Game Developers Conference

EVOLVE GDC>04

Advanced Visual Effects with Direct3D

Presenters: Mike Burrows, Sim Dietrich, David Gosselin, Kev Gee, Jeff Grills, Shawn Hargreaves, Richard Huddy, Gary McTaggart, Jason Mitchell, Ashutosh Rege and Matthias Wloka

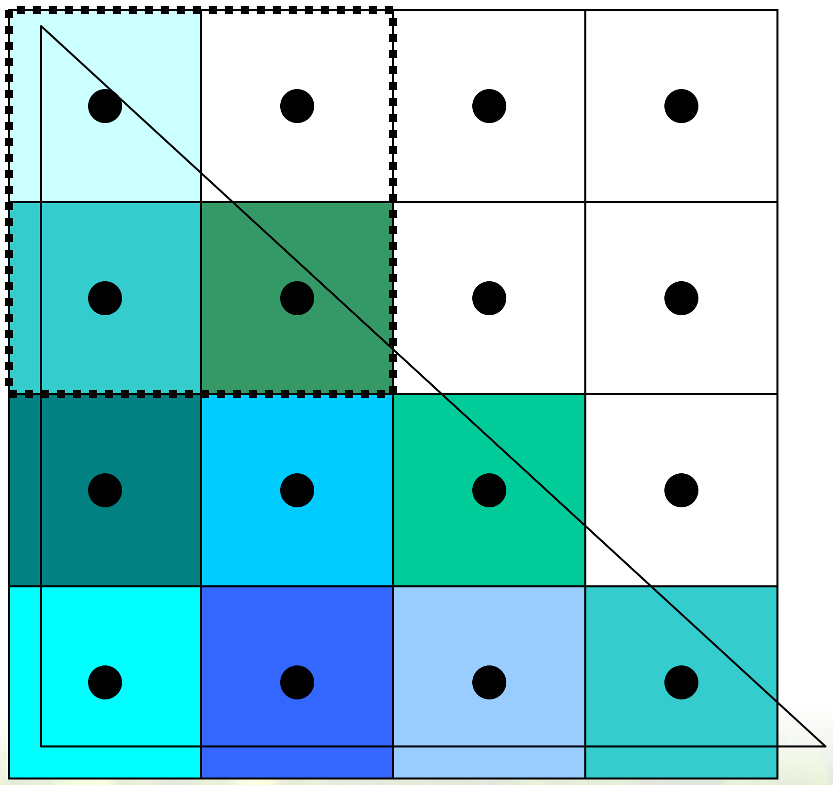
All About Anti-Aliasing

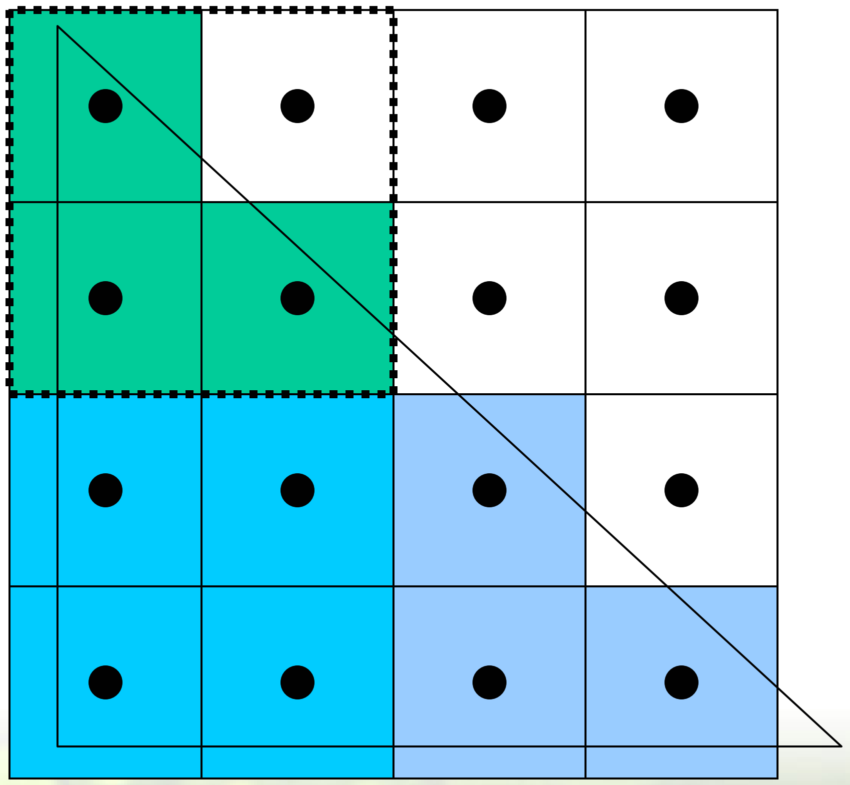
- What is it?
- Explanation of Multi-sampling
- Problems & Solutions

What is Anti-Aliasing?

- On current consumer cards, it’s
  - Super-sampling
    - Just render the scene to a 2x2 larger back & zbuffer & filter down
  - Multi-sampling
    - Like the above, but compute coverage at a higher frequency than shading
  - A Mix of the two, 2x multi and 2x super-sampling simultaneously

4x Super- vs 4x Multi-Sampling

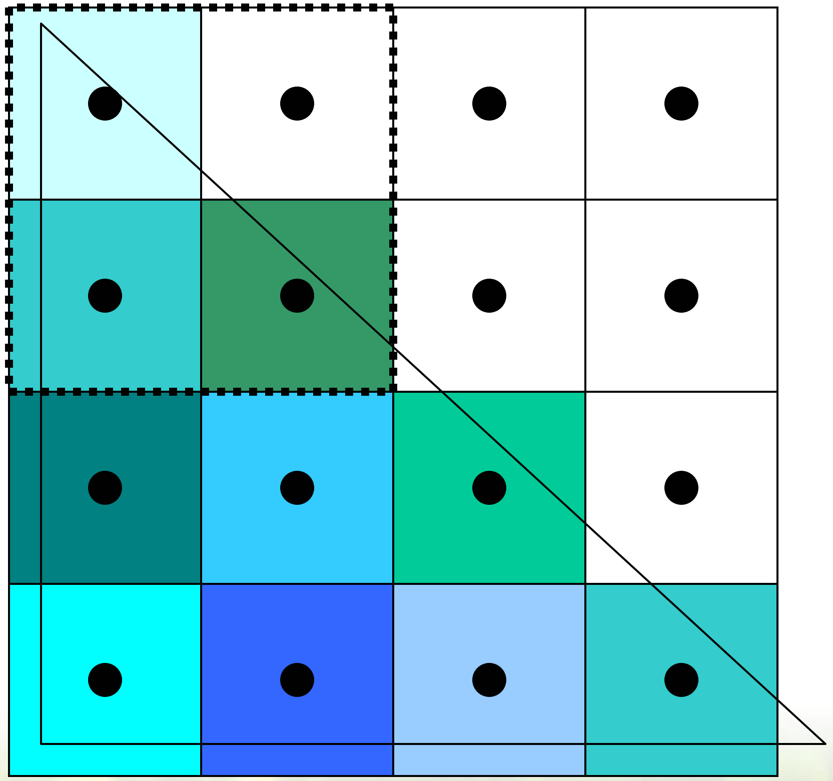


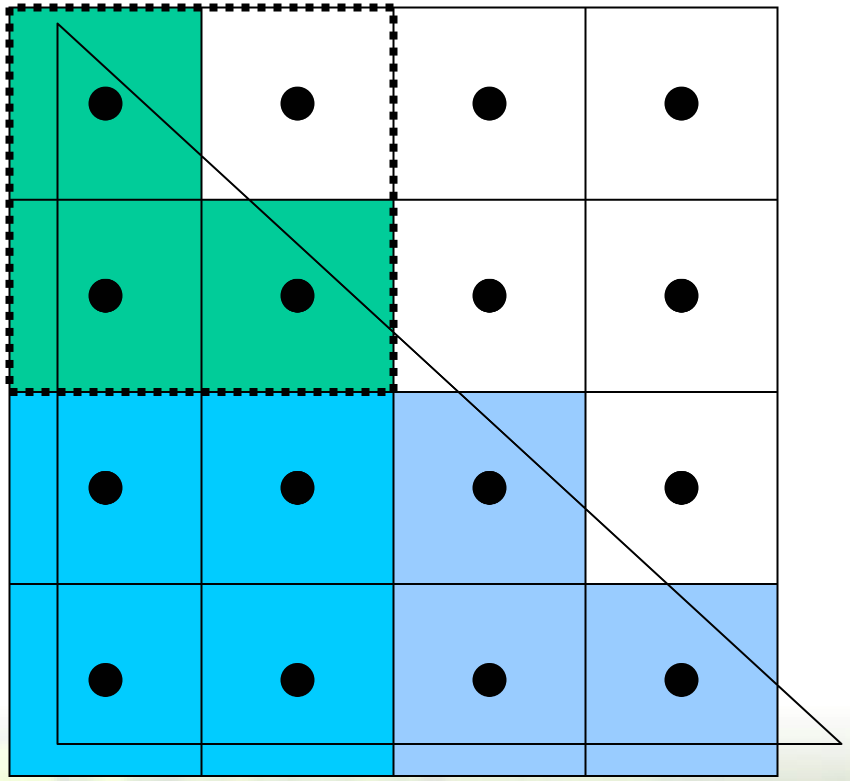


Game Developers Conference

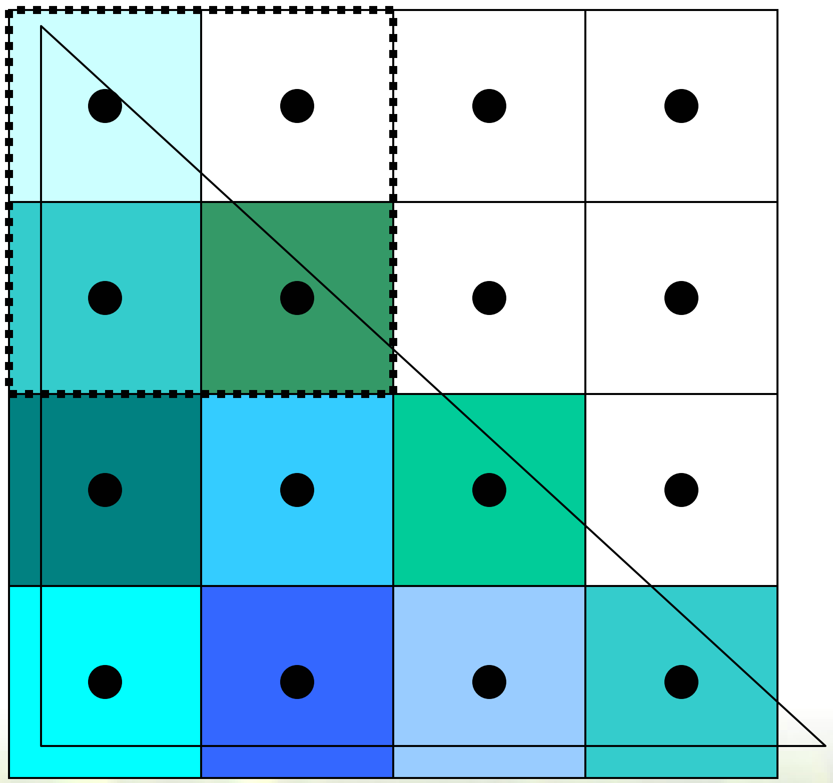
EVOLVE GDC>04

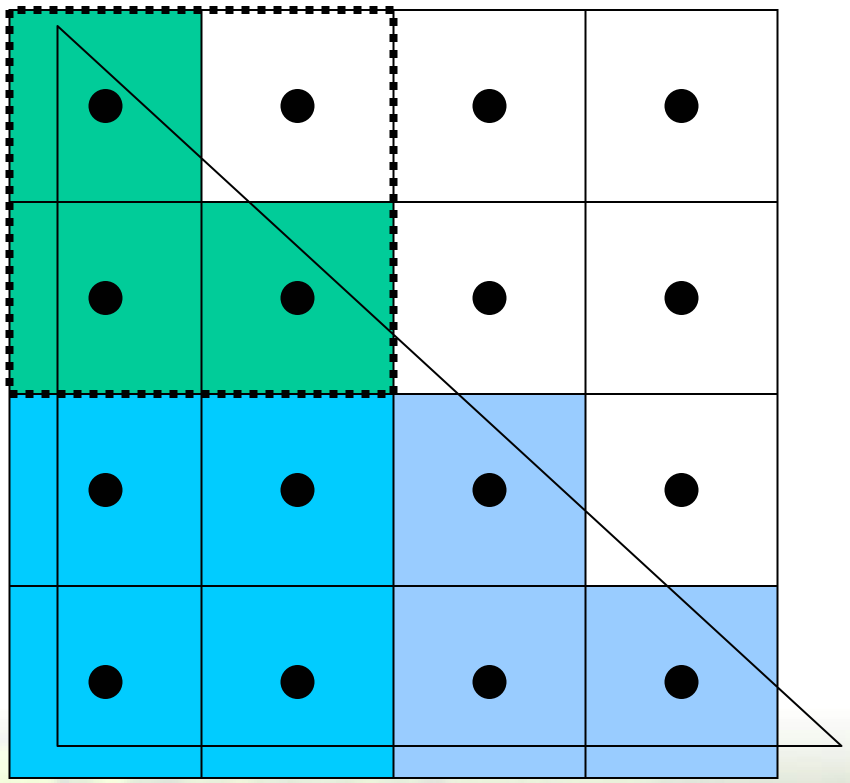
Note how the super-sampled Image has different shading results for each 2x2 area, and the multi-sampled one has only one color per 2x2.


4x Super


4x Multi

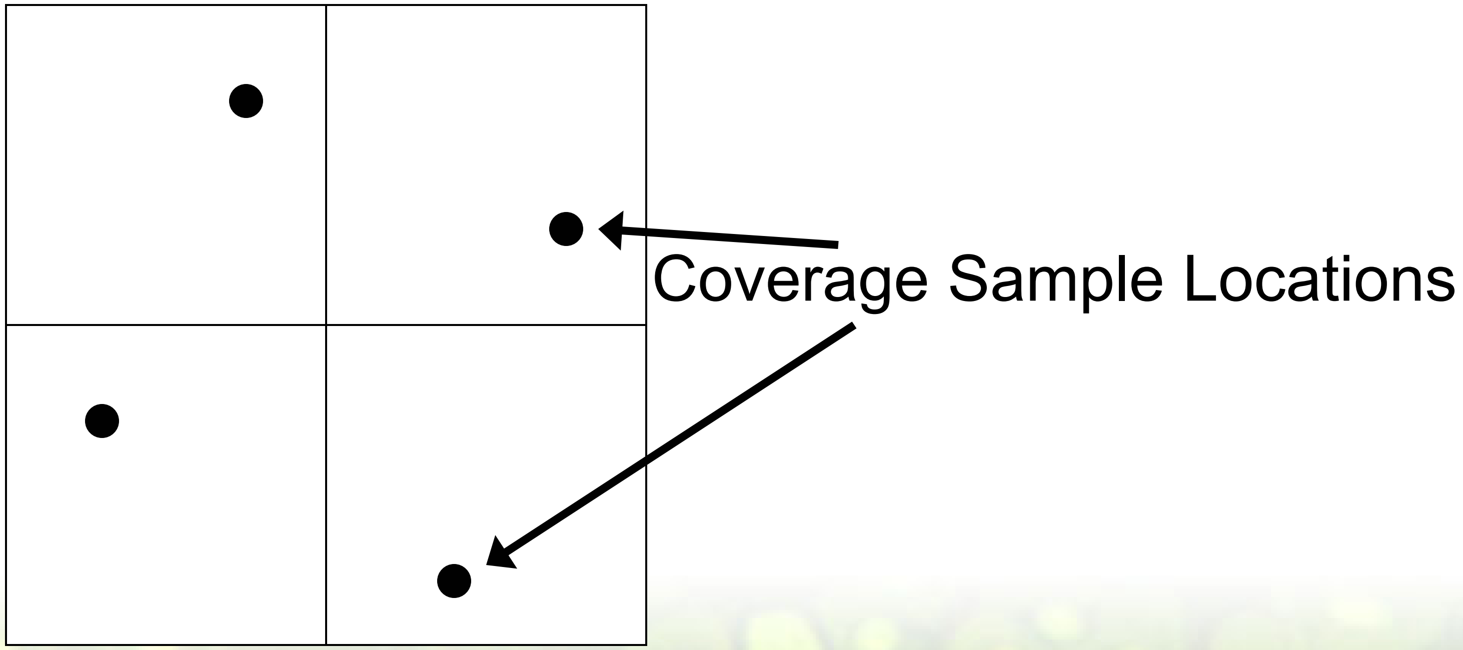
Multi-sampling saves performance by decoupling shading and coverage computation frequency


4x Super

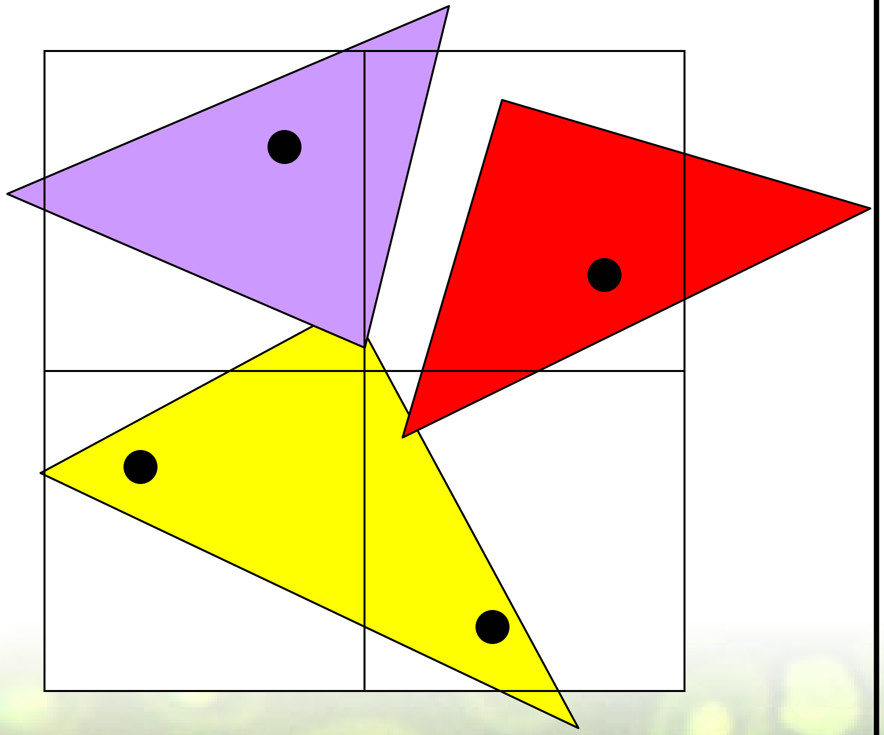

4x Multi

4X Multi-Sampling on a 1x1 Frame Buffer

Step 1 : Render Scene to 2x2 Larger Back Buffer



4X Multi-Sampling on a 1x1 Frame Buffer

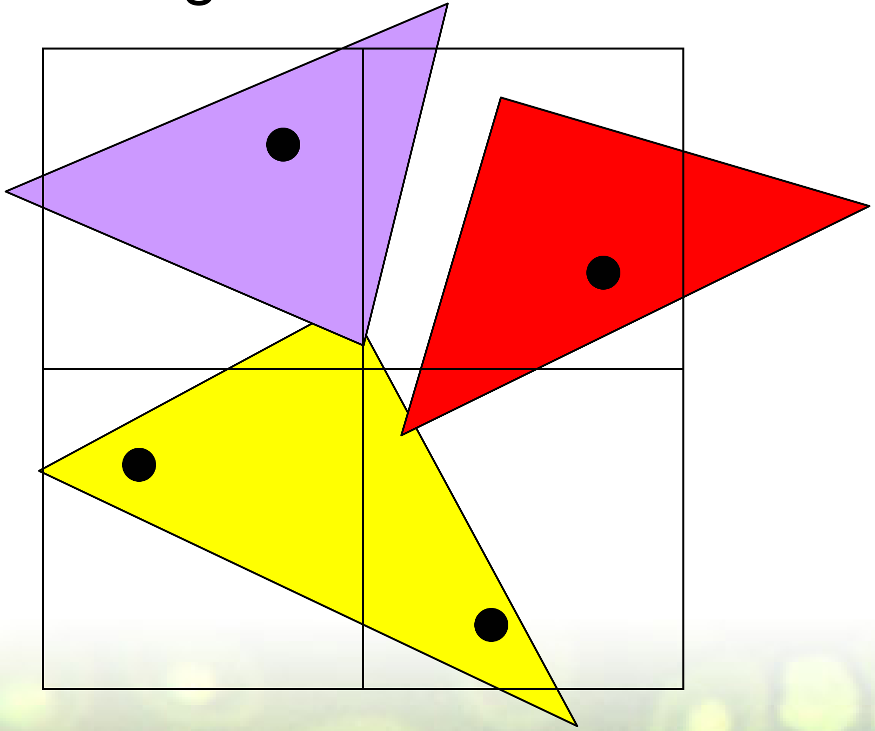


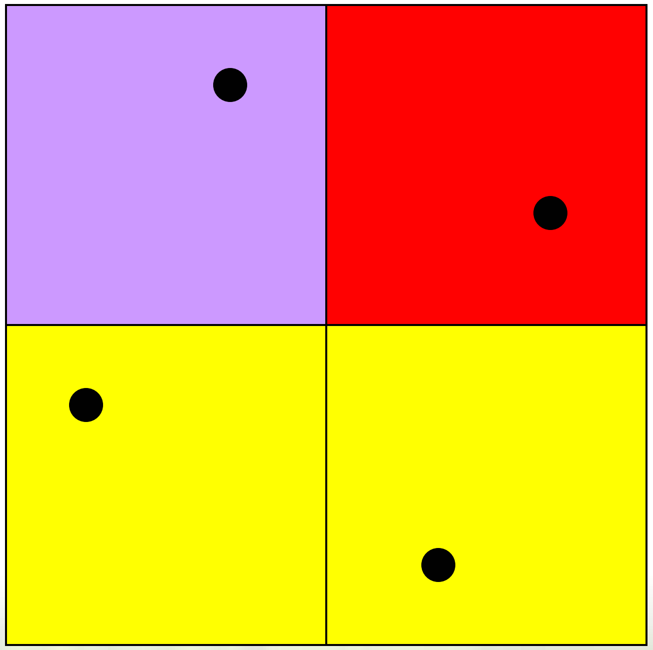
Triangles that cross at least one sample location are rasterized, Z/Stencil tested at **each** covered sample location

The Yellow triangle has 2 Z & Stencil values

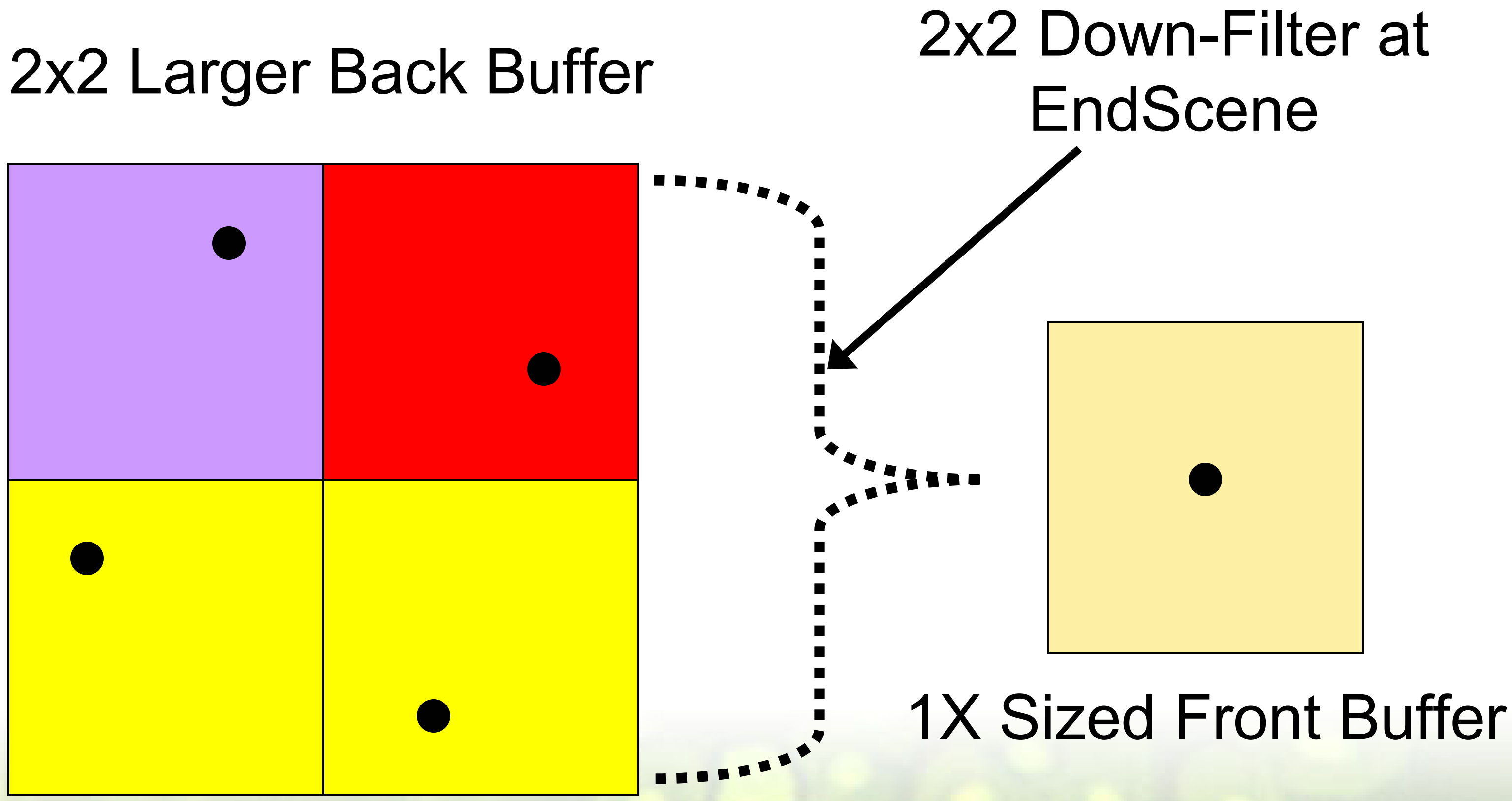
Those triangles that cover > 1 sample point are still shaded only **ONCE**

4X Multi-Sampling on a 1x1 Frame Buffer





4X Multi-Sampling on a 1x1 Frame Buffer



Things that don't get AA'd with Multisampling

- Render to Texture – can't assume it's color
- Clip planes – may be implemented in raster
- Full screen quads
- MRT
- Pixel Shaders
  - Z-replace shaders
  - Texkill
  - Alpha-Test

Things that will Break or Slow Down AA

- Back Buffer Locking
- StretchRect()
  - Can Force a down-sample
- Z Buffer Locking
  - Can Force a ‘down-sample’
- Applying AA Zbuffer to Aliased Texture
  - How is this supposed to work?
  - Just Re-render your z buffer to be sure
- Multiple EndScenes()

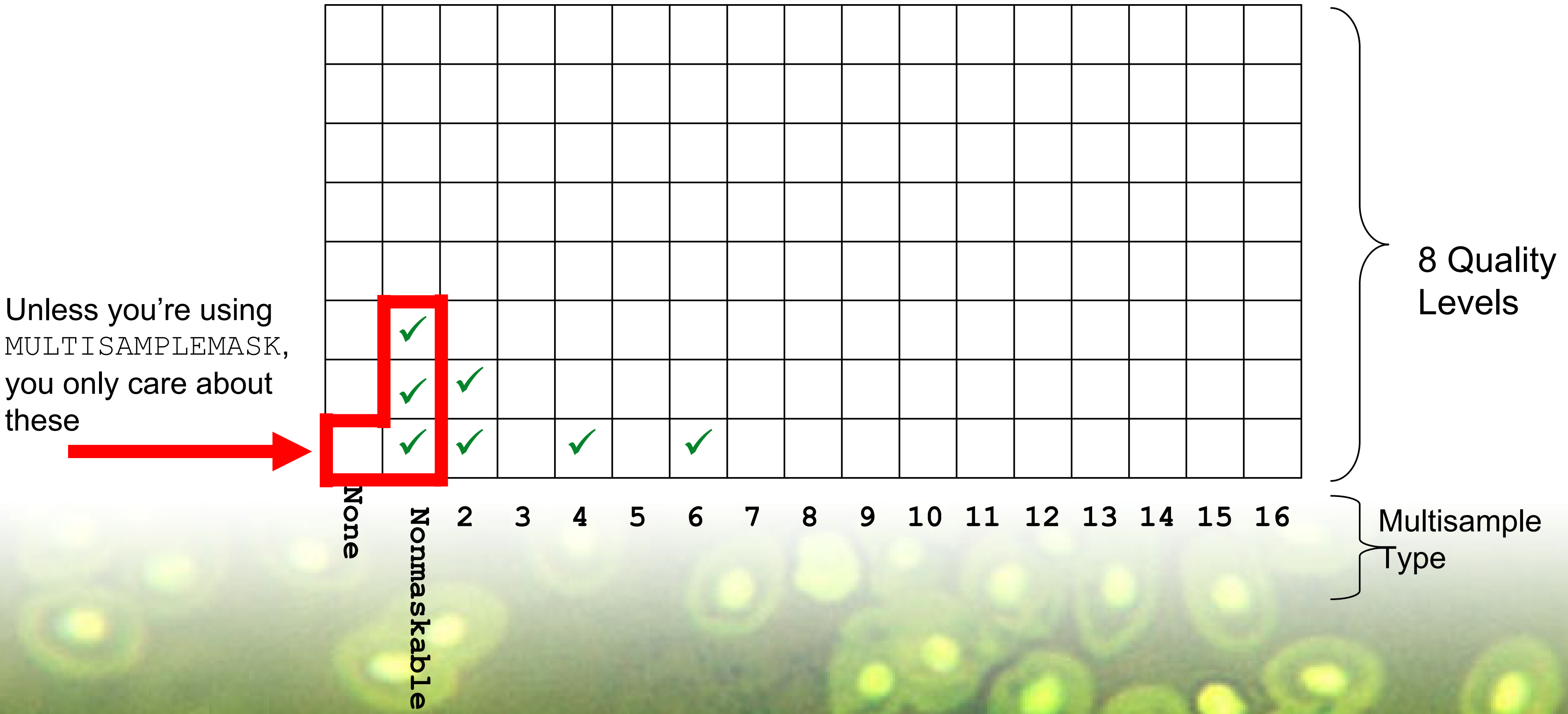
How to Enable Multi-Sampling

- During `CreateDevice()`

```
PresentParameters.MultiSampleType
```

Selecting Multisample Antialiasing

- Control this in your application
  - Use the API!
  - Let your users set the quality



Unless you're using `MULTISAMPLEMASK`, you only care about these
8 Quality Levels
Multisample Type

None	Nonmaskable	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	✓															
	✓	✓														
	✓	✓		✓		✓										

Multi-Sample Mask

- Lets you selectively update each individual multi-sample

- Not supported if

  ```
  d3dcaps9.RasterCaps &
  D3DPRASTERCAPS_STRETCHBLTMULTISAMPLE
  ```

Questions about Multisampling

?

Issues & Solutions

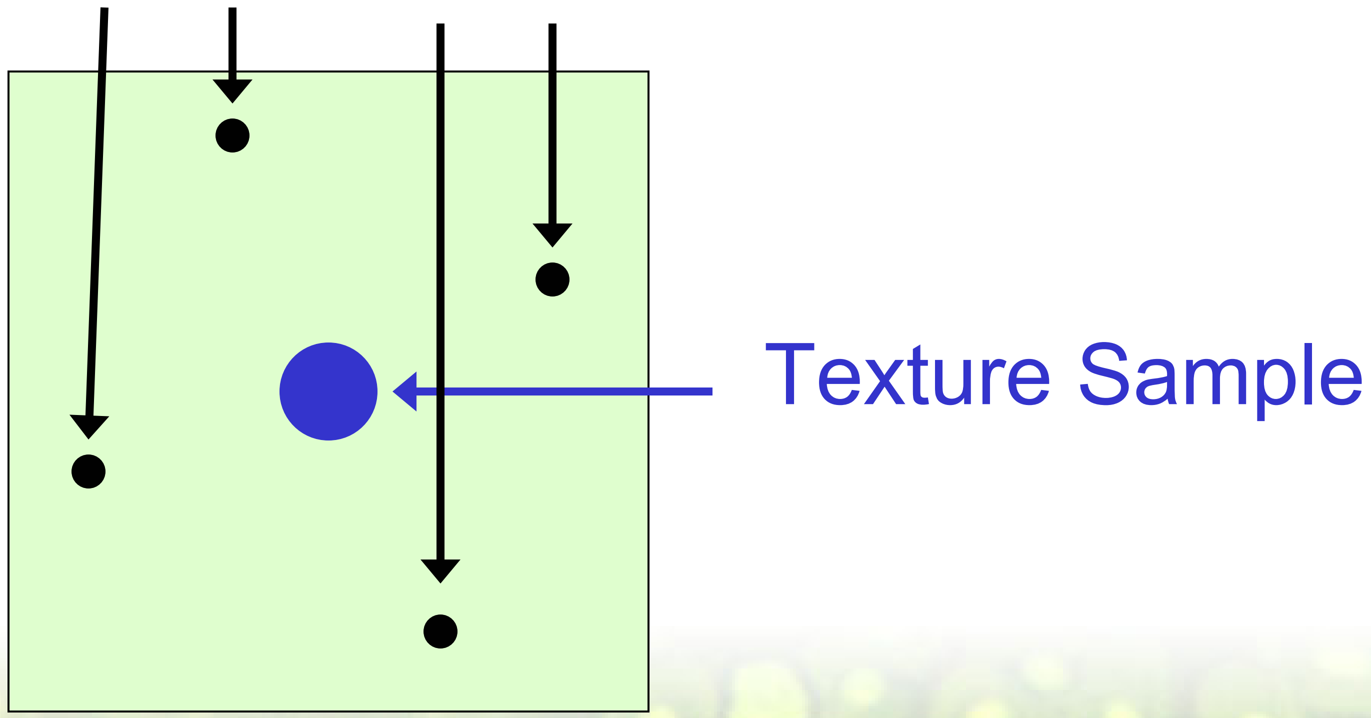
- Texture Atlases
- Video RAM Usage
- Format Incompatibility
- Variable Bandwidth
- MRT Incompatibility

Issue : Texture Atlases

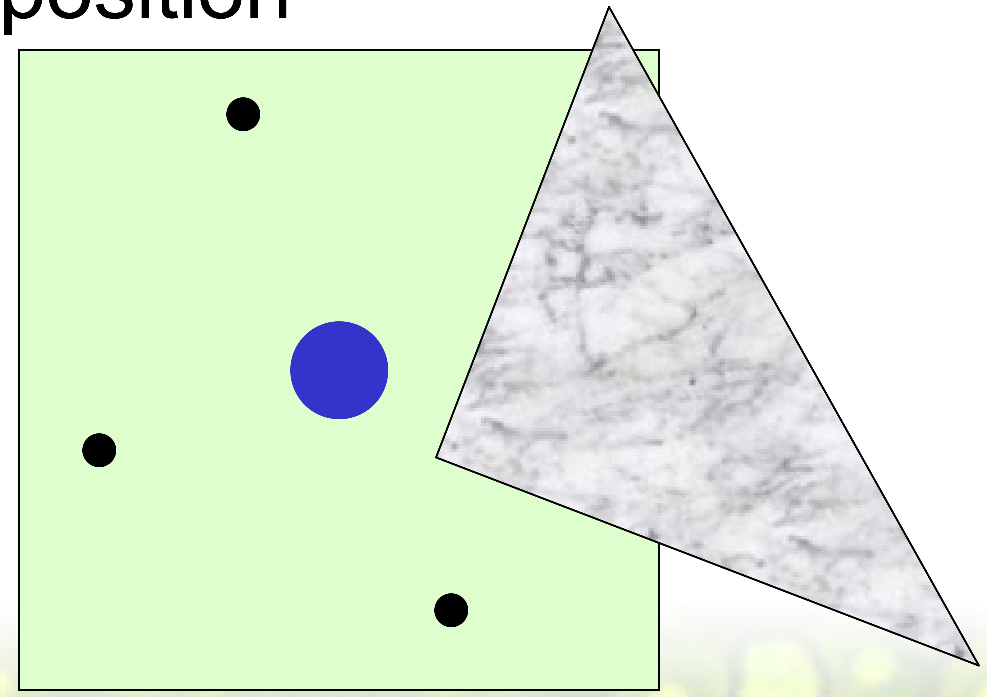
- Games with Texture Atlases can have problems with flashing texels at certain angles
  - Like packed lightmaps
  - Multiple Character skins per texture page

4X Multi-Sampling & Texture Atlases

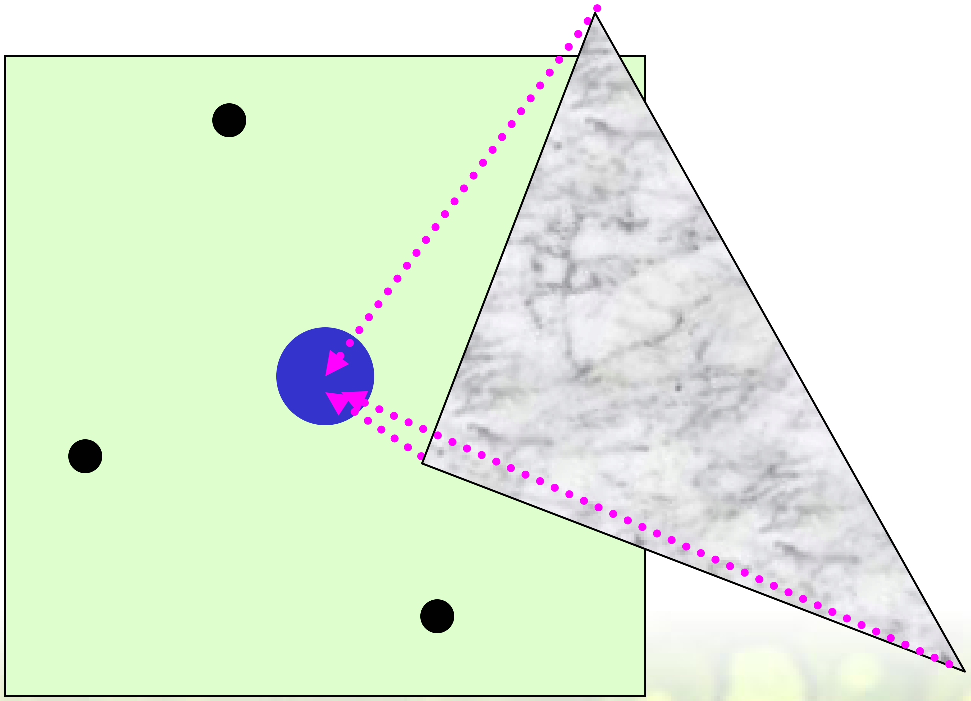
- Coverage Samples



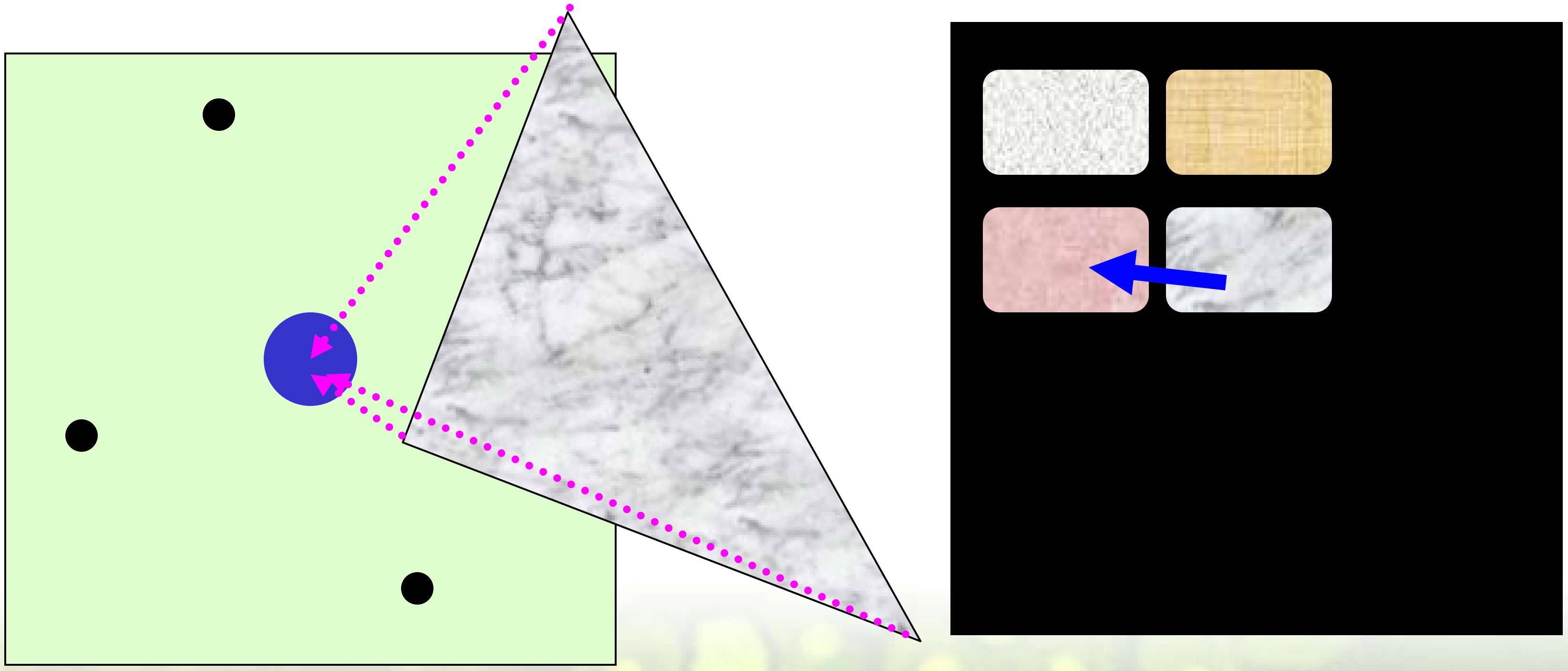
- Although this triangle does cross a sample location - this triangle *fails* to cross the pixel’s texture sampling position



- …so, the uv coordinate is extrapolated *outside* the triangle‘s uv gamut



- If using texture atlases, this can cause an incorrect texel to be selected from a different chart



Texture Atlas

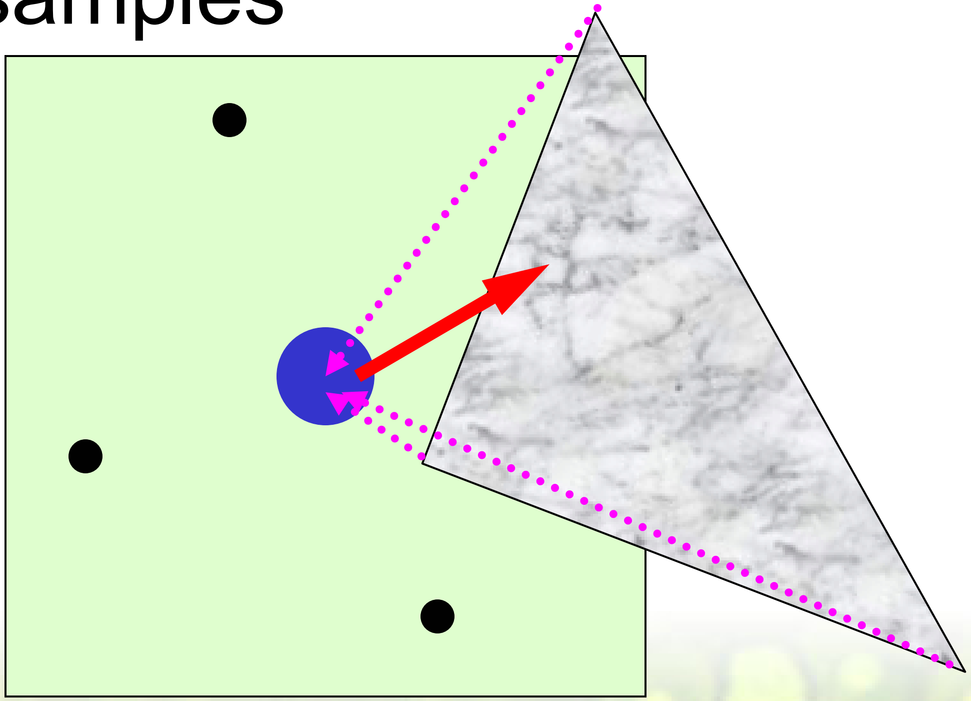
How Bad Is It?

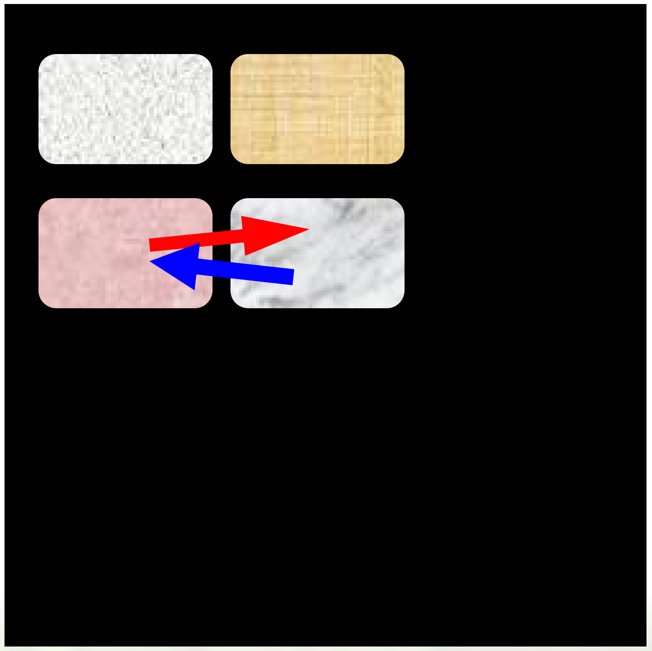
- Well, all games ever shipped have had this problem with multi-sampling
- So it's not fatal
- But artifacts can be seen on triangles edge-on to the view
- Gets worse if atlas contains many different colors

Enter the Centroid

- DirectX9 introduced Centroid Sampling to address this issue
  - Basically, if a texel sample falls outside of the triangle’s valid UV gamut, it’s snapped to be inside the UV gamut

- Centroid sampling forces the interpolated parameter to stay in the triangle's valid uv gamut – at the centroid of the covered samples





Texture Atlas

Other Solutions

- Centroid is available on some pixel shader 2.0+ hw
- Other options include
  - Clamping texture coordinates in the pixel shader to chart's uv rect
  - Using a separate clamped mask texture corresponding to chart
  - Live with it, but store similarly colored lightmaps together
  - Add a border to each chart via dilation filter or calculation outside of chart gamut

Issue : Greater Video Ram

- Multi-sample AA requires more video ram than aliased rendering
- Simple formula for 4X AA often wrong :

```
front_buffer_size +
4 * front_buffer_size +
 4 * z_buffer_size
```

- Exactly how much is not obvious, and can depend on IHV, GPU and driver

More Memory Than Anticipated

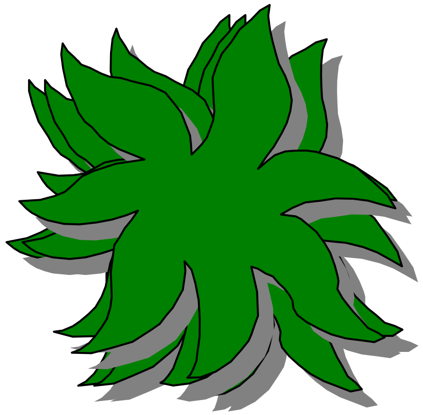
- There may be 2 large back buffers
  - Some HW scans out of super-buffer using DAC
- There may be > 1 front-buffer-sized back buffer
  - To hold filtered, but not yet displayed buffers
- Best bet is to query the AvailableVidMem() after device creation in case AA is forced on

Issue : Sparkly Alpha Test

- Using alpha test without alpha blended edges looks noisy
- Especially apparent with trees
- Multi-sampling only samples once per final pixel, not per-sample, so alpha test is binary

Solution : Custom Super-Sampling

- Use multi-sample masking to render the leaves of an alpha tested tree several times



- Each render is offset a half pixel or so
- Not Z correct, but for leaves, ok

Solution : Custom Super-Sampling

- The blending between the 4 versions of the leaves happens at the normal downfilter time :
  - Either Present()
  - Or StretchRect()

Issue: fp Render Target Incompatibility

- Multisampling doesn't work with fp16 or fp32 render targets
  - It could, just a limitation of current HW

- If you want higher quality, you can do your own super-sampling

Solution : Custom AA

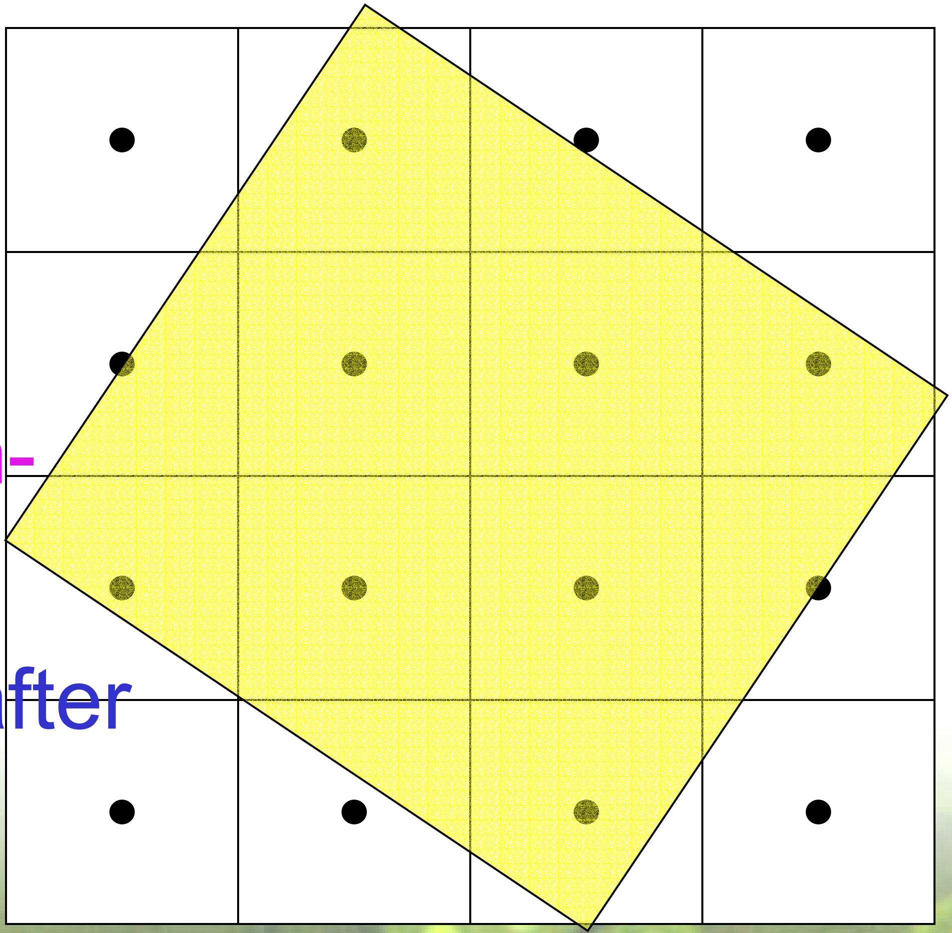
- You can perform your own edge anti-aliasing one of several ways
  - Render your scene to a 2x2 larger texture, with 2x2 larger z/stencil buffer then bilinear filter it down to the back buffer
    - Ordered Grid Sampling – Not Ideal
    - Performs Shader AA also
    - More Expensive than HW Multisampling
    - Needs no extra render passes of scene geometry

Custom AA

Use a rotated back buffer for Rotated Grid AA Sampling

Unrotate during Down-Filtering

Draw HUD after downfilter



Custom AA

- Render your scene multiple times into different buffers, then average together at EndScene() via pixel shader
  - – ala 3dfx T-Buffer
  - – Requires multiple scene passes
  - – Needs more than 1 Z buffer

Issue : Variable BW Costs

- During low-action scenes, it would be nice to have very high AA levels
- During fast-action scenes, especially w/ alpha particles & sounds, frame rate is more important than image quality
- How to balance these conflicting desires?

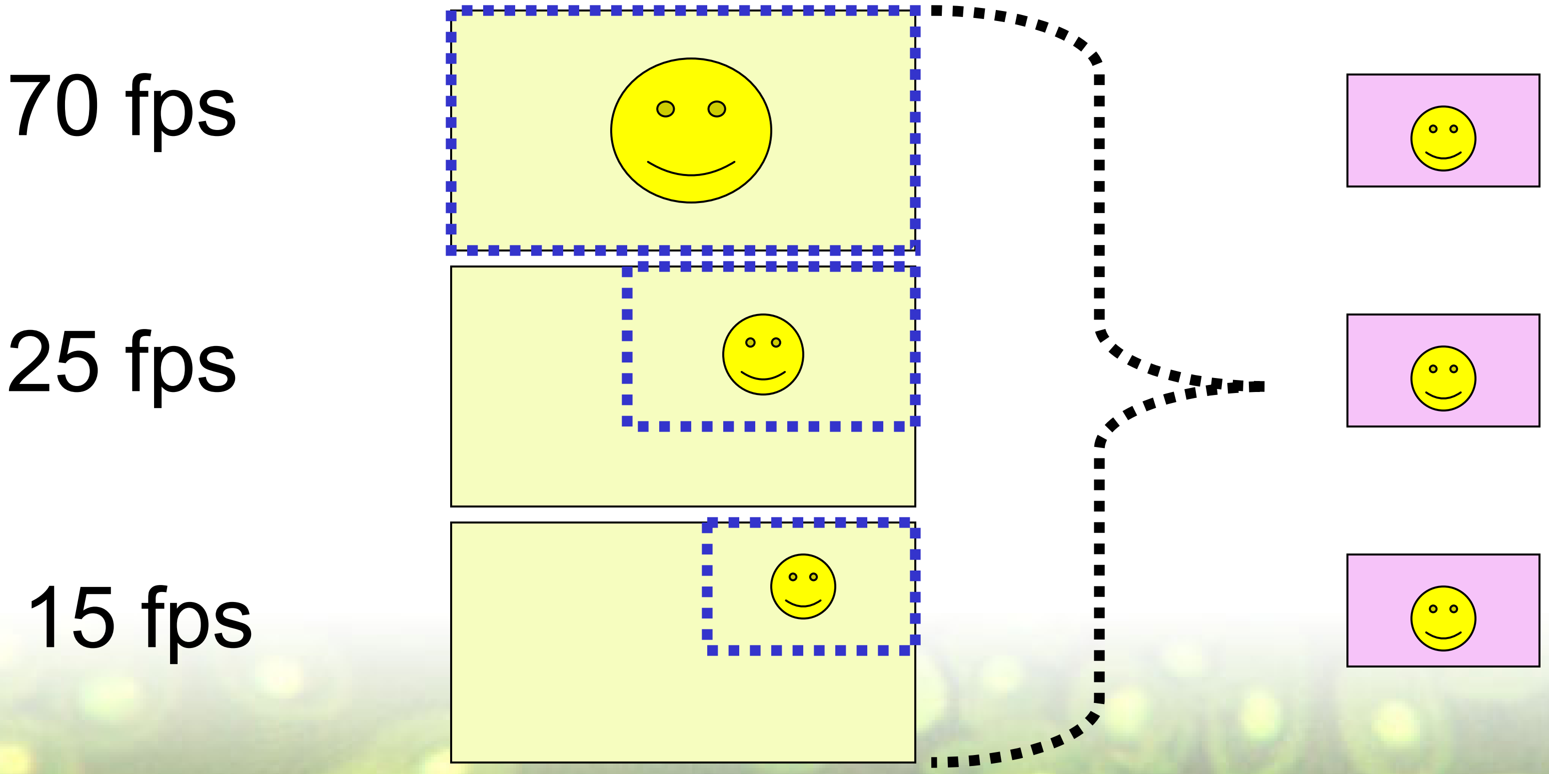
Solution : Dynamic AA

- Variation on Custom AA
- Allocate 2x2 larger back buffer for AA
- In high frame rate scenes, just perform normal 4X multi-sampling, but perform your own downfilter using StrechRect()

Dynamic AA

- During low-frame rate periods, reduce your viewport size on the 2x2 larger back buffer
- Still StretchRect() to same sized buffer
- Render HUD Afterwards
- Keeps framerate more even

- Back Buffer Shrinks w/ FPS
- Down-Filter to Constant Front Buffer Size



Test Results

- Looked good
  - Except for text, which crawled
  - Just render HUD after the StretchRect()
- Variable framerate smoothed out
  - Non-integer AA samples don’t quite look as good (a bit blurry)
    - But restricting the technique to only choose 2x1, 2x2, 3x2, etc. doesn’t give enough options
  - Only helps if b/w or shader bound

Issue : Post-Processing w/ AA

- Can't get have a Multi-Sampled Render Target Texture
- Can't blt from Multi-Sample Back Buffer to texture in DirectX 9.0a

Solution : DirectX 9.0b & StretchRect

- The DirectX 9.0b+ runtimes introduced the ability to StretchRect() from a multisampled back buffer to an offscreen texture
- This can then be manipulated w/ glows, filters, HDR, etc.

Issue : Deferred Lighting

- One of the main ideas about deferred lighting is to render the light bounds as geometry during lighting passes
- This is instead of rendering the scene objects again, saving vertex & CPU
- You can’t have a multi-sampled MRT on current HW

Deferred Lighting w/o MRT

- So, if we want AA, we either have to perform our own custom AA
- Or, we can try to mix Deferred Lighting and Multi-Sampling
  - Allocate a 4x Multi-Sample Back Buffer
  - Create offscreen surfaces for normal, depth, etc.
    - What size? 1X, or 4X?

Super-Sampled Lighting?

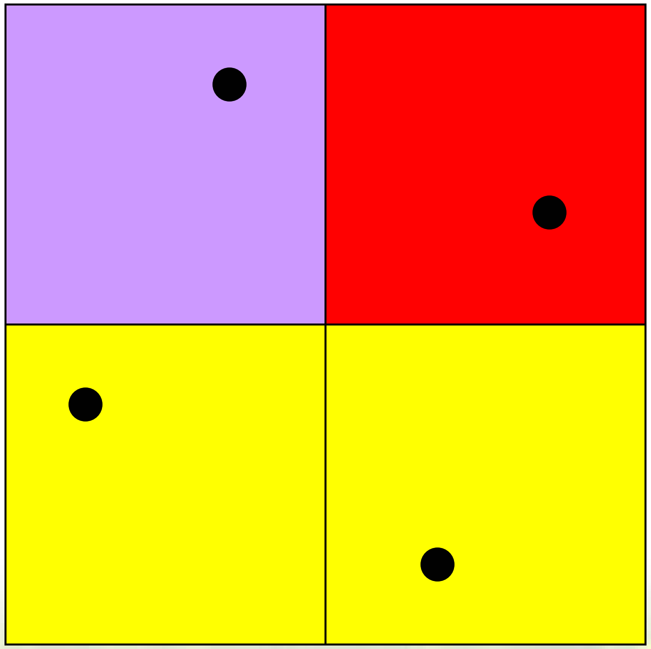
- Ideally, we would want to treat the multi-sample back buffer as super-sampled
- That way you could 2x2 over-sample the lighting
- But, you can't get at the multi-sample buffer this way
- And there's no guarantee the HW stores it as a contiguous buffer
- Also exact multi-sample locations are unknown

Back To StretchRect()?

- So, we’re forced to Down-Filter to a 1X buffer for each term
  - Diffuse & Specular
  - Normal – Must Renormalize
  - Depth?
  - Triangle Edges aren’t really correct

Broken Edges

- Multi-sampling effectively performs super-sampling when the primitive covers only some sample locations.
Filtering these 4 values before lighting is just wrong.



Broken Edges

- The only way to selectively update the right sub-pixel positions is to re-render the scene geometry!
- Thus defeating one of the main points of Deferred Shading
- Rendering the Light geometry on top of the down-filtered normals, depths, etc is wrong.

So, MultiSampling & Deferred Shading Don’t Get Along

- You really need to re-render your scene geometry every time you want to light it
- Or face, color, depth and normal discontinuities

Questions?

sdietrich@nvidia.com