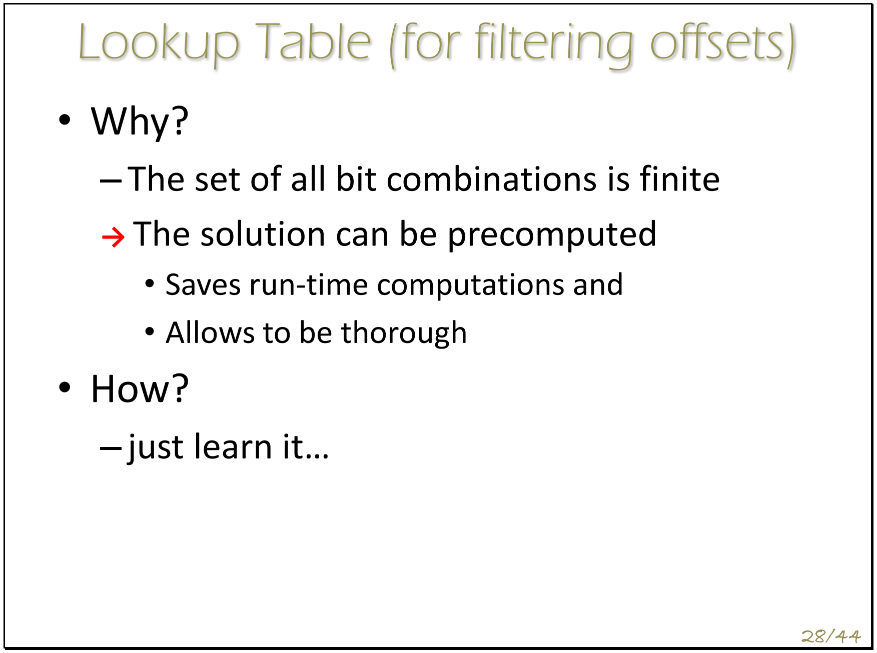
```
static const float2 uvt[];
```
- Pre-computed lookup table

```
pos += scale(neigborhood) * uvt[bitmask];
```
- Tuning

```
color = img.Sample(LinearSampler, pos/size);
```
- Bilinear sampling

27/44

Now we have an index, so we need an array to index into.



Lookup Table (for filtering offsets)

- Why?
  - The set of all bit combinations is finite

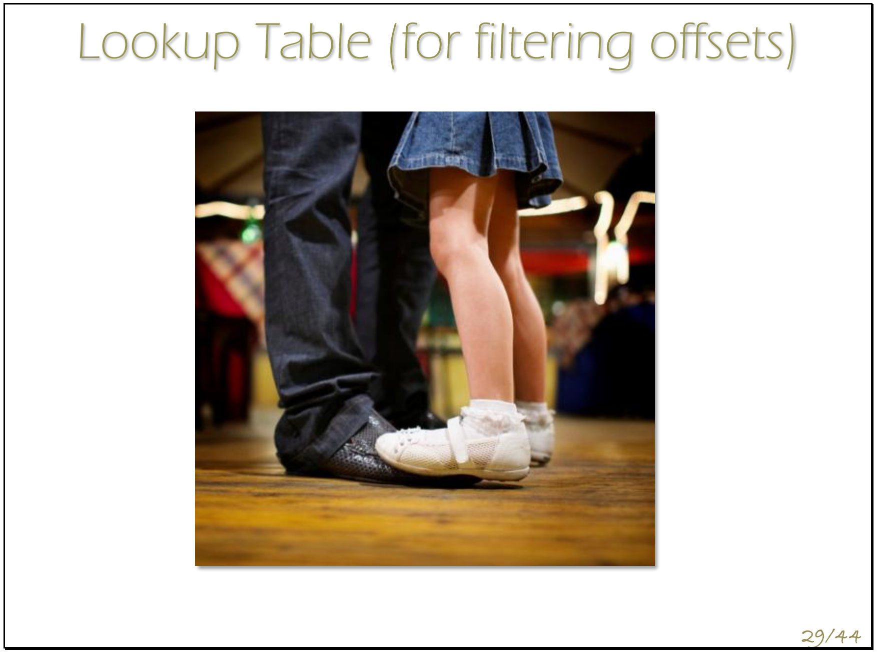
  → The solution can be precomputed
    - Saves run-time computations and
    - Allows to be thorough
- How?
  - just learn it…

28/44

The set of all bit combinations is finite, so it is possible to precompute all possible solutions. It will save run-time computations, and will allow us to be thorough.

We still don’t know how to solve this problem, but we can learn it.

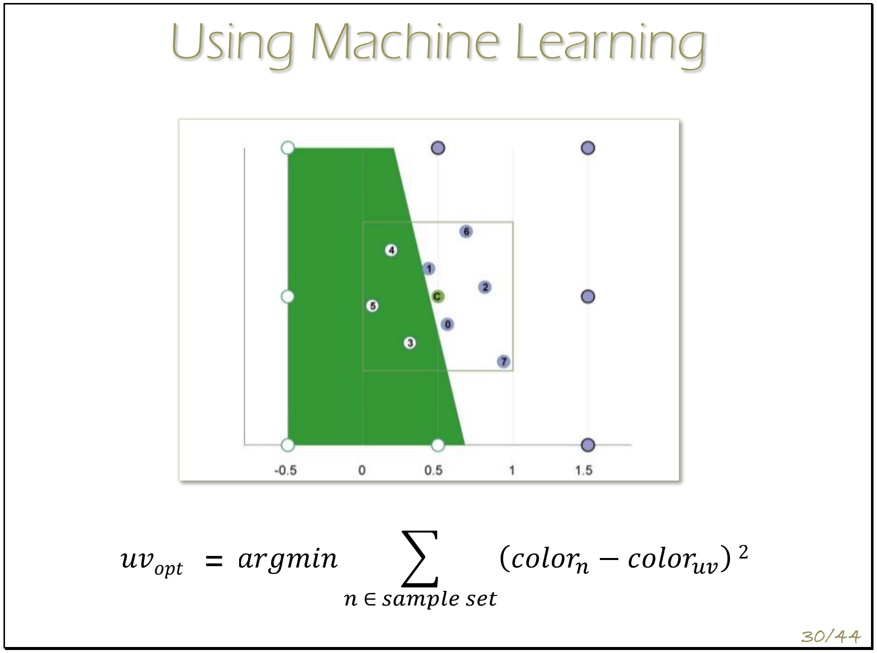
Slide 29



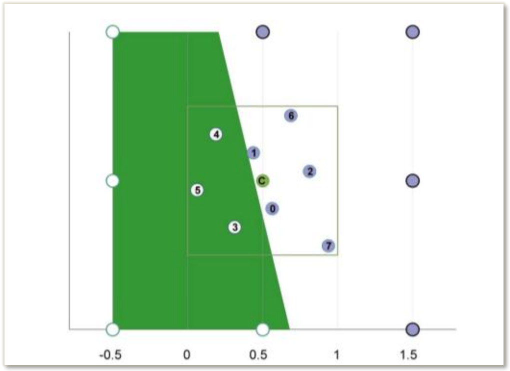
Lookup Table (for filtering offsets)

Not quite like this but close.

Slide 30



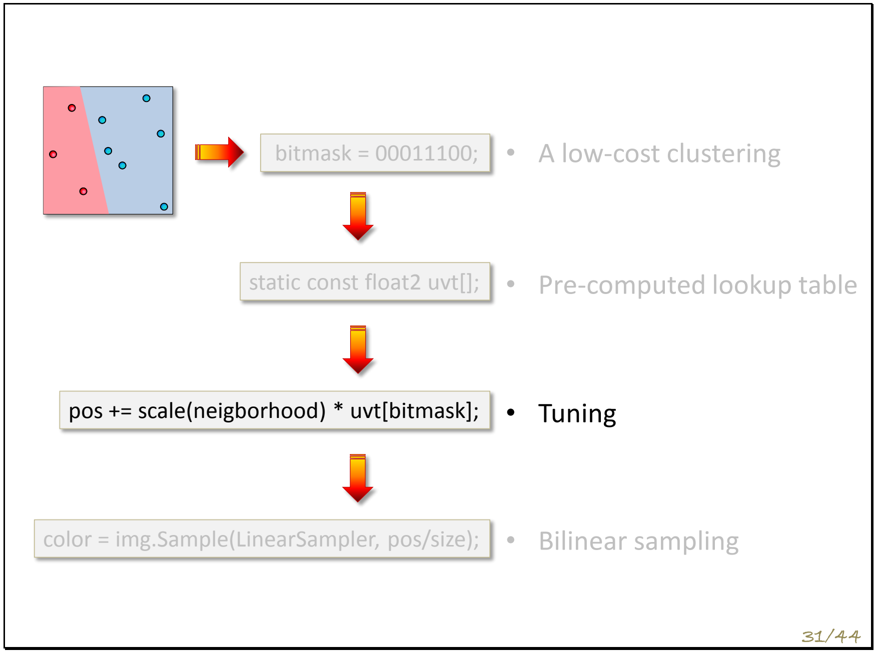
Using Machine Learning

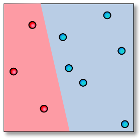


$$uv_{opt} = argmin \sum_{n \in sample\ set} (color_n - color_{uv})^2$$

We use a simple training data set, for which anti-aliasing can be done analytically, and for each bit combination find offset vector which minimizes average error for the training data.

Slide 31





```
bitmask = 00011100;
```

- A low-cost clustering

```
static const float2 uvt[];
```

- Pre-computed lookup table

```
pos += scale(neigborhood) * uvt[bitmask];
```

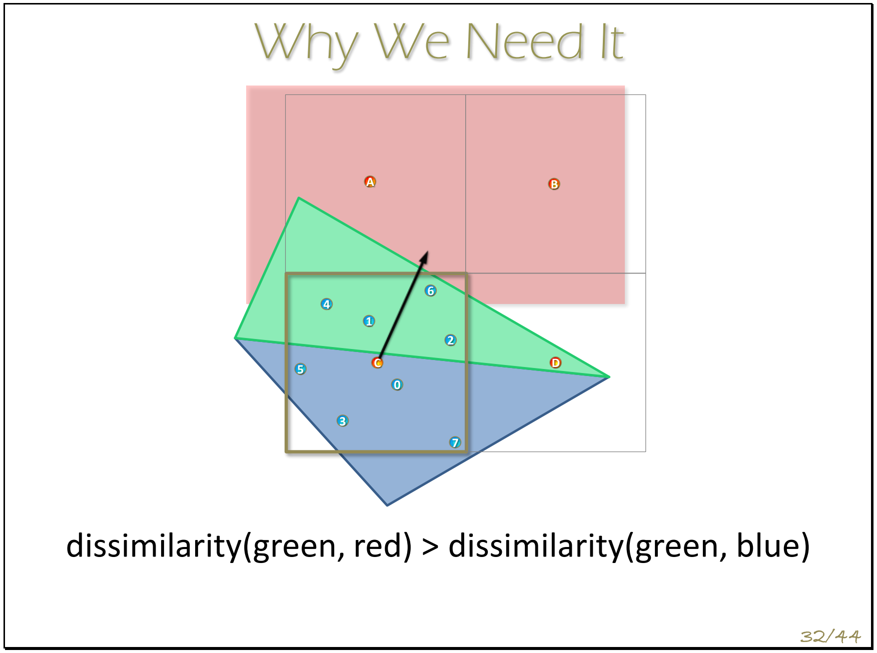
- Tuning

```
color = img.Sample(LinearSampler, pos/size);
```

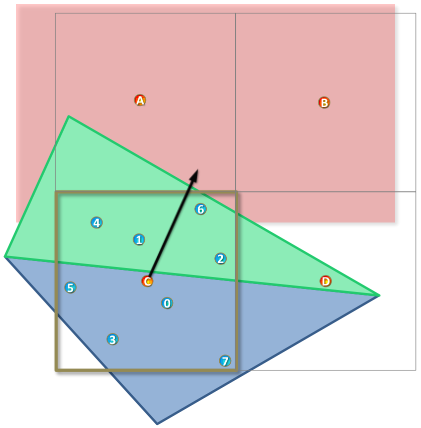
- Bilinear sampling

This will result in a good solution which can be further improved by looking into local neighborhood.

Slide 32



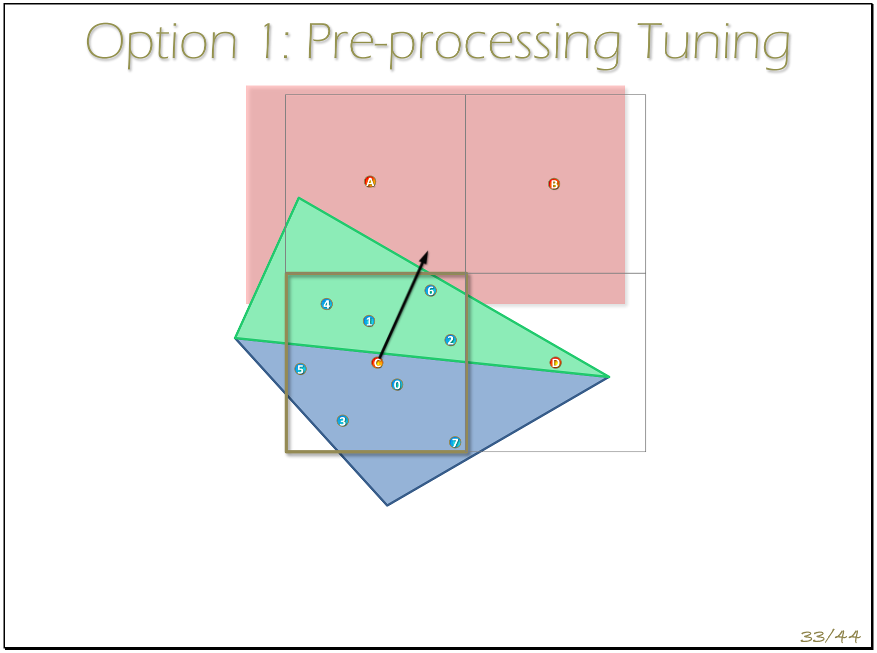
Why We Need It



dissimilarity(green, red) > dissimilarity(green, blue)

Let’s talk why it may be necessary. We have 2 small triangles overlapping a given pixel. By looking on the pixel subsamples, we decided to interpolate the final color at the position defined by the black vector. We were hoping that samples in this direction are representative of green subsamples, which is not true. It will result in blending of blue, red, and a little bit of green, which is not what we need.

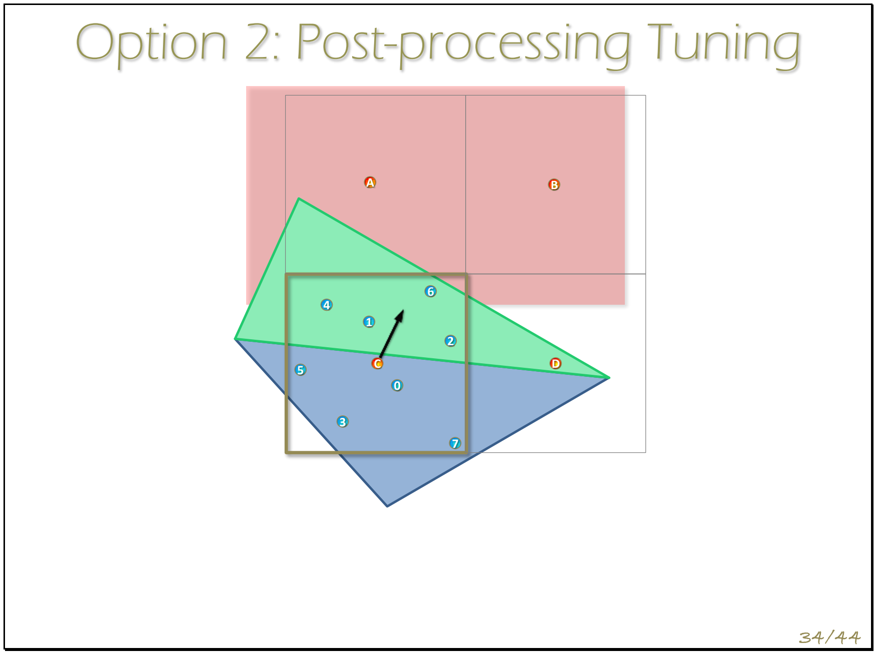
Slide 33



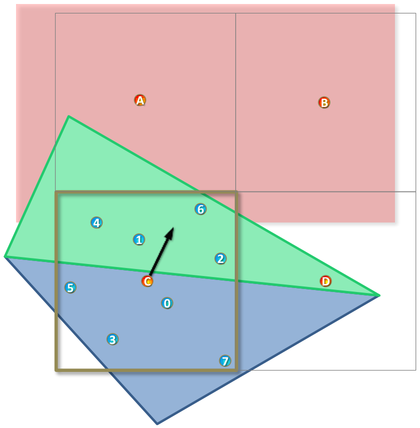
Option 1: Pre-processing Tuning

We can fix this situation by using SADP measure. One way is to compare green subsamples with closest outside samples and keep only ones which have their matches (it will be subsample 2).
This requires extra work, which is unnecessary if pixel is not resampled.

Slide 34

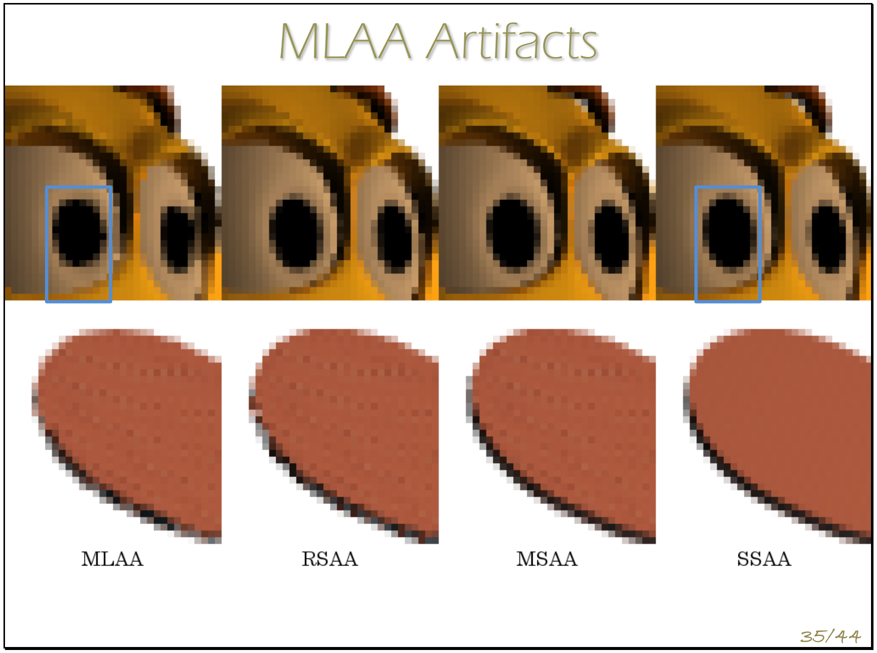


Option 2: Post-processing Tuning

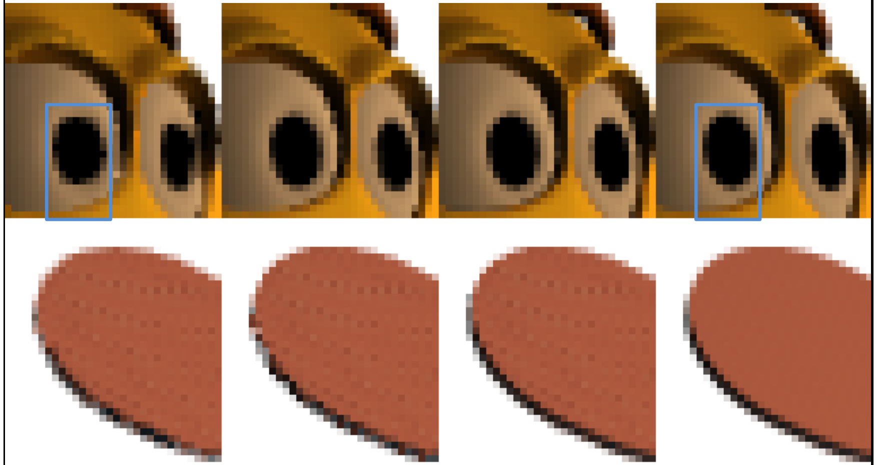


Another approach is to make sure that SADP difference between blue and red samples is greater than the difference between blue and green subsamples and adjust the sampling vector if this is not the case. This can be done once we already decided to resample the pixel, so it is a little bit faster with a tidbit worse quality.
Both tuning methods help improving peak signal to noise ratio by up to 3 dB.

Slide 35

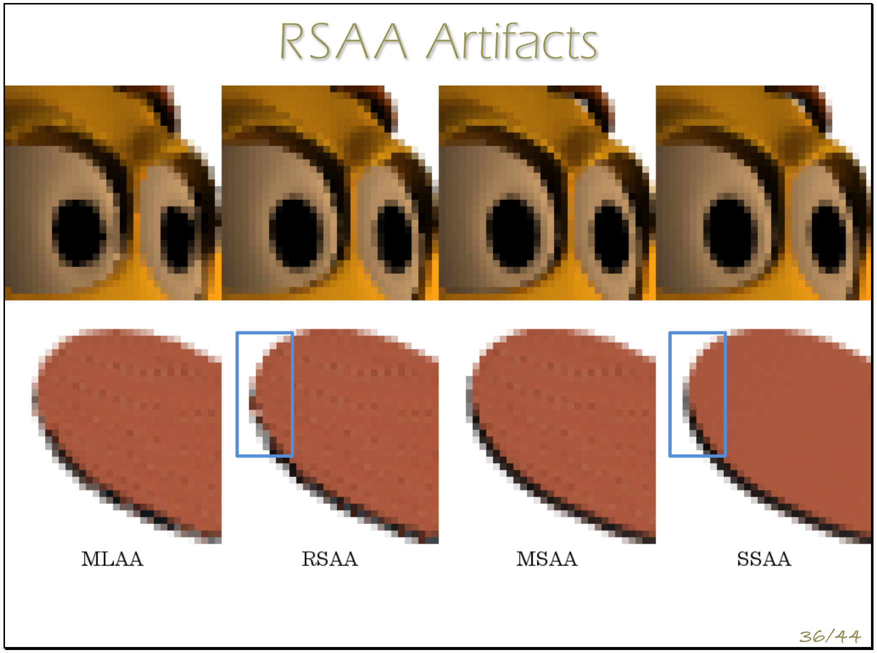


MLAA Artifacts

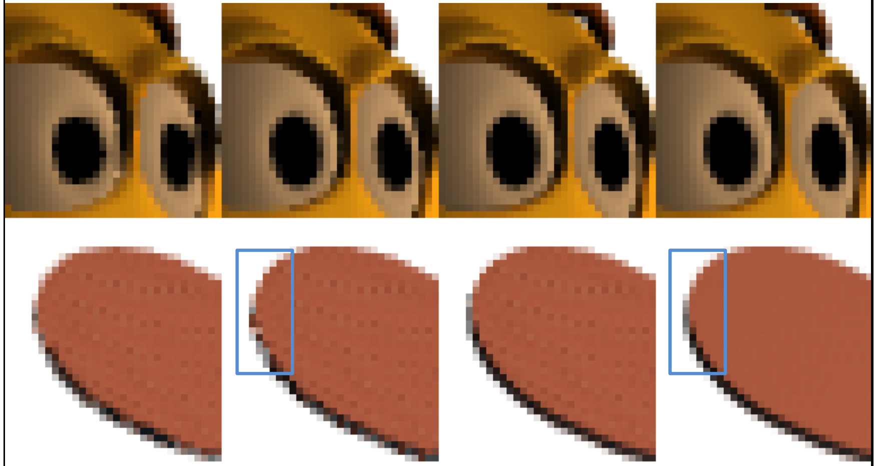


Let’s talk about artifacts. This picture shows 4 antialiasing methods side by side. Let’s first look on MLAA issues. The right pupil looks OK by itself but it is very different from the correct shape. It is because MLAA hallucinates silhouettes due to lack of any more detailed data. It is bad in spatial and especially temporal domain, resulting in flickering.

Slide 36

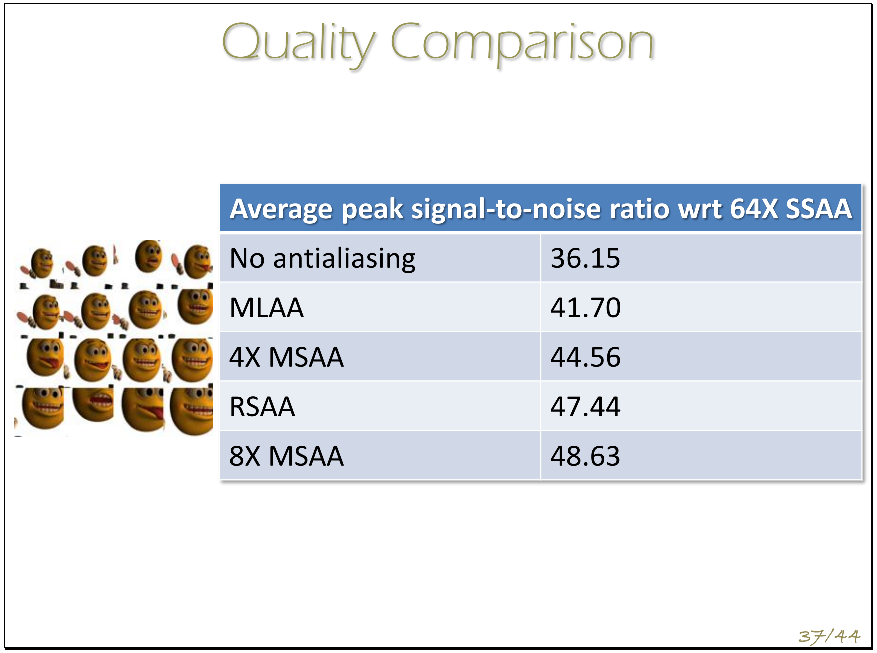


RSAA Artifacts

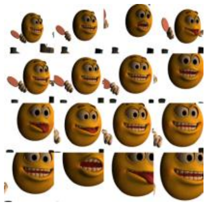


RSAA problems are most noticeable when its assumptions are broken, and there are more than two distinct surfaces overlapping a pixel.

Slide 37



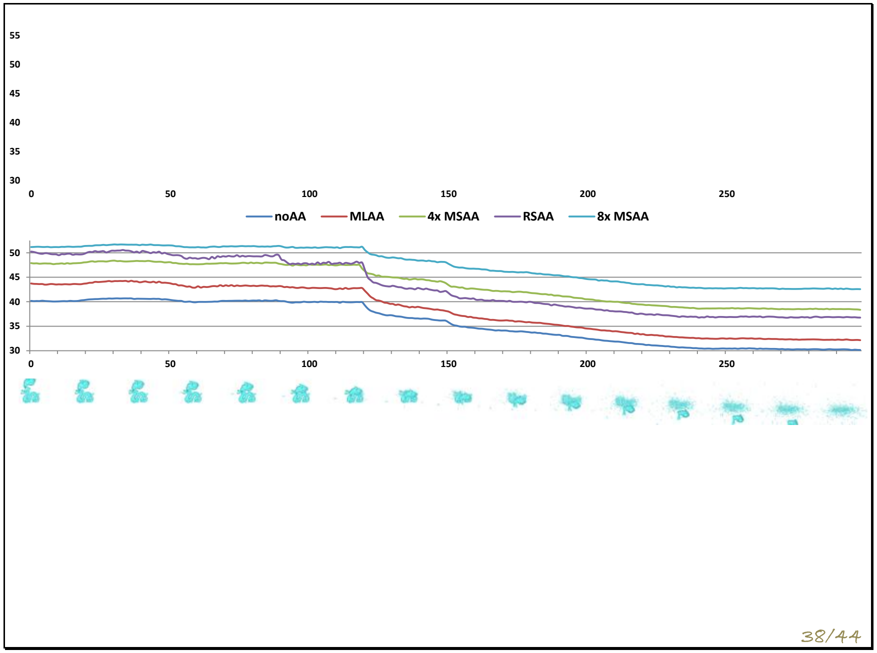
Quality Comparison



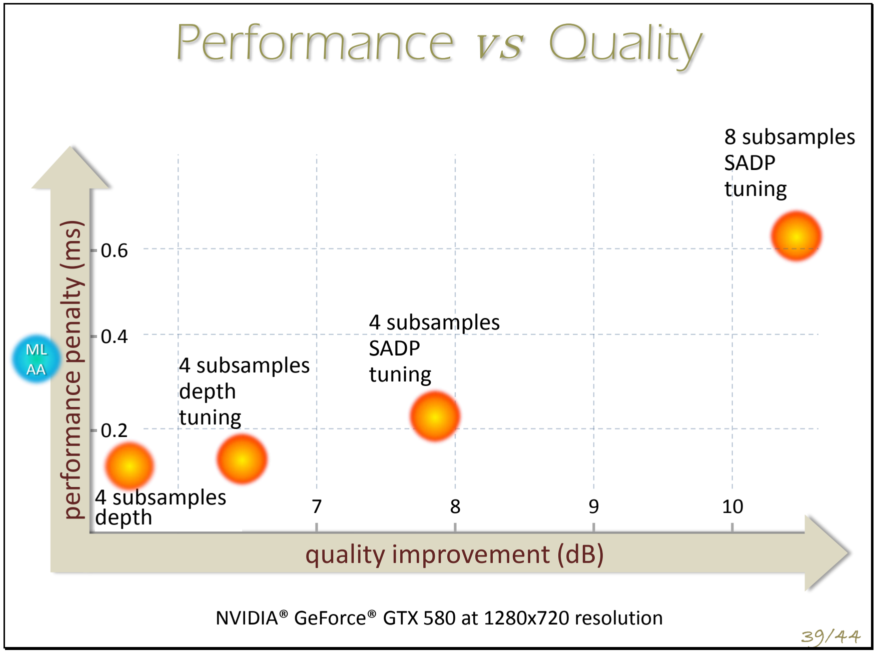
Average peak signal-to-noise ratio wrt 64X SSAA	
No antialiasing	36.15
MLAA	41.70
4X MSAA	44.56
RSAA	47.44
8X MSAA	48.63

Overall, RSAA allows about 10 decibel quality improvement, compared with 5 for MLAA. Generally, RSAA is a little bit worse than MSAA which uses the same number of subsamples.

Slide 38

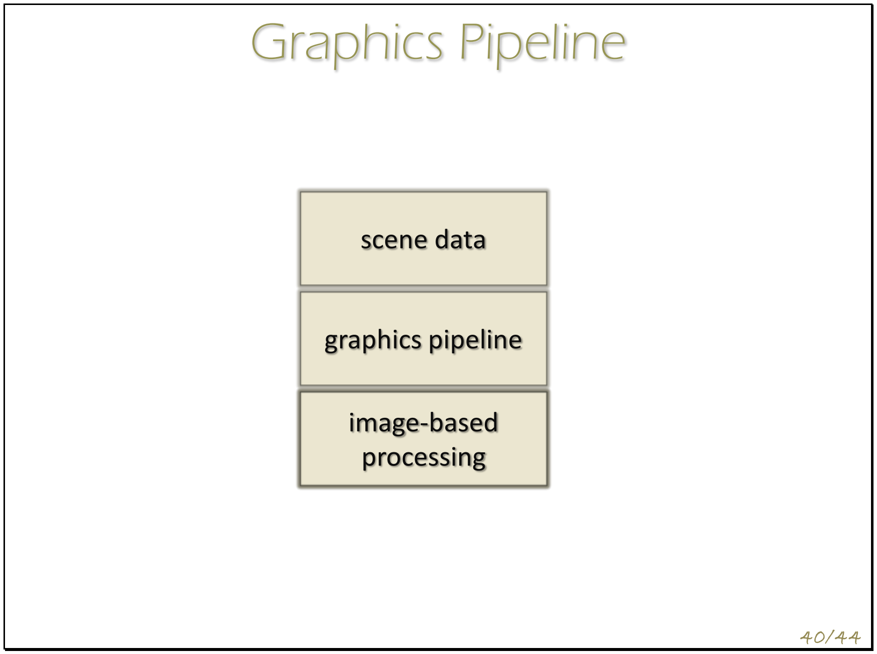


There is one noticeable exception. When there is lot of incoherent subpixel geometry, RSAA quality nosedives, while still exceeding one for MLAA. Just by looking on this chart, it is possible to identify the moment of collision.



At highest settings, RSAA allows about 10 decibel quality improvement, taking 0.6 milliseconds on 580.
If we use only 4 subsamples, quality will go down by 2 decibel, saving about 0.4 milliseconds.
We will not safe much by using depth test instead of SADP, but will loose in quality.
And, it doesn't make much sense avoiding tuning either.
The performance data in this chart excludes the time required for computing and storing per-pixel normals, which we assume could be a part of the deferred shading pipeline anyway. My implementation of this feature takes about 0.25 ms.
And, for comparison, MLAA will be right about here at blue blub (using Jorge Jimenez implementation).

Slide 40

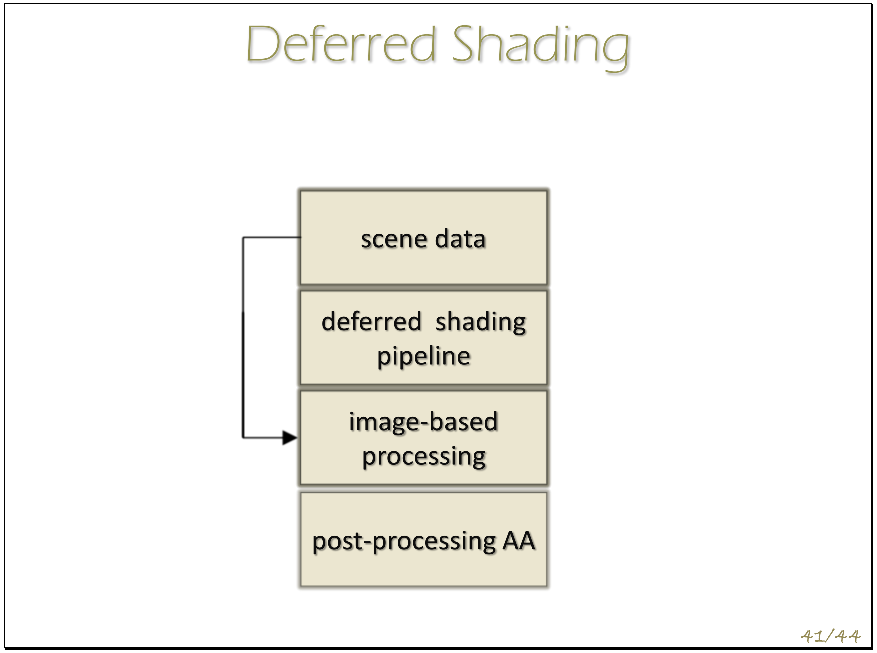


Graphics Pipeline

scene data
graphics pipeline
image-based processing

On the other hand, MLAA is completely orthogonal to graphics pipeline.
During rendering, a 3D scene description is converted to 2D image, as alluded on this chart.

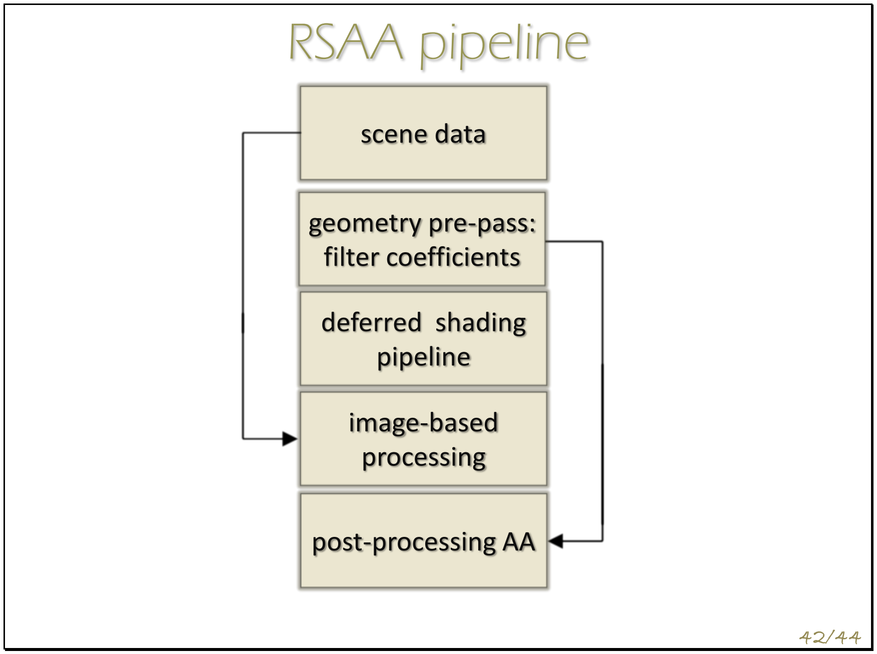
Slide 41



Deferred Shading

In deferred shading, some of the processing, well, is deferred till the last stage, allowing greater artistic freedom is scene composing and lighting.
Post-processing anitaliasing techniques are a natural match for deferred shading, since they are completely orthogonal to the rest of the pipeline.

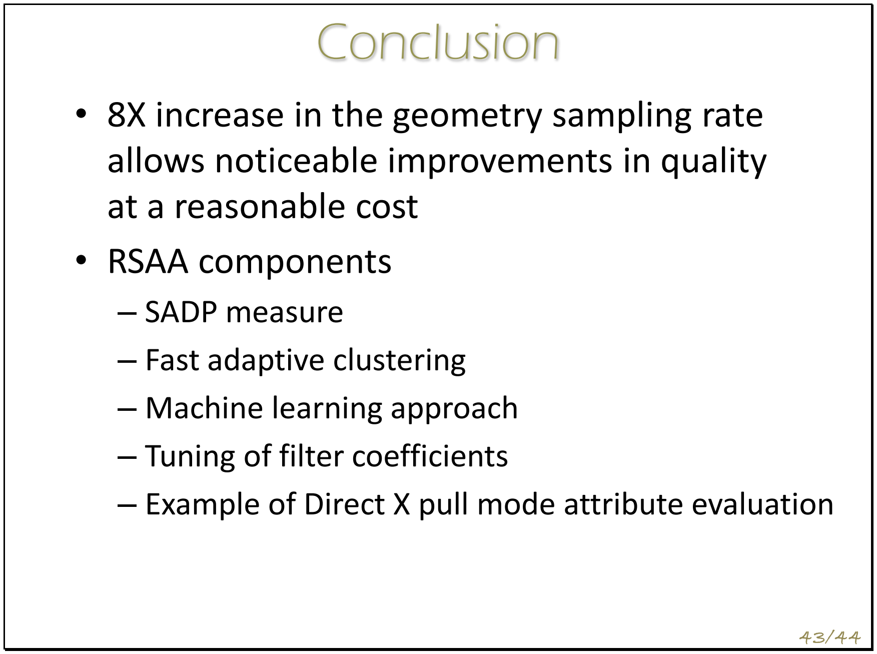
Slide 42



RSAA pipeline

scene data
geometry pre-pass: filter coefficients
deferred shading pipeline
image-based processing
post-processing AA

Technique, which I presented today, introduces an additional pre-processing stage, at which filter coefficients are computed. These coefficients are then used at a post-processing stage to antialias the final image.



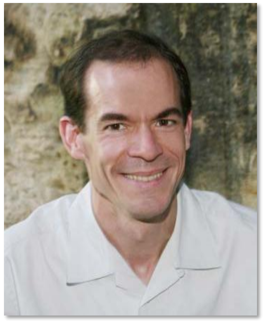
Conclusion

- 8X increase in the geometry sampling rate allows noticeable improvements in quality at a reasonable cost
- RSAA components
  - SADP measure
  - Fast adaptive clustering
  - Machine learning approach
  - Tuning of filter coefficients
  - Example of Direct X pull mode attribute evaluation

It is indeed possible to improve quality by considering additional geometric samples but still shading only once per pixel.
RSAA includes some features which could be used in other computer science disciplines.
I did not discuss the last component here and ordinarily I wouldn't advice relying on my Direct X code, since I am opposite to an expert in this area. However, there are simply no other examples of using the pull mode evaluation on the Web and it is a rather kinky concept.

Slide 44

Acknowledgments



Thank you!