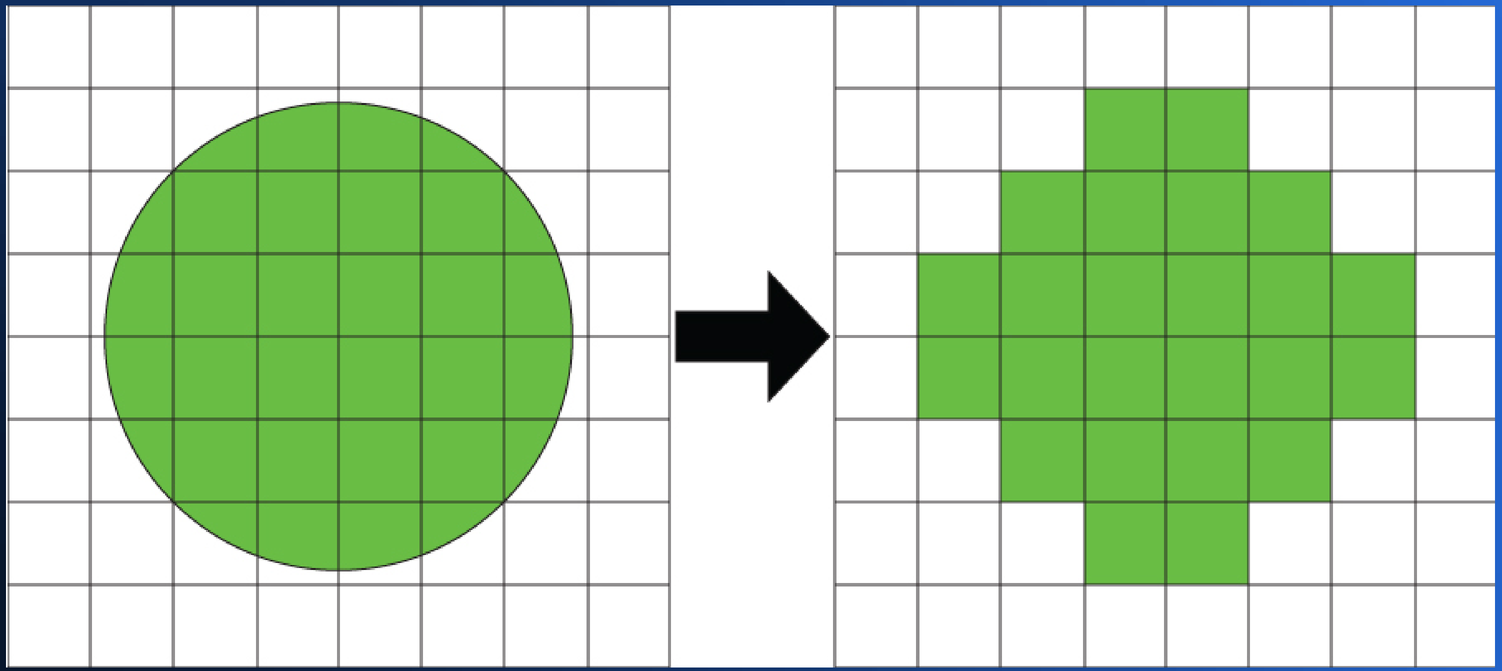
# Anti-Aliasing

Adam Mally


http://doom.wikia.com/wiki/Imp

# What is Aliasing?

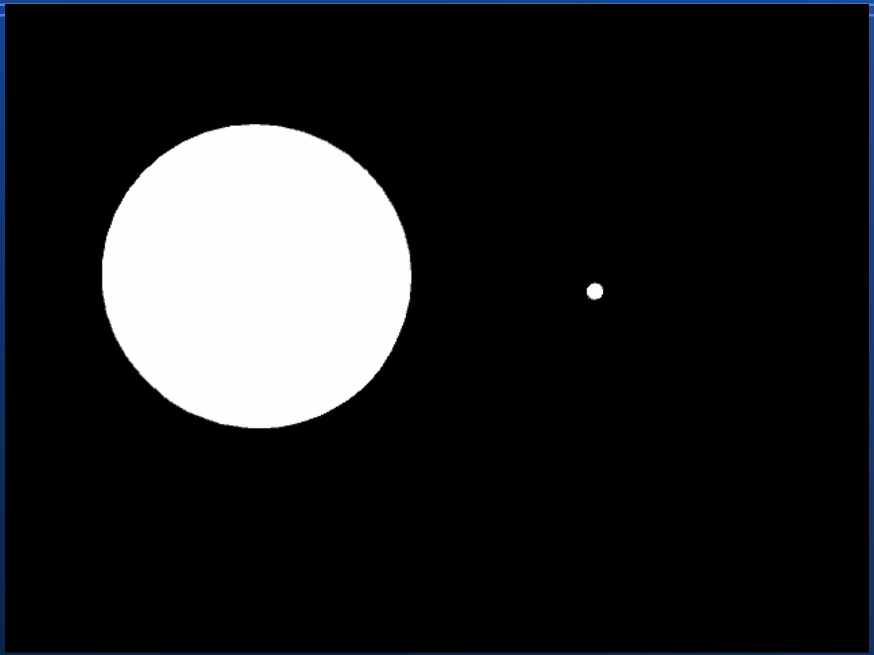
- The result of projecting objects in world space onto a screen grid (pixels)



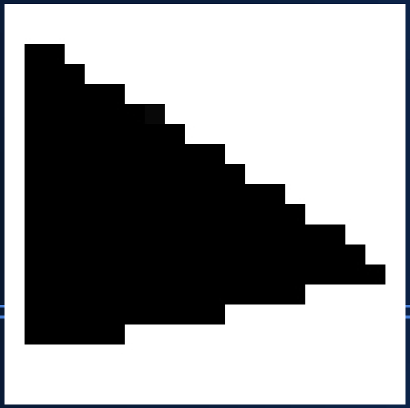
# What is aliasing?

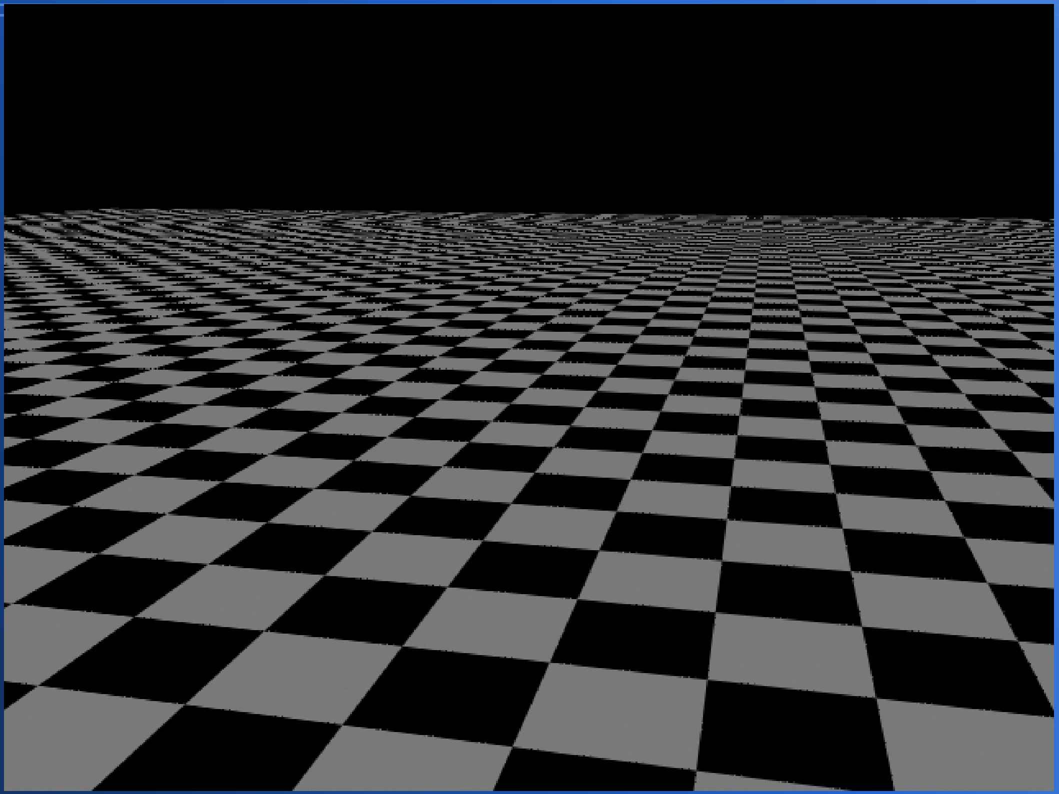
- Screen resolution not as high as that of the human eye
- Sacrifice image precision because of hardware limits
- Aliasing causes still images to look jagged and moving images to look “crawly”
- Should a screen's pixels be dense enough, no anti-aliasing would be needed

# What is aliasing?



Hard edges





Texture disintegration

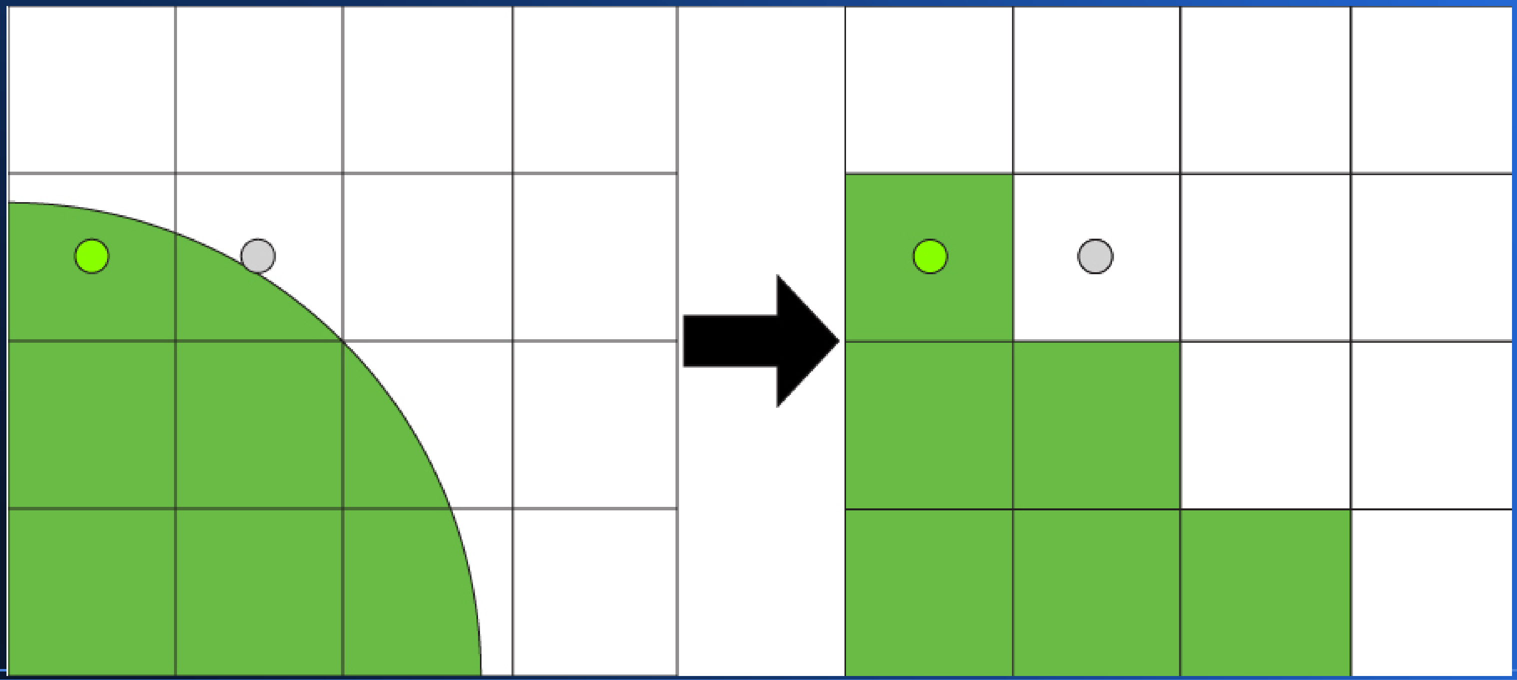
# What is aliasing?


Subpixel elements lost on left, regained on right

http://www.codinghorror.com/.a/6a0120a85dcdae970b015437fee512970c-800wi

# What causes aliasing?

- In the most basic methods of rendering to the screen, the center of each screen grid cell is tested to see if it contains a polygon



# What causes aliasing?

- If the cell center does intersect with a polygon, the corresponding pixel is colored accordingly
- This binary approach to pixel coloring is the root cause of aliasing
- Aliasing also occurs within textures for the same reason

# How do we combat aliasing?

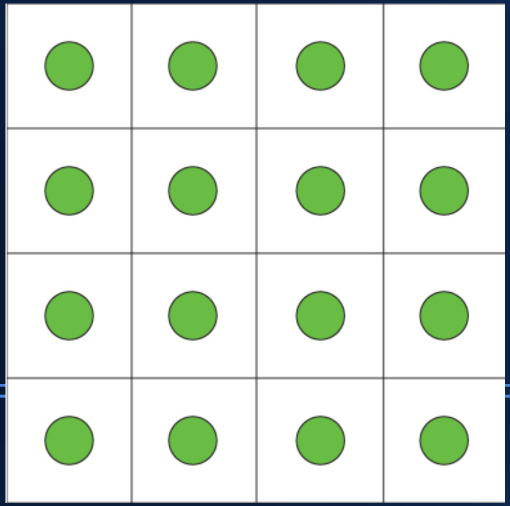
- By using anti-aliasing techniques, of course!
- Numerous AA methods, varying in time cost and effectiveness
- No method is perfect; the only way to eliminate aliasing completely is to have more pixels (higher resolution screen)

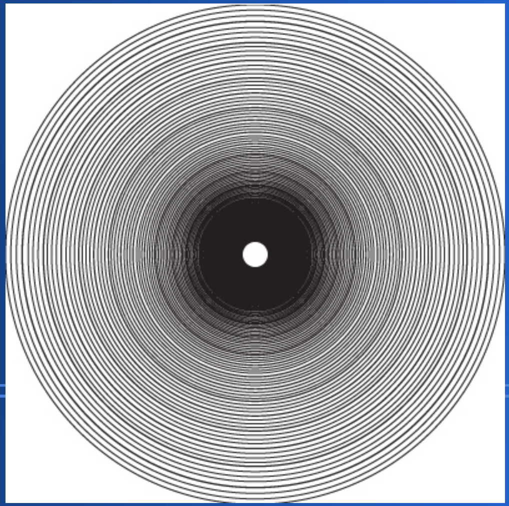
# Super-Sampling Anti-Aliasing

- For each cell of the screen grid, use multiple samples to determine the corresponding pixel's color
- Averages n color samples within the cell to determine its color, where n is an integer defined by the user or method of SSAA

# Super-Sampling Anti-Aliasing – Grid

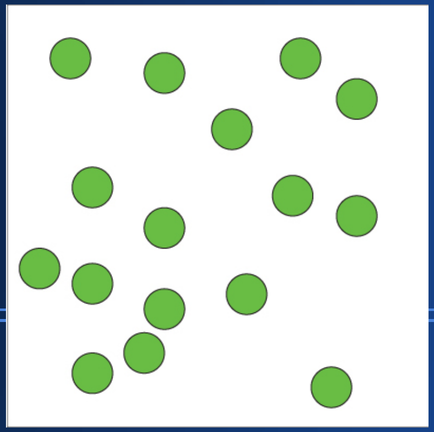
- The simplest SSAA method is grid sampling
- Take n samples evenly distributed across the cell
- Fastest SS method but can cause moiré patterns to appear in areas of dense texture





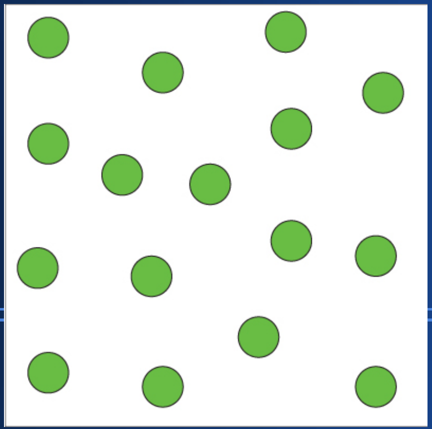
# Super-Sampling Anti-Aliasing – Random

- A simple way to reduce or remove moiré patterns is to sample randomly instead
- Take n samples randomly distributed across the cell
- About as fast as the grid approach, but is not deterministic and may produce lopsided samples



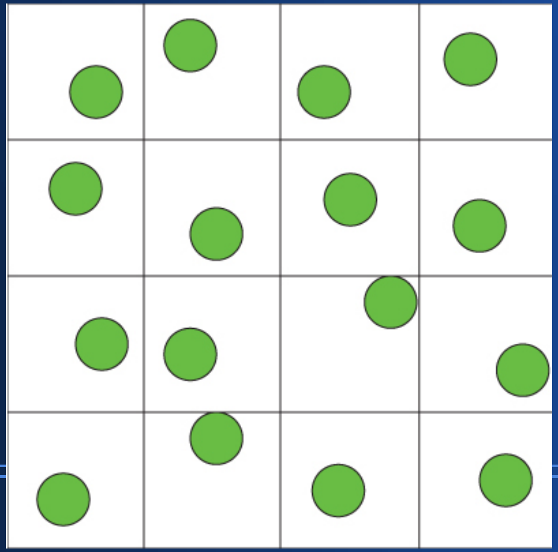
# Super-Sampling Anti-Aliasing – Poisson Disc

- A method of counteracting the potential lopsidedness of random sampling
- Take n randomly distributed samples, but ensure that no two samples are within m units of one another
- Too costly for real-time use



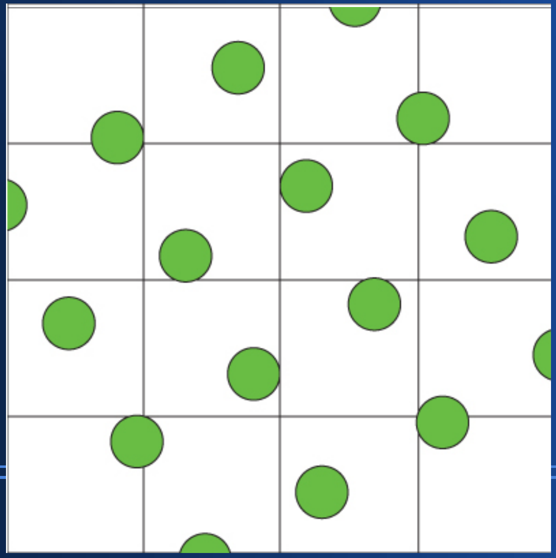
# Super-Sampling Anti-Aliasing – Jitter

- A more randomized approach to grid sampling
- Divide the screen grid cell into n subcells
- Rather than sampling from the center of each cell, sample from a random location within the cell



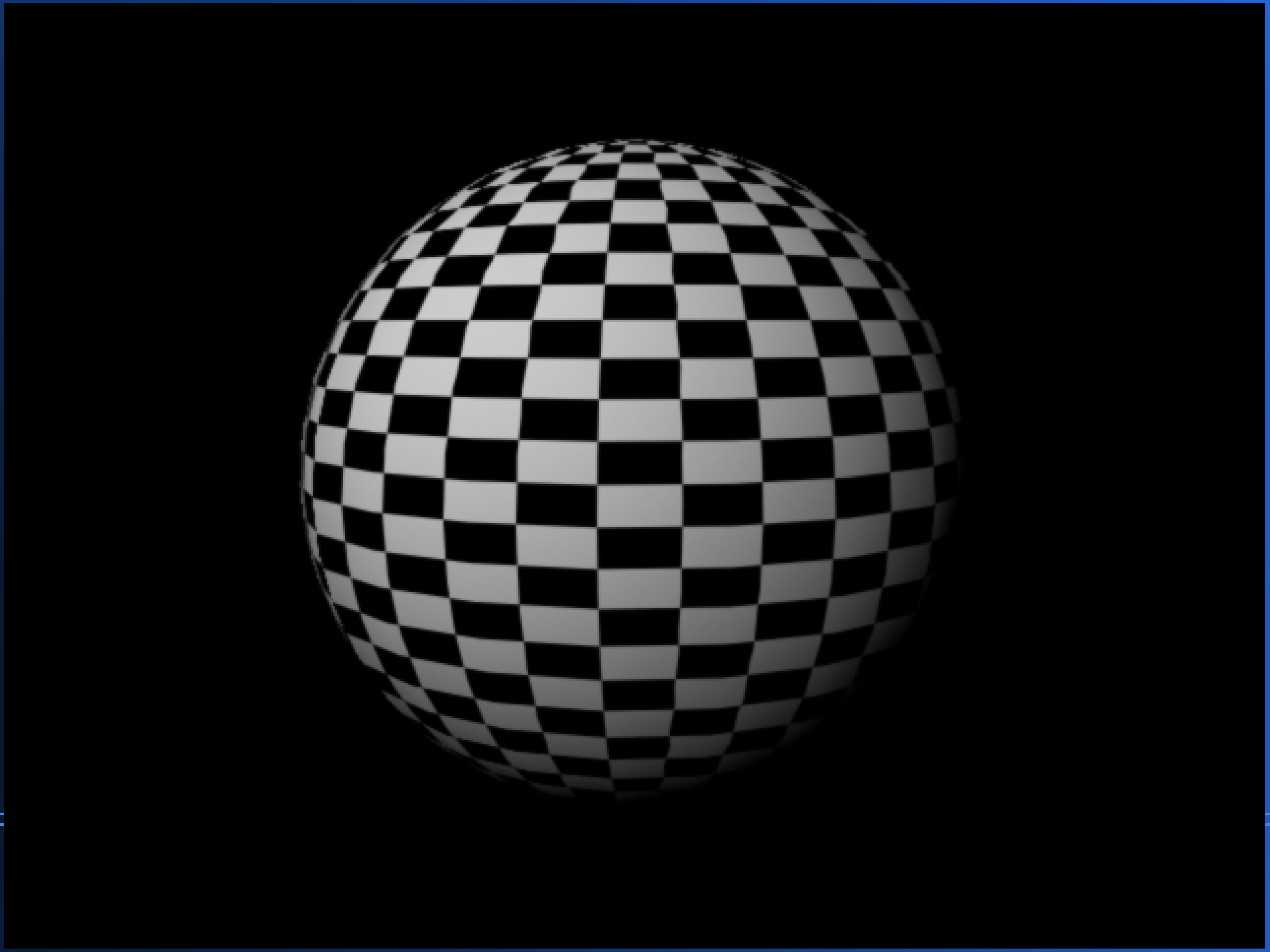
# Super-Sampling Anti-Aliasing – Rotated Grid

- Similar to grid sampling, but rotate the sample positions off-axis
- Helps eliminate aliasing along the X and Y axes, where it is most apparent to humans



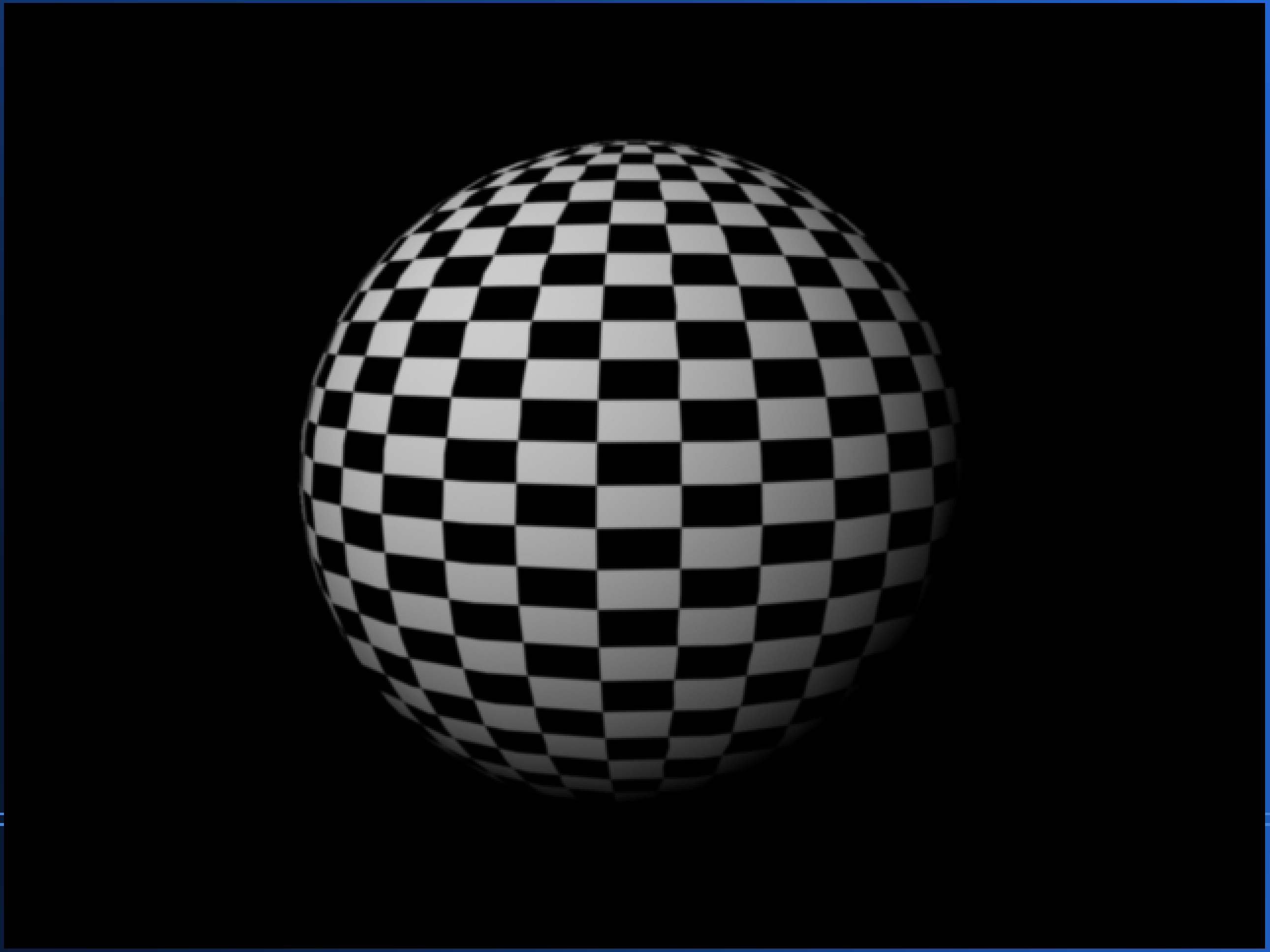
# Super-Sampling Anti-Aliasing – Example

- The image below cycles through increasing sample sizes for a simple grid SSAA: 1 sample, ~ 1 second



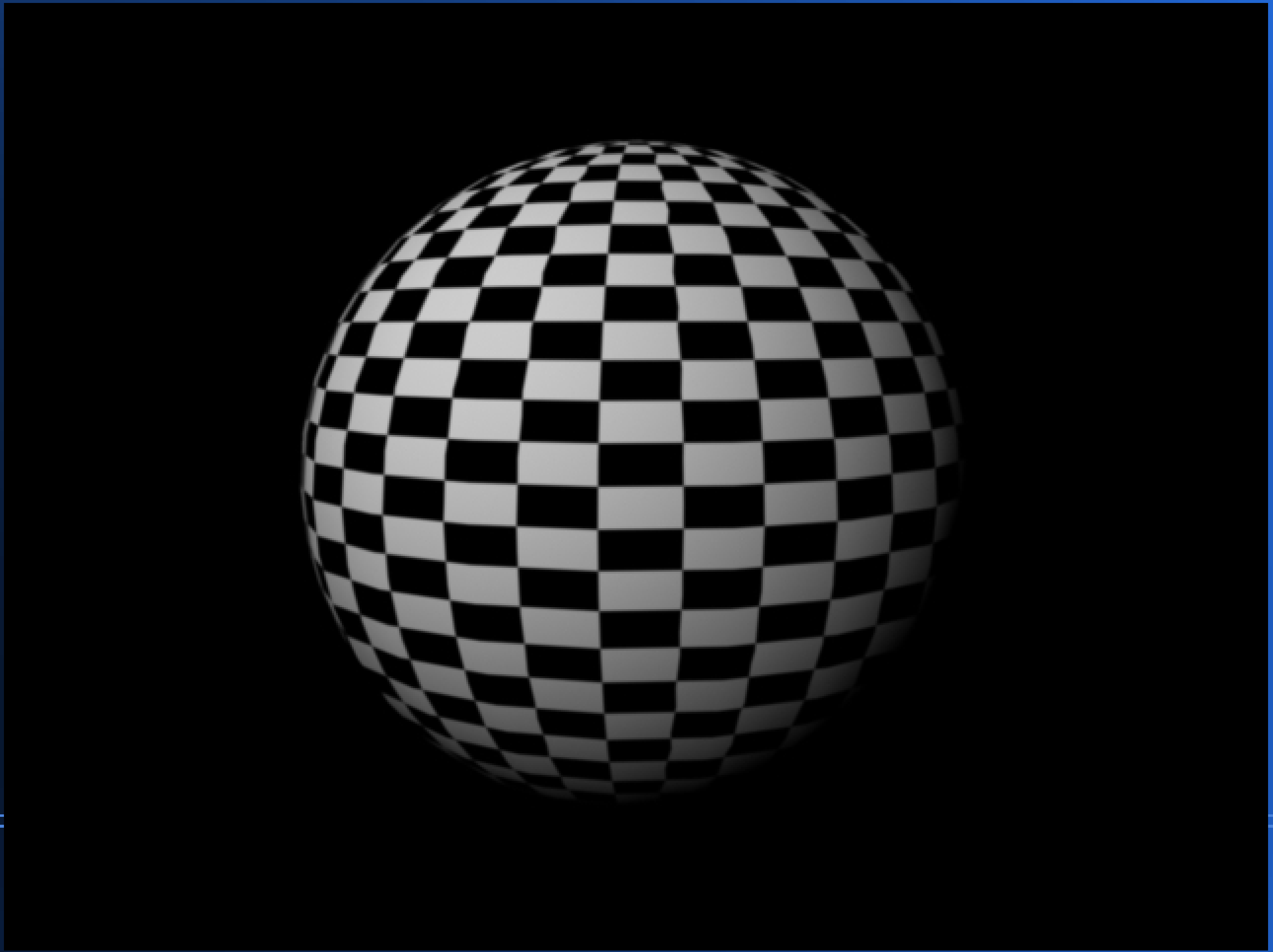
# Super-Sampling Anti-Aliasing – Example

- The image below cycles through increasing sample sizes for a simple grid SSAA: 4 samples, ~ 2 seconds



# Super-Sampling Anti-Aliasing – Example

- The image below cycles through increasing sample sizes for a simple grid SSAA: 16 samples, ~ 4 seconds



# Multisample Anti-Aliasing

- An optimized form of super-sampling
- Use one of the previously mentioned sampling patterns
- Check each sample position versus each fragment
- If all samples are covered by a fragment, sample from the center of the pixel
- Else, shift the sampling position based on how many samples are covered
- Most importantly, each fragment's shade is computed only once, even if multiple samples intersect it

# Coverage-Sample Anti-Aliasing

- An optimized form of multisampling
- Created by nVidia
- Claims to have AA results equivalent to 16-sample MSAA for only the cost of 4-sample MSAA
- In addition to reducing the number of fragments computed as with MSAA, also reduces the number of color, z-depth, and stencil samples while maintaining a high number of coverage samples
- Reduces computation cost for each pixel due to fewer non-coverage samples

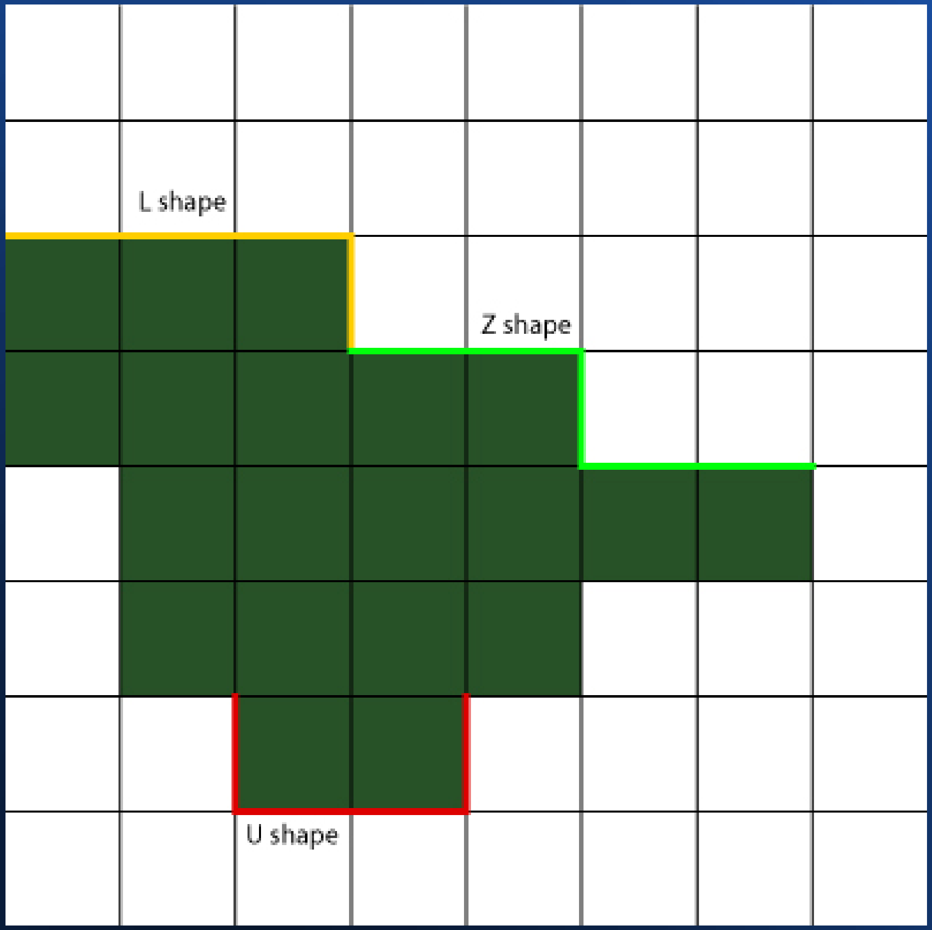
# Custom Filter Anti-Aliasing

- Created by AMD
- Samples from a radius around a pixel rather than just within the pixel
- Also uses an edge detection algorithm to aid in concentrating anti-aliasing effects near areas of distinct color change
- Effectively searches for areas where aliasing is most common (edges)

# Morphological Anti-Aliasing

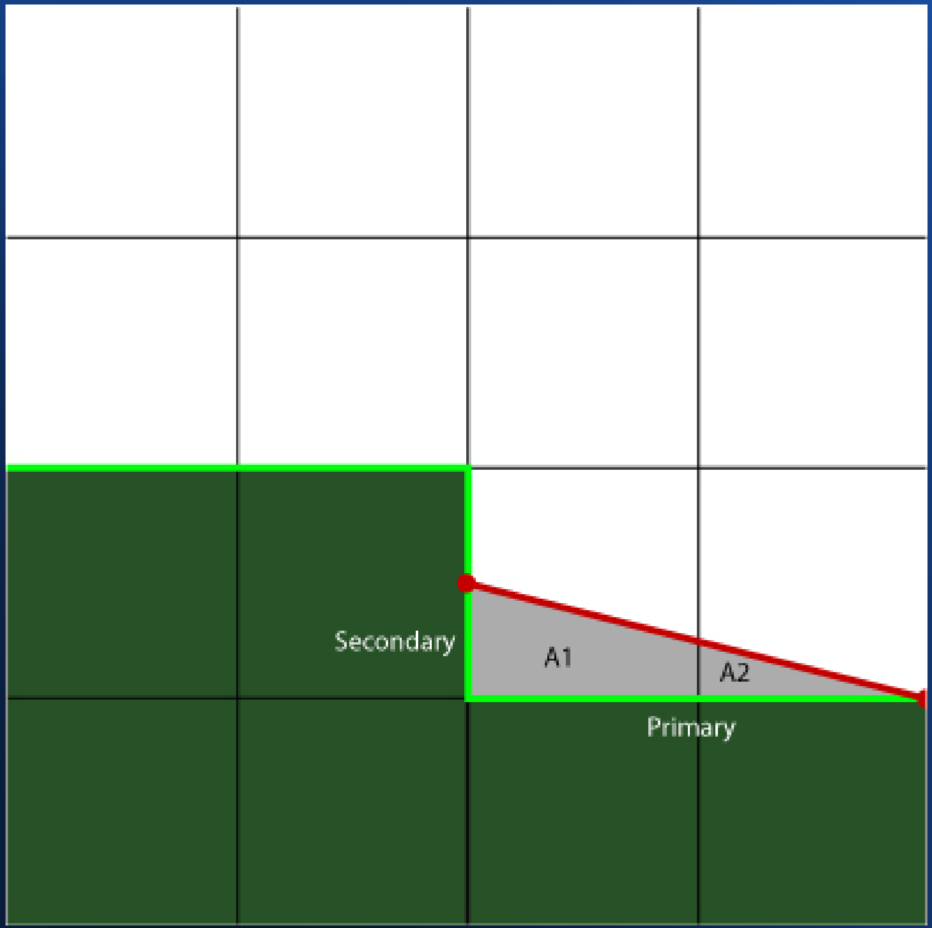
- Multi-pixel anti-aliasing rather than single-pixel
- Use an algorithm to find edge shapes between different colors
- Use these edge shapes (L, Z, and U) to determine the shading of bordering pixels.

# Morphological Anti-Aliasing



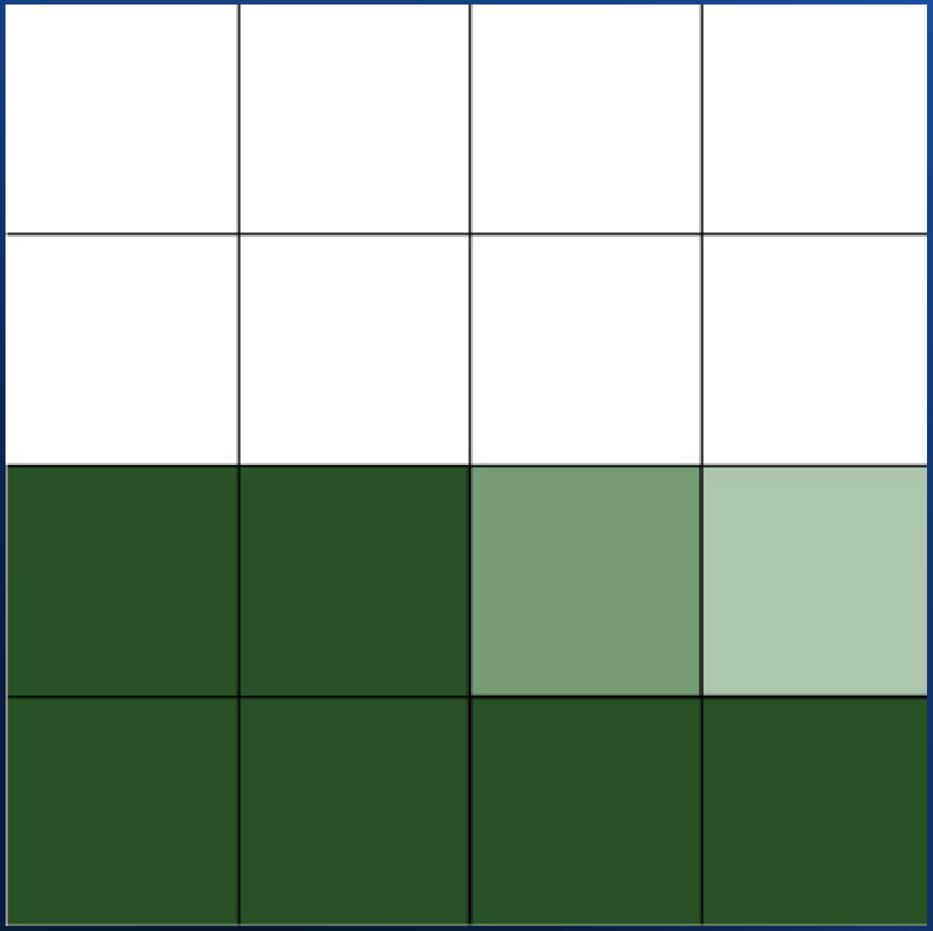

- Step 1: Find all edges between pixels of different colors
- Step 2: For each found edge, find dividing edges that are orthogonal to it and touch one of its endpoints
- Step 3: Process each shape, decomposing Z shapes and U shapes into pairs of L shapes

# Morphological Anti-Aliasing




- Each L shape is comprised of two edges: a primary edge and a secondary edge.
- Primary was the edge found in step 1, while secondary was the edge found in step 2
- Create a triangle from the midpoint of secondary and the outer point of primary.

# Morphological Anti-Aliasing



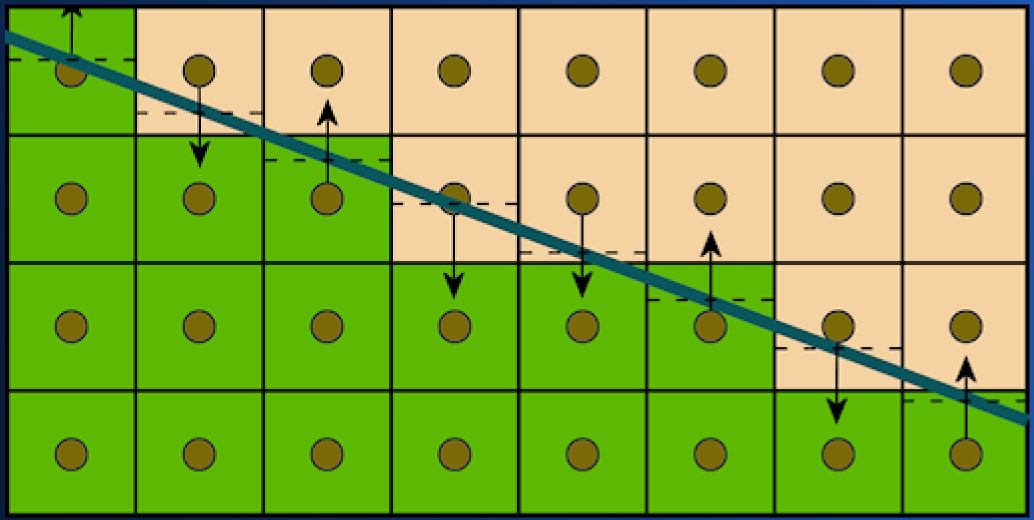
- The color of a cell covered by a trapezoid or triangle is modified as follows:
- cell_color = (1 – area) * current_color + area * new_color
- These triangles and trapezoids simulate a possible form for the object being represented by the pixels
- The computed shade represents this form by averaging out the color of the subpixel form across the entire pixel

# Subpixel Morphological Anti-Aliasing

- While MLAA helps to regain the form of a pixelized shape, it tends to wear down sharp corners due to the assumptions it makes about L shapes
- This occurs because MLAA assumes there as only one pixel sample and that it was taken at the center of the pixel
- SMAA attempts to mitigate corner loss by checking L-shape side lengths (the longer they are the more of a corner a pixel area will be)
- SMAA also mitigates overly fuzzy diagonals by implementing searches for roughly 45-degree lines in addition to the horizontal/vertical searches from MLAA
- SMAA also takes into account all subpixel samples when calculating trapezoids, averaging the different areas of the vectorizations

# Geometric Post-Process Anti-Aliasing

- Initially render the scene with no anti-aliasing
- This frame is then copied to a texture, where geometry edges are matched to pixels and color is blended as follows:
- The centroid of a pixel touched by an edge is compared to the edge and shifted toward the edge, blending color with the pixel in that direction with attenuation based on the distance shifted



http://www.humus.name/index.php?page=3D

# Geometric Post-Process Anti-Aliasing

- Edge smoothing is very efficient
- Cost comes from copying the aliased frame (backbuffer) to a texture
- Main disadvantage is that geometric information is needed in order to smooth edges
- Additionally, will only smooth geometric edges
- Textures are not affected by this particular AA method

# Anti-Aliasing in Raytracing vs. Rasterization

- When sampling with a raytracer, each sample must represent a ray cast
- n samples means n raycasts
- Each raycast is in 3D, which means reflections and intersection tests
- Very costly; generally too inefficient for real-time

- When sampling with a rasterizer, each sample is just a fragment
- Fragments are already two-dimensional, so sampling them is as simple as computing their shade
- Very fast compared to raytracing

# Anti-Aliasing in Games

- Multi-sample anti-aliasing was widely used
- Games are rendered with rasterization, and MSAA works extremely well and efficiently with rasterization
- Coverage sample and custom filter AA were and are widely used as well, since they are specialized MSAA methods
- Morphological AA
- More sophisticated AA methods, such as subpixel morphological AA, are being developed and put into use

# References


- Real-Time Rendering (Third Edition) by Akenine-Moller, Haines, and Hoffman
- Morphological Antialiasing by Alexander Reshetov, Intel Labs
- SMAA: Enhanced Subpixel Morphological Antialiasing by Jorge Jimenez, Jose I. Echevarria, Tiago Sousa, and Diego Gutierrez
- http://www.tomshardware.com/reviews/anti-aliasing-performance,3065-19.html
- http://doom.wikia.com/wiki/Imp
- http://strattonbrazil.blogspot.com/2009/04/ray-tracing-simpler-than-rasterizing.html
- http://www.codinghorror.com/.a/6a0120a85dcdae970b015437fee512970c-800wi
- http://visual-computing.intel-research.net/publications/papers/2009/mlaa/mlaa.pdf
- http://www.iryoku.com/smaa/
- http://www.tested.com/news/how-to/articles/1194-how-to-choose-the-right-anti-aliasing-mode-for-your-gpu/2/