Visual Appearance: Antialiasing and Transparency

CS425: Computer Graphics I

Fabio Miranda

https://fmiranda.me

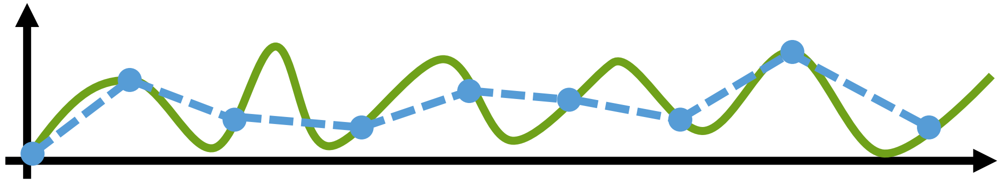
COMPUTER SCIENCE

Overview

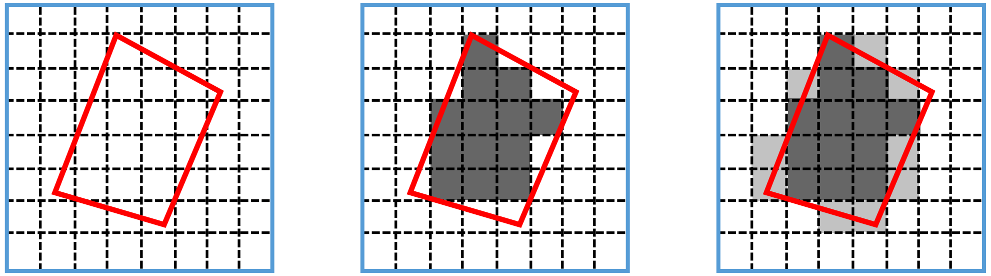
- Aliasing
- Sampling and filtering
- Screen-based antialiasing
- Transparency

Aliasing

- Under sampling a high-frequency signal can result in aliasing.



- Rendering images is inherently a sampling task: generation of an image is the process of sampling a 3D scene in order to obtain color values for each pixel.

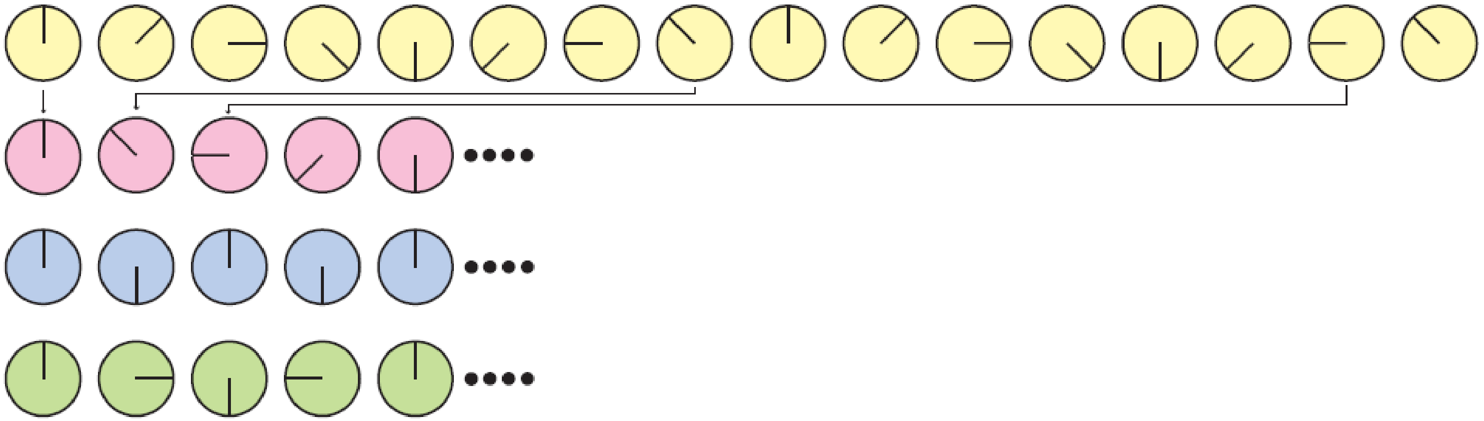


Aliasing

- Whenever sampling is done, aliasing can occur.
- Examples of aliasing in computer graphics:
  - “Jaggies” of a rasterized line or polygon edge.
  - Flickering highlights.
  - Texture with a checker pattern is minified.

Temporal aliasing

- Classic example: spinning wheel filmed by a movie camera.
- Wheel spins faster than the camera records images. Result: wheel may appear to be spinning slowly.



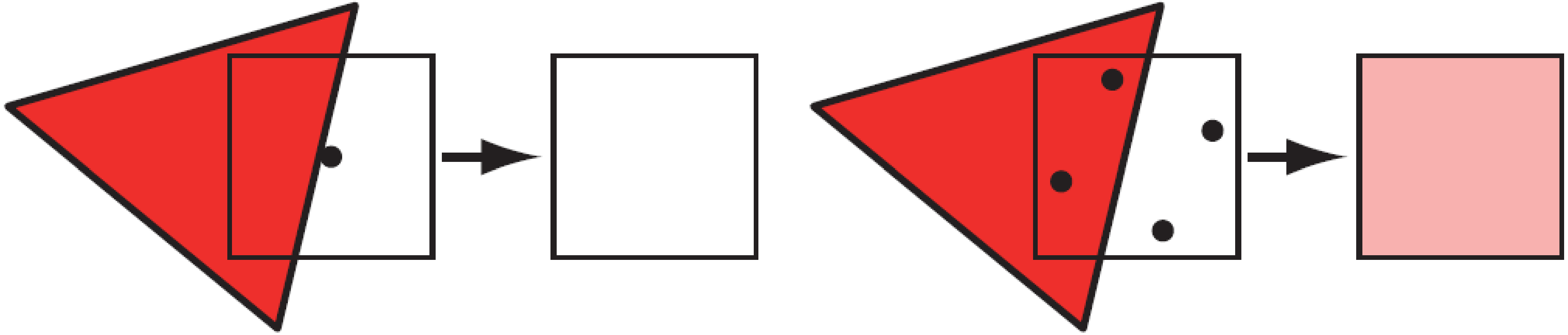
Real-Time Rendering, 4th Ed.

Anti-aliasing

- Techniques to remove aliasing effect.
- Different methods to achieve this:
  - SSAA: Supersampling antialiasing.
  - FXAA: Fast approximate antialiasing.
  - MSAA: Multi-sampled antialiasing.
  - TAA: Temporal antialiasing.
  - SRAA: Sub-pixel Reconstruction antialiasing.

Screen-based antialiasing

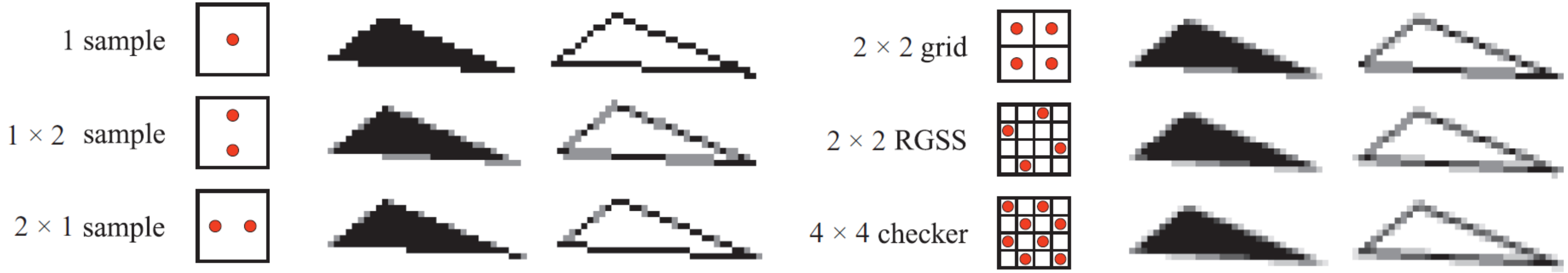
- Operate on the output samples of the graphics pipeline, without any knowledge of the objects being rendered.



$$\mathbf{color}(x, y) = \sum_{i=1}^{n} w_i \mathbf{c}(i, x, y)$$

Supersampling

- Spatial antialiasing method.
- Renders the scene at higher resolution, then average neighboring samples.



Real-Time Rendering, 4th Ed.

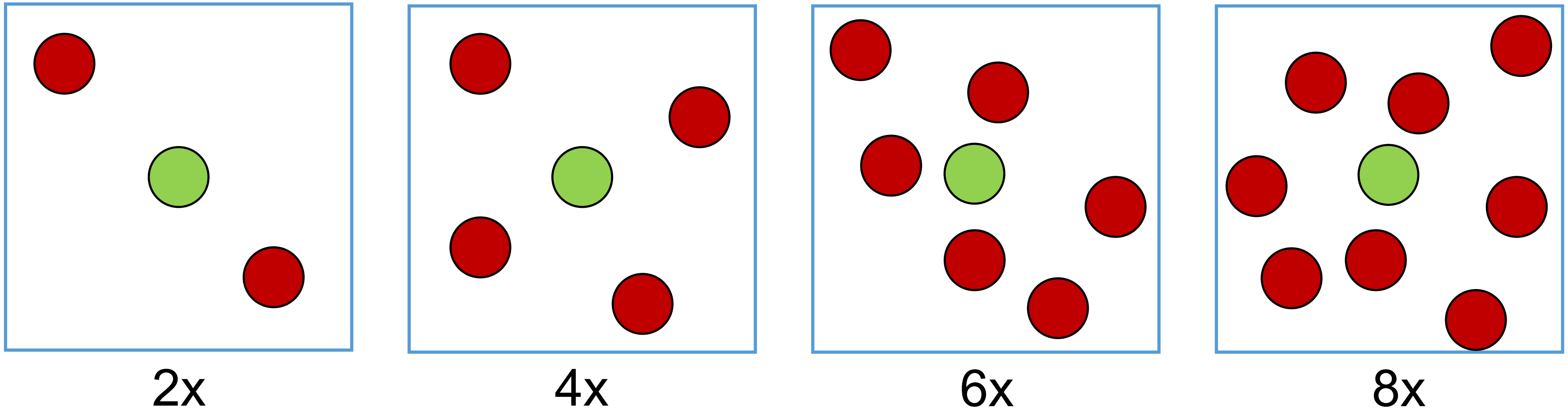
Supersampled antialiasing

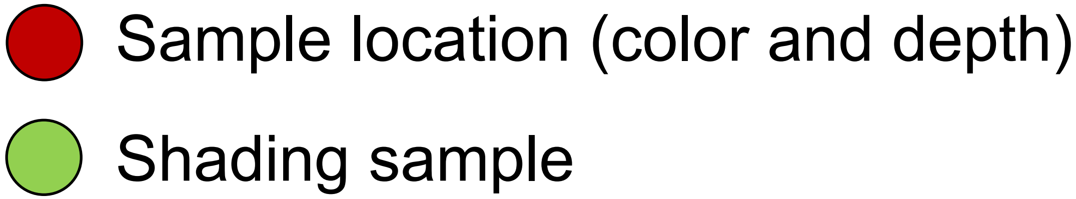
- Render the scene at a higher resolution, down sample with a filter.
- Multiple locations sampled per pixel.
- Example: target is a 1000x800 pixels image; render at 2000x1600 pixels, then average each 2x2 area on the screen.
- Simple, but computationally expensive: amount of buffer used is several times larger. All subsamples must be fully shaded and filled.

Multisampled antialiasing

- Supersampling is prohibitively expensive.
- MSAA is faster than a pure supersampling technique, because the fragment is shaded only once (i.e., shader run once per pixel).
- Observation: aliasing of triangle visibility (geometry aliasing) only occurs at the edges of rasterized triangles.
  - Mipmaps already solve texture aliasing.
  - MSAA only needs to solve geometry aliasing:
    - Increase depth buffer.

Multisampled antialiasing

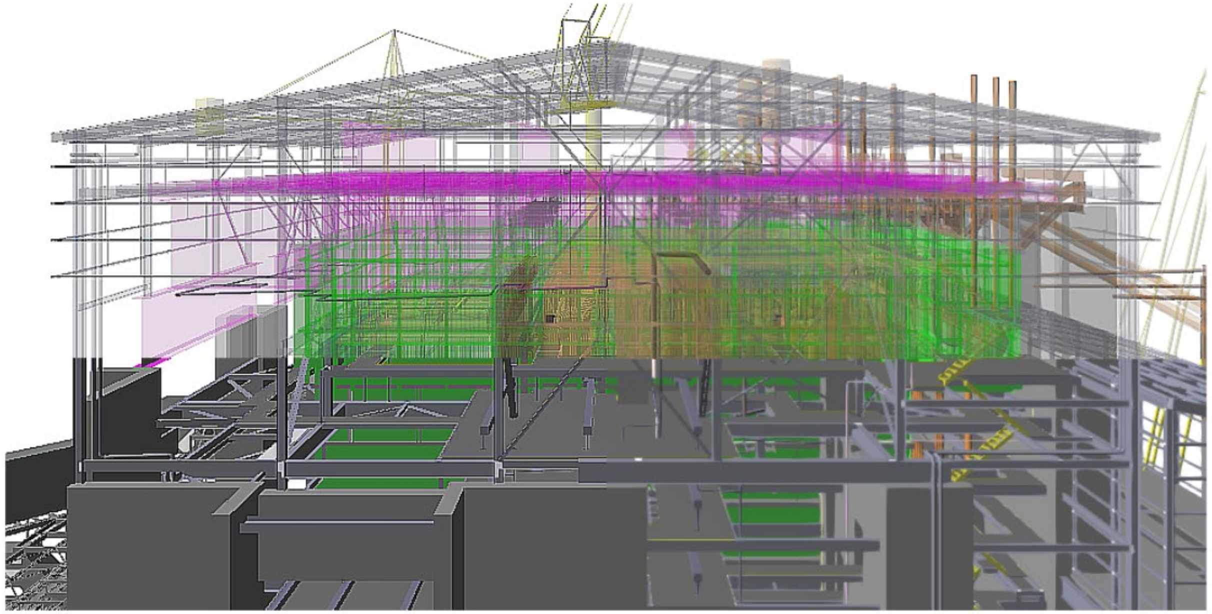


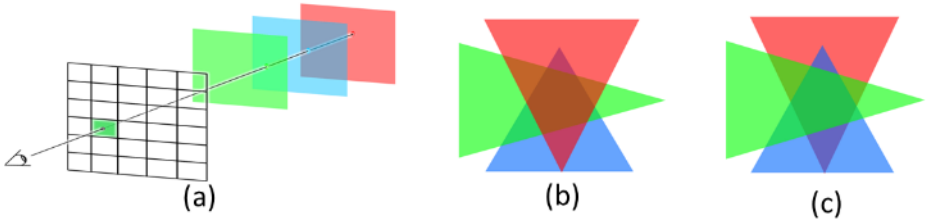


If all samples are covered by fragment, shading sample is in the center of the pixel.

Transparency

- Important to denote relationships among objects in a scene.
- One of the five major challenges in interactive rendering [Andersson, 2010].





If not all fragments are opaque, how can we blend them?

[Maule et al., 2012]

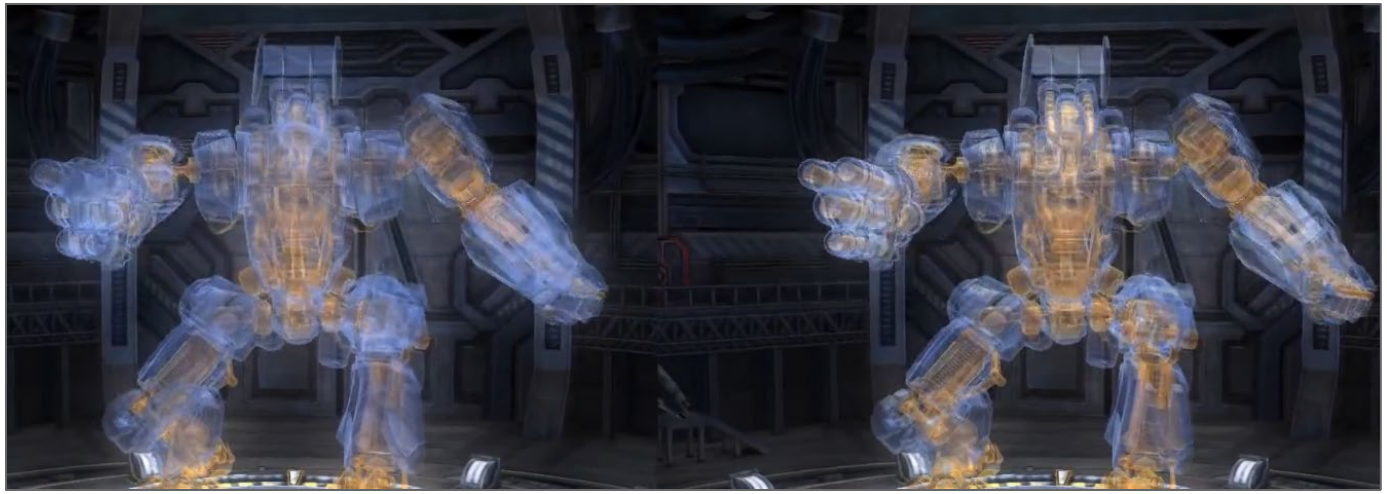
Transparency

- Blend fragment color and opacity such that the resulting pixel color is:

$$\mathbf{c} = \mathbf{c}_n + (1 - \alpha_n) \ldots [\mathbf{c}_2 + (1 - \alpha_2)[\mathbf{c}_2 + (1 - \alpha_1)\mathbf{c}_0]]$$

- Order dependent: final pixel color depends of the fragment color.

Incorrect result as fragments are generated and blended in ‘random’ order.



Correct result, sorting fragments.

ATI Tech Demo

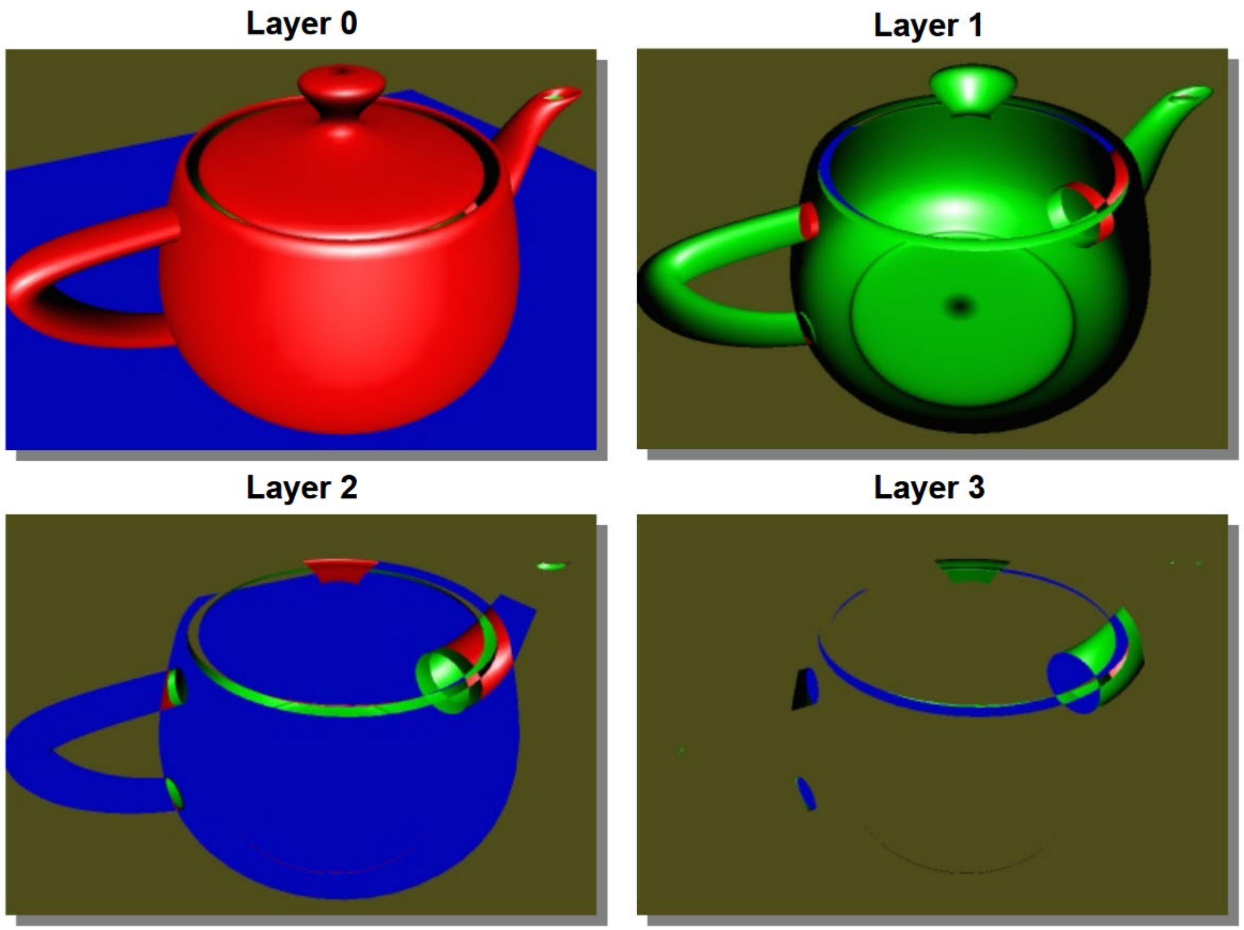
Transparency: basic approach

- Sort the primitives, render and blend the fragments using regular OpenGL pipeline.
- Sort the fragments:
    - Depth peeling [Everitt, 2001].
    - Dual depth peeling [Bavoil and Myers, 2008].
    - Efficient depth peeling via bucket sort [Liu et al., 2009].
    - Stencil routed A-buffer [Myers and Bavoil, 2007].
    - R-buffer [Wittenbrink, 2001].
- Approximations:
    - Stochastic transparency [Enderton et al., 2010].
    - Adaptive transparency [Salvi et al., 2011].

Depth peeling approach

- Order-independent transparency technique.
  - OIT: techniques that do not require rendering geometry in sorted order.
- Uses an implicit sort to extract multiple depth layers.
  - First pass: finds front-most fragment color/depth.
  - Each successive pass: finds the fragment color/depth for the next-nearest fragment on a per-pixel basis.
  - Dual depth buffers to compare previous nearest fragment with current.
- With n passes over the scene, we can get n layers deeper into the scene.

Depth peeling approach



Cass Everitt, NVIDIA

Depth peeling approach

- Create depth texture:

```
this.depthTexture = gl.createTexture();
gl.bindTexture(gl.TEXTURE_2D, this.depthTexture);
gl.texImage2D(gl.TEXTURE_2D, 0, gl.DEPTH_COMPONENT24, width, height, 0, gl.DEPTH_COMPONENT, gl.UNSIGNED_INT, null);

gl.framebufferTexture2D(gl.DRAW_FRAMEBUFFER, gl.DEPTH_ATTACHMENT, gl.TEXTURE_2D, this.depthTexture, 0);
```

- Blend in fragment shader:

```
if(depthPeelPass > 0 && gl_FragCoord.z <= olddepth)
    discard;
frontColor.rgb += color.rgb * color.a * alphaMultiplier;
frontColor.a = 1.0 - alphaMultiplier * (1.0 - color.a);
```

Linked-list approach

- Depth peeling: requires multiple (n) peeling passes.
- Linked-list approach: “collect” all fragments, then sort them and draw, but with a dynamic amount of memory (as we don’t know how many fragments a scene will generate).

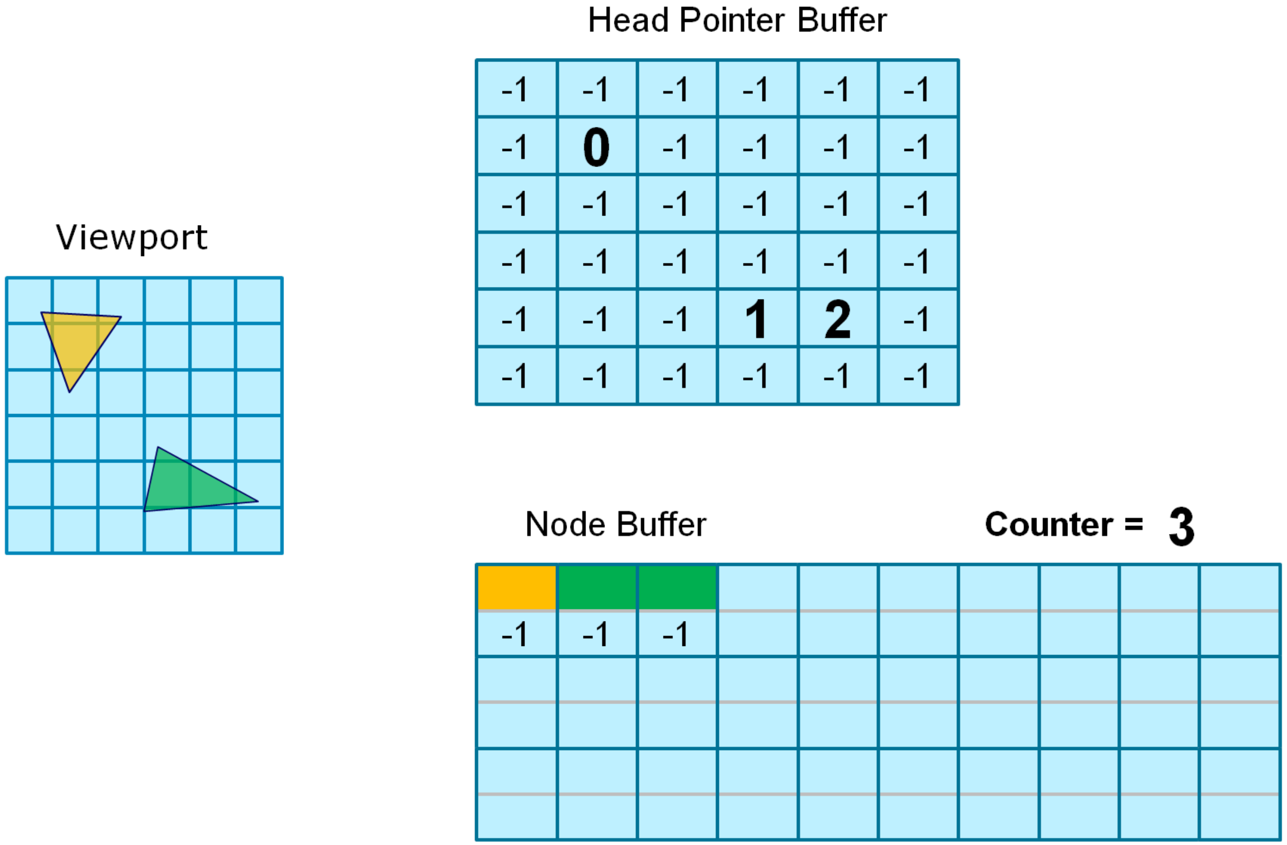
Linked-list approach

- Real-Time Concurrent Linked List Construction on the GPU [Yang et al., 2010]:
  - Two buffers: head pointer, node buffer.
  - Uses **atomic** operations to append to node buffer and increase counter. New fragment points to the previous fragment.

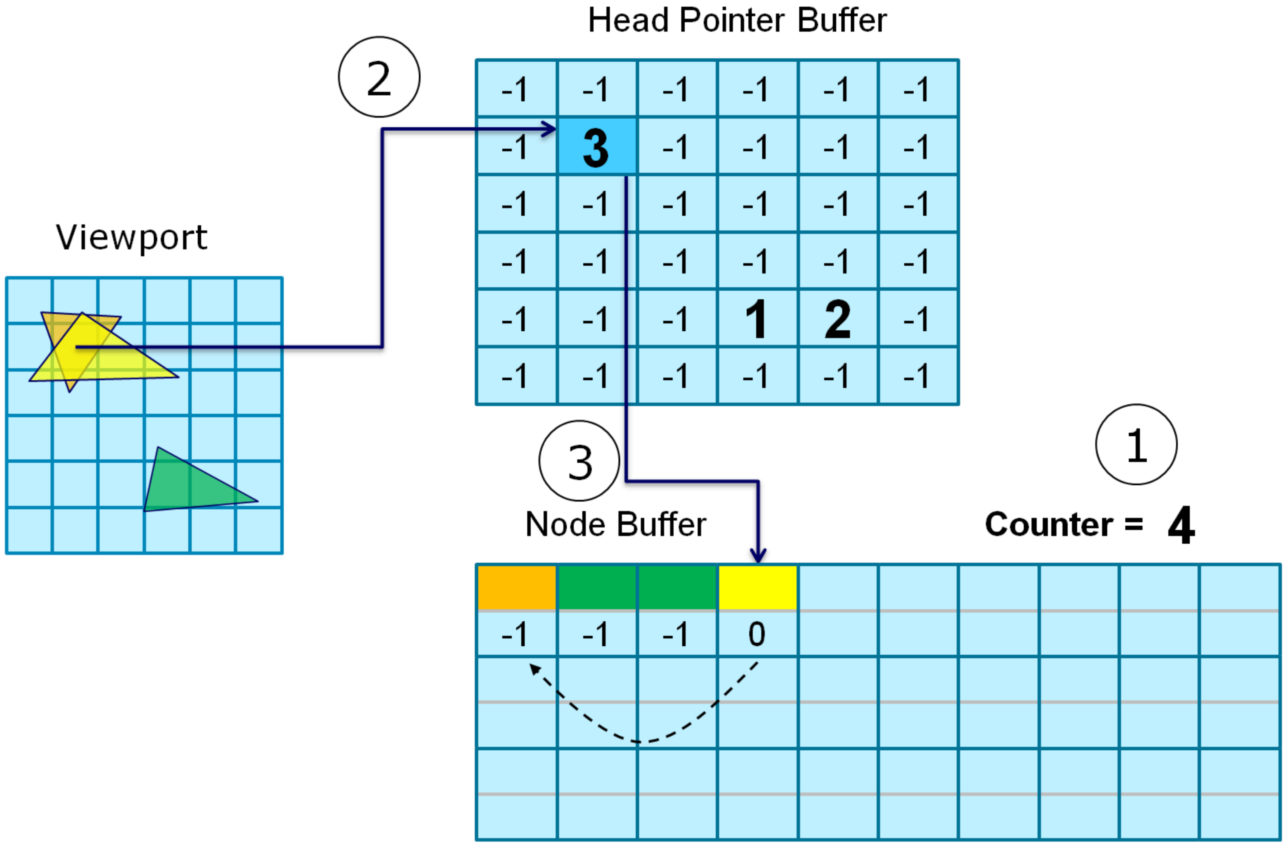
```
uint baseindex = uint(texture(texscan, (gl_FragCoord.xy-vec2(1,0)) / 800).a);
uint offset = imageAtomicAdd(img_offsetlookuptable, ivec2(gl_FragCoord.xy), 1);
int index = int(baseindex + offset);

imageStore(img_fragredtable, index, vec4(shade.r));
```

Linked-list approach



Linked-list approach



Linked-list approach

- The fragments are inserted out of order, so what about sorting?
  - Sorting is done after the linked list is constructed.
  - Yang et al. use an insertion sort, considering that each list contains few fragments.
  - Faster than depth peeling by about 10x.