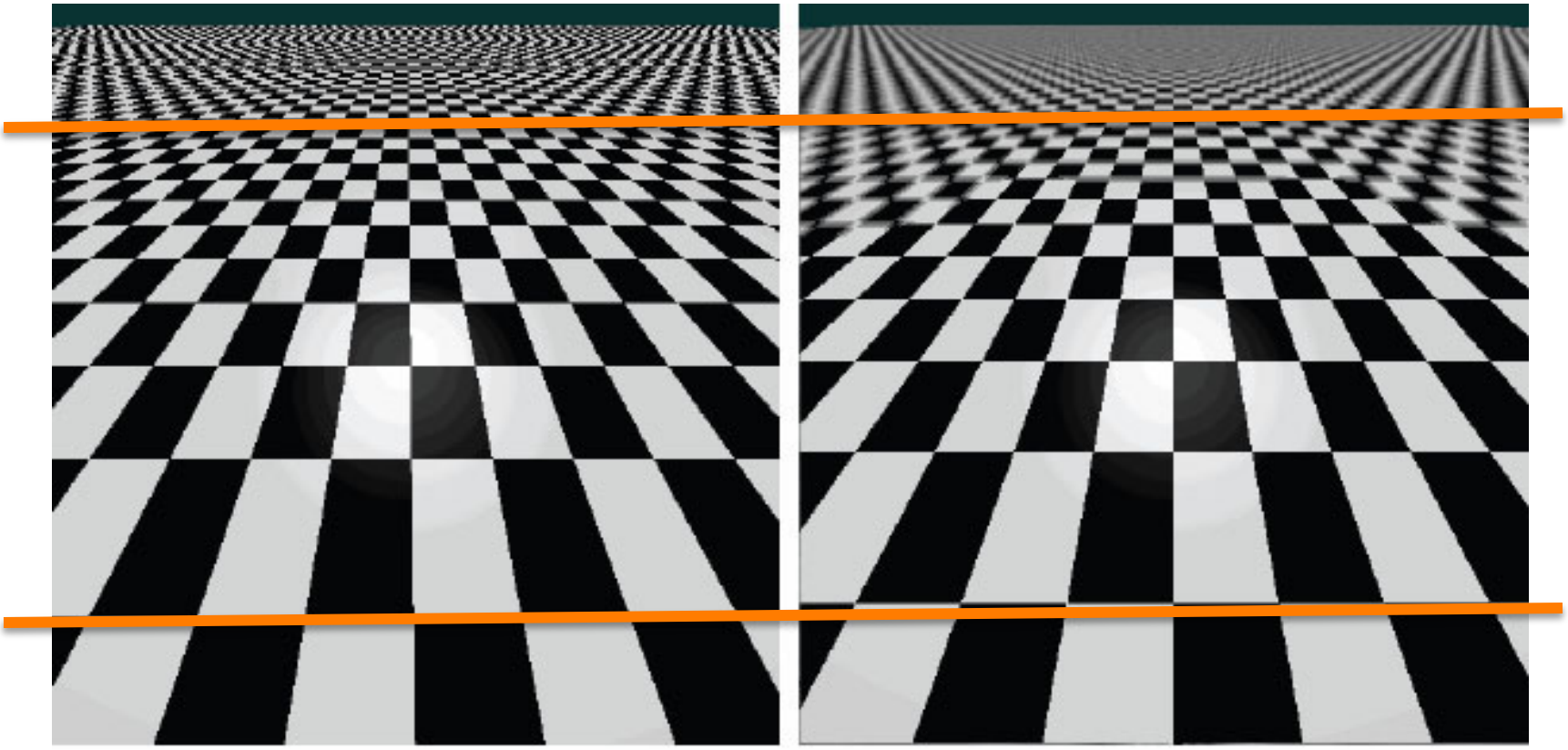
DigiPen INSTITUTE OF TECHNOLOGY

CS300
Anti-Aliasing

Introduction

- What is aliasing?

Example



Introduction

- Classically, it is an artifact when high frequency information is 'aliased' as low frequency information.
- What causes it?

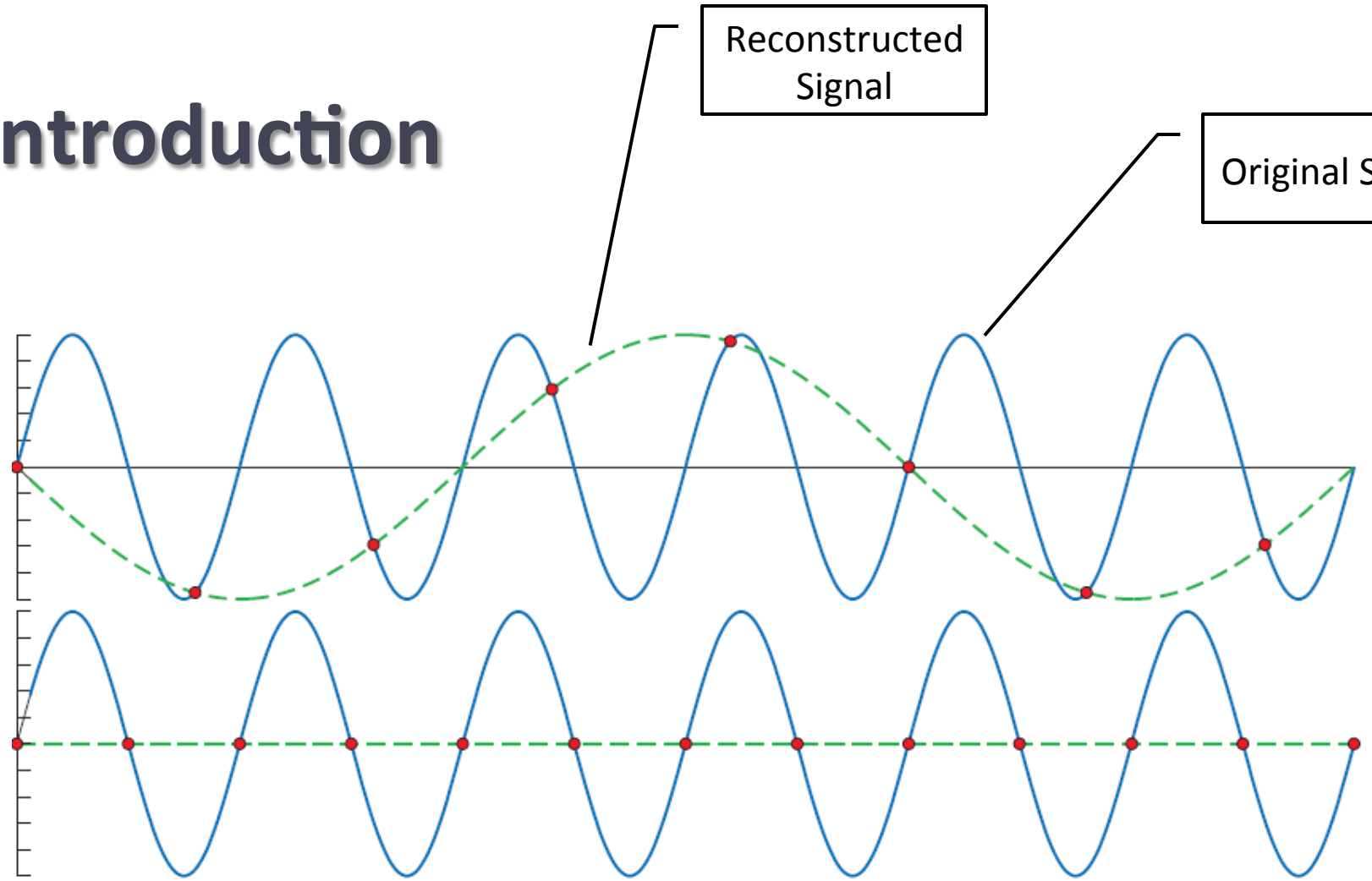
Introduction

- Classically, it is an artifact when high frequency information is 'aliased' as low frequency information.
- Inadequate sampling or poor sampling of continuous data that result in an inability to reconstruct the data correctly.
- How to sample the continuous data?

Introduction

- Classically, it is an artifact when high frequency information is 'aliased' as low frequency information.
- Inadequate sampling or poor sampling of continuous data that result in an inability to reconstruct the data correctly.
- If the sampling rate is more than twice the highest frequency in the data, then the data can be reconstructed correctly. The sampling rate is called *Nyquist rate*.

Introduction



Aliasing in Computer Graphics

- So, what is aliasing in computer graphics?

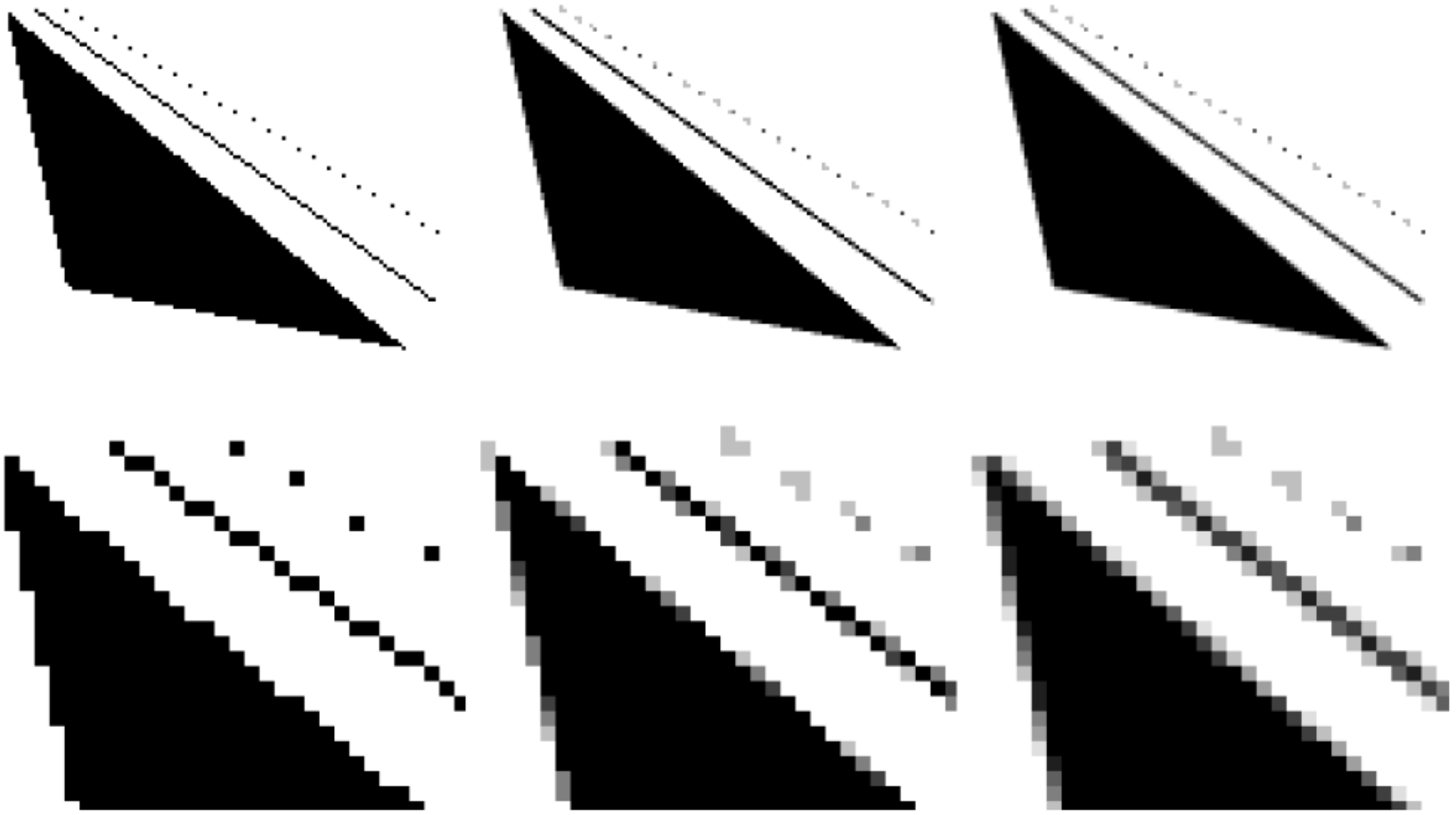
Aliasing in Computer Graphics

- In computer graphics, the term ‘aliasing’ is used rather loosely.
- It generally means any unwanted artifacts in the image.
- What causes aliasing in computer graphics?

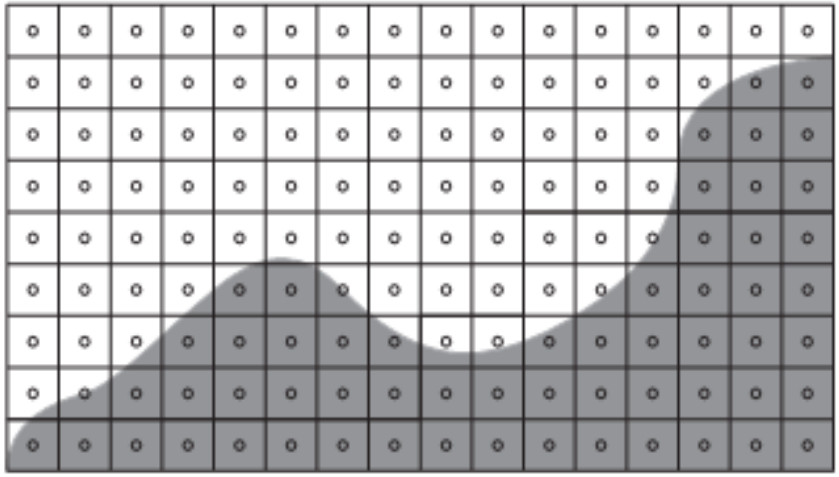
Aliasing in Computer Graphics

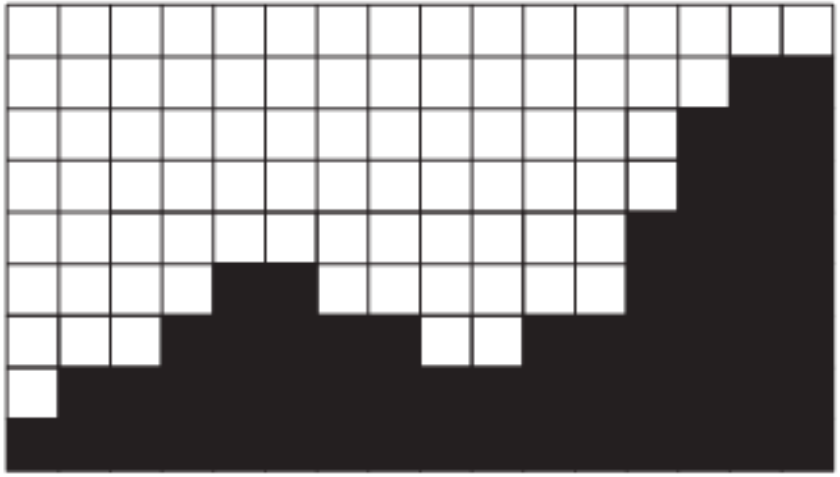
- In computer graphics, the term ‘aliasing’ is used rather loosely.
- It generally means any unwanted artifacts in the image.
- Sampling rate is limited to screen resolution. As such, any information that is at higher frequency than the screen resolution will most likely exhibit aliasing problem.

Example



Example of aliasing





Aliasing in Computer Graphics

- Typical aliasing in computer graphics:
  - Jagged edges
  - Disintegrating texture details
  - Animation flickering (temporal aliasing)

- Simple solution is to not introduce frequency aliases during rasterization. Either increase the sampling resolution to match the data or remove the high frequency component that would be aliased before sampling.

Temporal Aliasing

- Wagon wheel effect
- https://youtu.be/QYYK4tlCMlY

- Objects appear stationary or moving backwards if the speed of their motion exceeds the capture framerate

Anti-aliasing Techniques

- The following are some anti-aliasing techniques employed to reduce aliasing:
  - Supersampling
  - Adaptive supersampling
  - Multisampling
  - Edge blurring
  - Area sampling
  - Texture mipmaps
  - Motion blur

Aliasing

Supersampling

- The most popular anti-aliasing technique
- It works by sampling the images at a higher resolution than the color buffer, then post-processes these extra samples to produce the final colors.
- The high resolution pixels are called *supersamples*.
- This techniques does not eliminate aliasing. Why?

Supersampling

- The super sample itself contains aliased information from frequency higher than the higher sampling rate, albeit at lesser magnitude.
- Things to be decided in super sampling algorithm:
  - number of samples
  - sample position (sample pattern)
  - post-filter method

Supersampling Sample Count

- More samples lead to better result.
- However, increasing the number of samples means increasing the memory and bandwidth requirement.
- Also, if an 'expensive' fragment shader is being used, things could get really slow.
- Typical number of samples are 4 to 16. Although, it is not uncommon to have a much higher number of samples for non-realtime application.

Supersampling Sample Pattern

- Simple choice would be a uniform grid. Unfortunately, it produces poor result. Why?

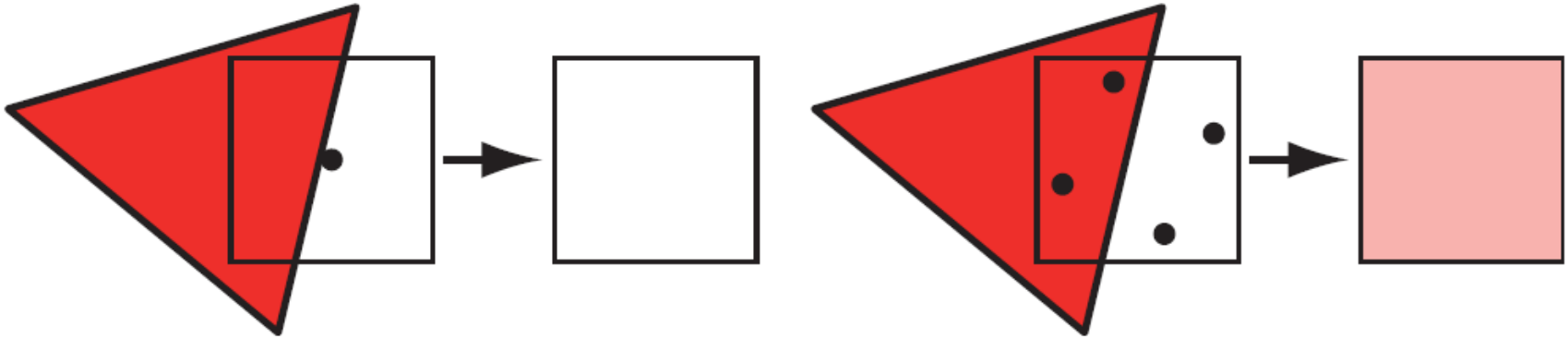
Supersampling Sample Pattern

- Simple choice would be a uniform grid. Unfortunately, it produces poor result.
- The remaining aliasing artifacts tend to occur in regular pattern that are more noticeable to our eyes than error from random spacing.
- Use a more random sample pattern to remedy the situation.

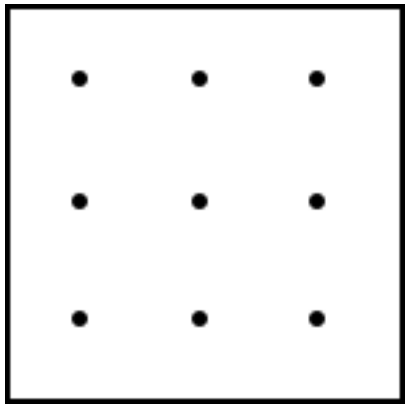
Supersampling Implementation

- Two methods to implement super sampling:
  - Render the scene at higher (4x, 16x, etc) resolution and then scale down the result to the final viewport size.
  - Multi pass super sampling
    - Let N be the number of sample for each pixel
    - For each sample position
      - Adjust the camera position accordingly
      - Render the scene
      - Add 1/N of the result to the accumulation buffer
    - Use the accumulation buffer content as the final result

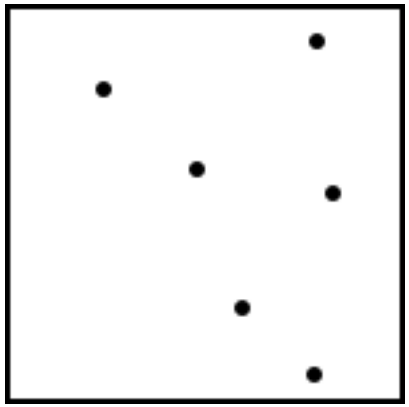
Random Samples



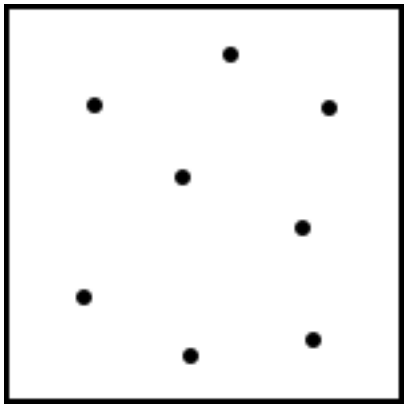
Supersampling



Grid based

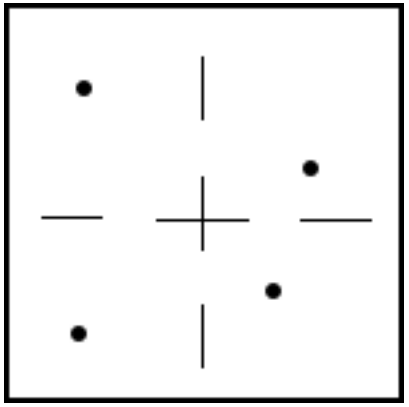


Random

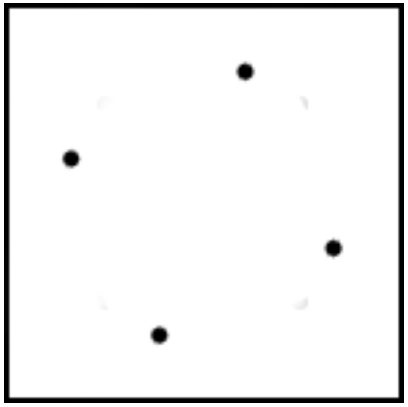


Poisson

Supersampling (2)

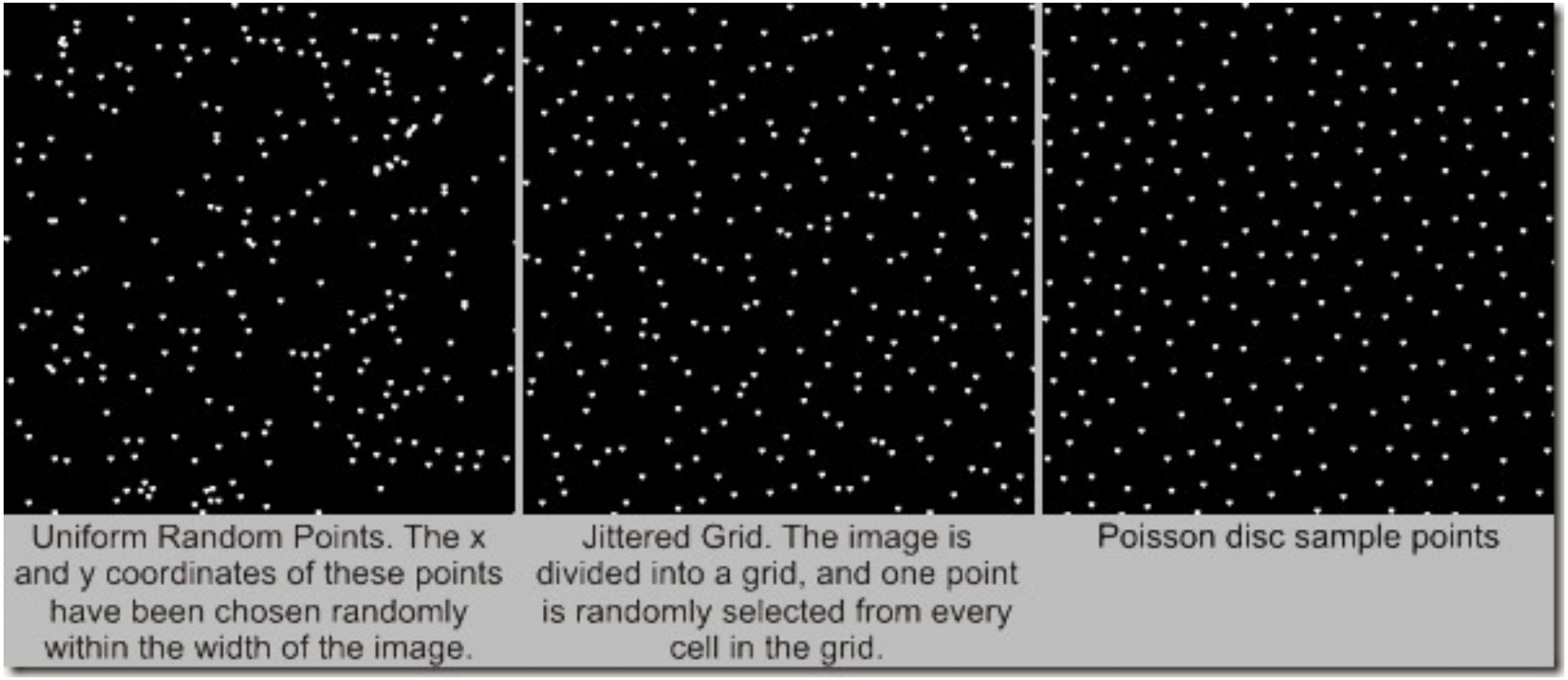


Jitter algorithm

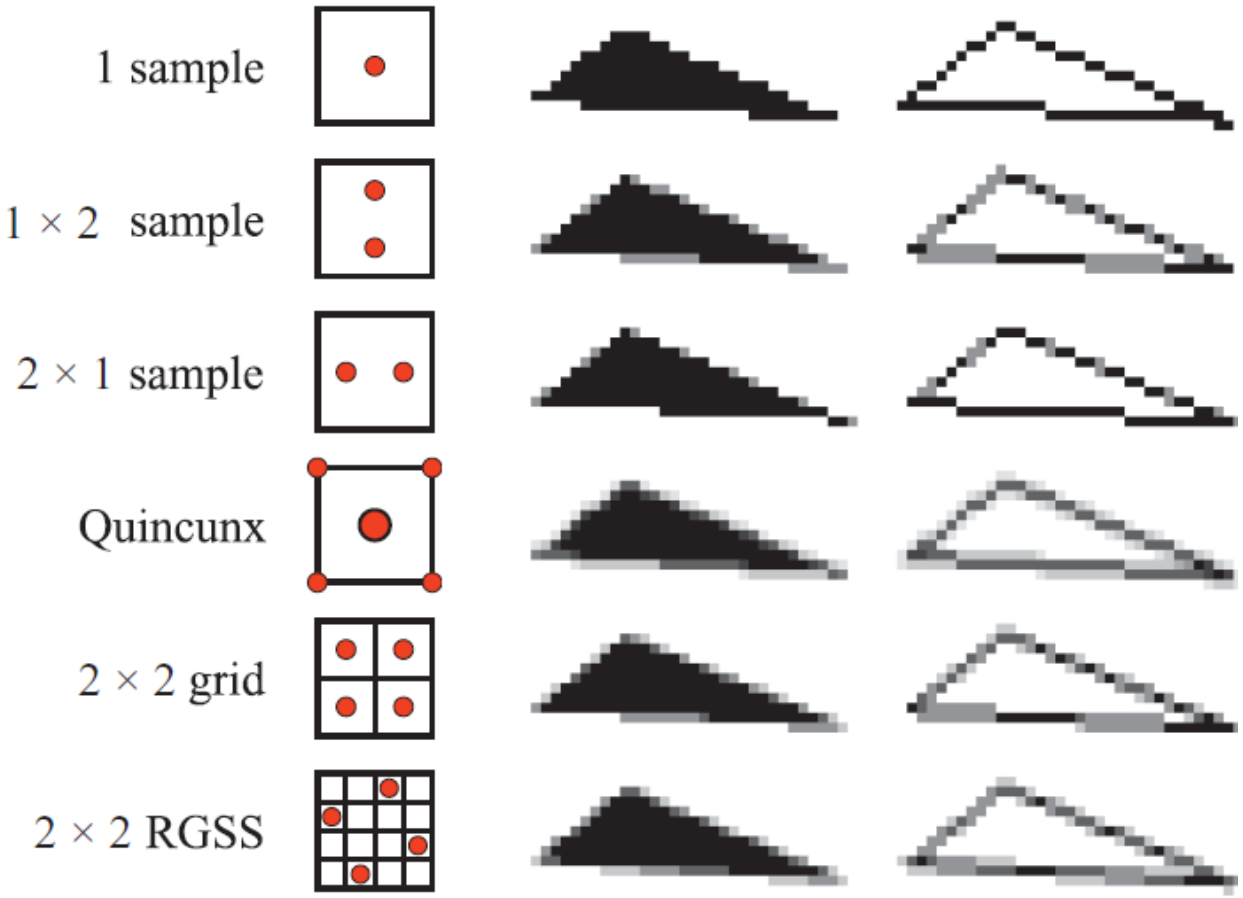


Rotated Grid

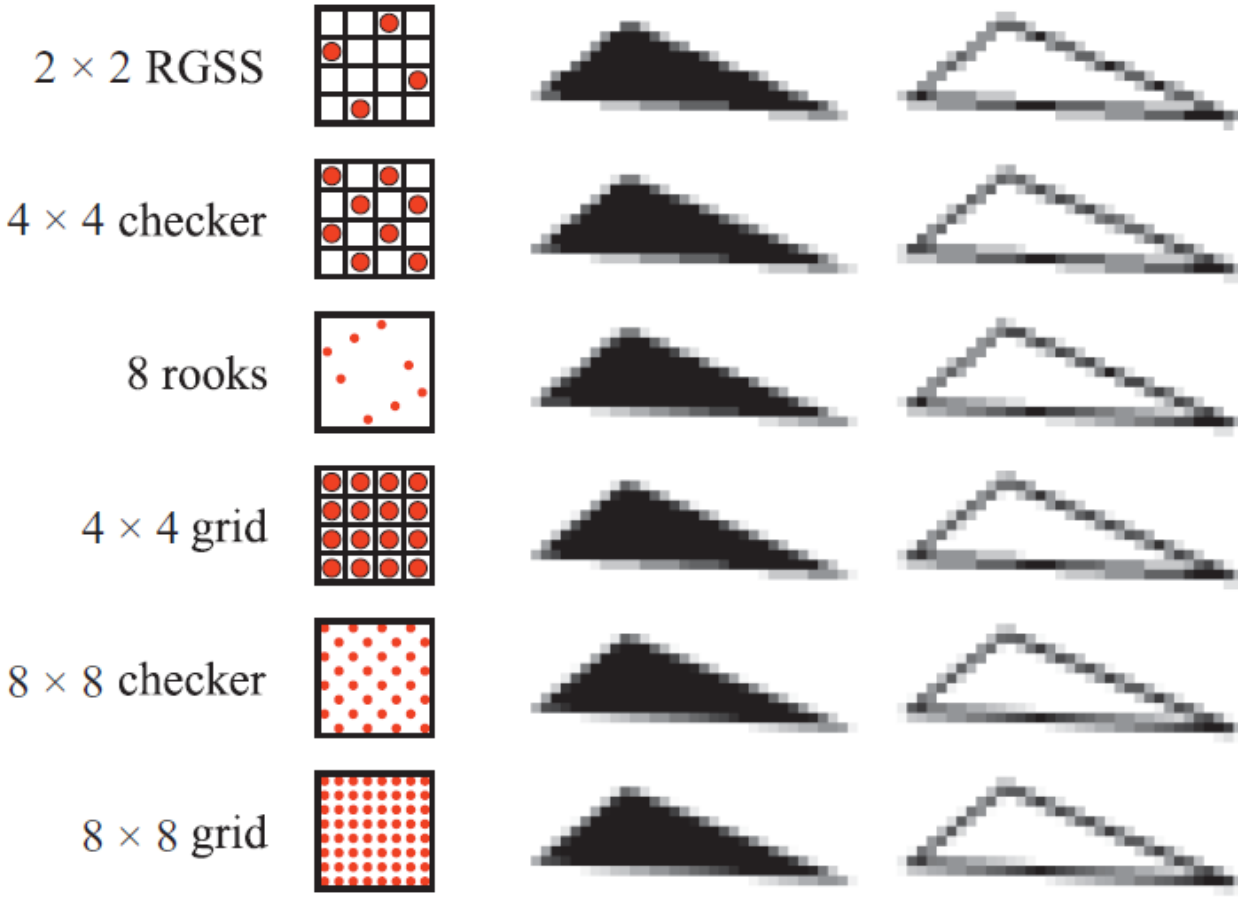
Example from cnblog.com



Common Sampling Patterns



Common Sampling Patterns (2)



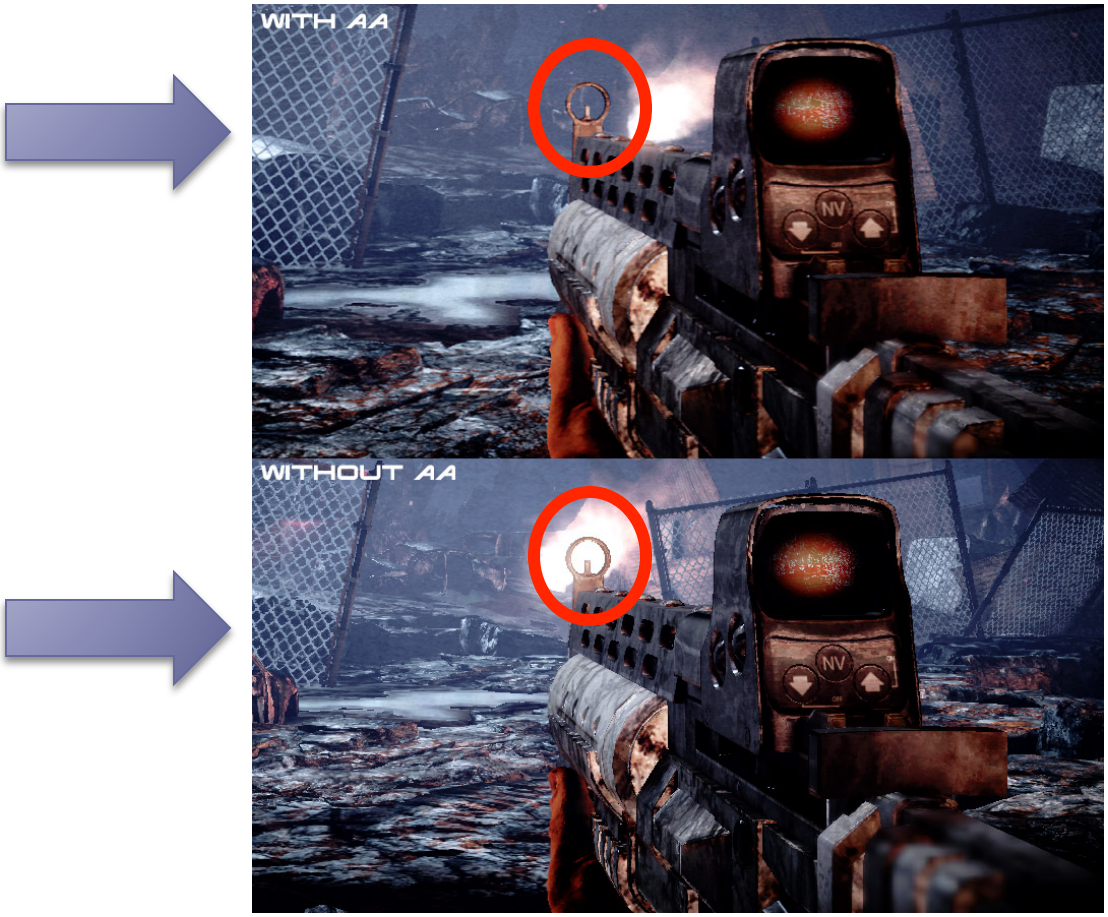
Zoom in to fewer (!) samples

4 × 4 checker

8 rooks

4 × 4 grid

FPS Terminator (indiedb.com)



Adaptive Supersampling

- Instead of using the same number of sample for each pixel, vary it based on some criteria. Basically, want to increase the sample count when needed to improve the image quality.
- It is a recursive subdivision algorithm.
- Need to set termination conditions for the algorithm:
  - The maximum allowed difference between the sample average and sample's color.
  - The subdivision depth.

Adaptive Supersampling

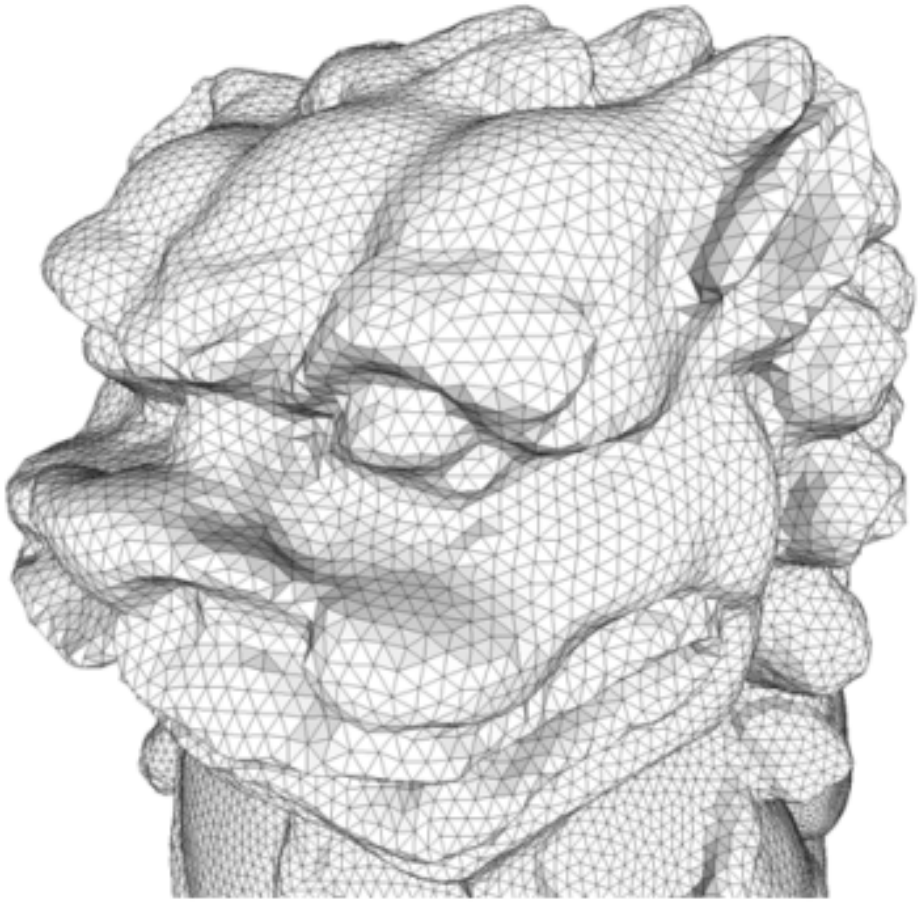
- Pseudo code:
  - For each pixel
    - Take the four samples at its corners
    - Calculate the sample average.
    - If the difference between the average and one of the corner is more than the set threshold AND have not reach the maximum depth
      - Subdivide the pixel into 4 pixels.
    - Else
      - Use the average as the pixel color
  - For each pixel in the final color buffer, calculate the weighted average of its subpixels.

Non-AA (top), Adaptive AA (bottom)

Zoom

Adaptive sampling in 3D! (graphics.eth.cz)

Cool application – Remeshing (graphics.eth.cz)



Multisampling

- Multisampling is a specific case of supersampling.
- Instead of increasing the sampling resolution for everything, increase it for some but not others.
- Typically, the color and texture sampling is done per pixel.
  - While the depth buffer is done per sub-pixel.

Edge Blurring

- Used in deferred shading.
- Basically, using the normal and/or depth buffer, detect the polygon edges and then blur the pixels along those edges based on some criteria.
- Criteria that can be used:
  - Difference in depth
  - Difference in normal orientation
- Done after the lighting stage.

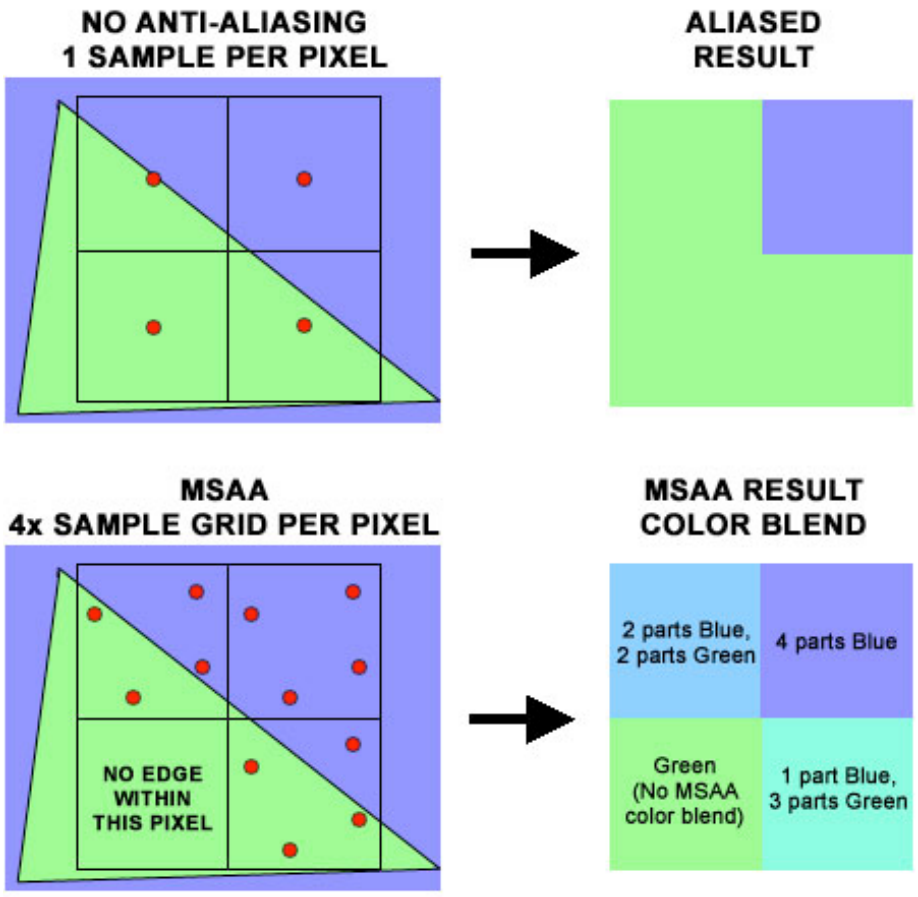
Edge Blurring

- How to detect the edges using normal and/or depth values?

Edge Blurring

- When there is an edge, there is most likely a discontinuity in normal and/or depth. So, what we really trying to detect is a large change in normal and/or depth.
- To detect a large change in value, we can add up all the differences in value between the current pixel's and all of its neighbors.
- The totaled difference can be used to adjust how much we should blur the color of that particular pixel.

MSAA example



Area Sampling

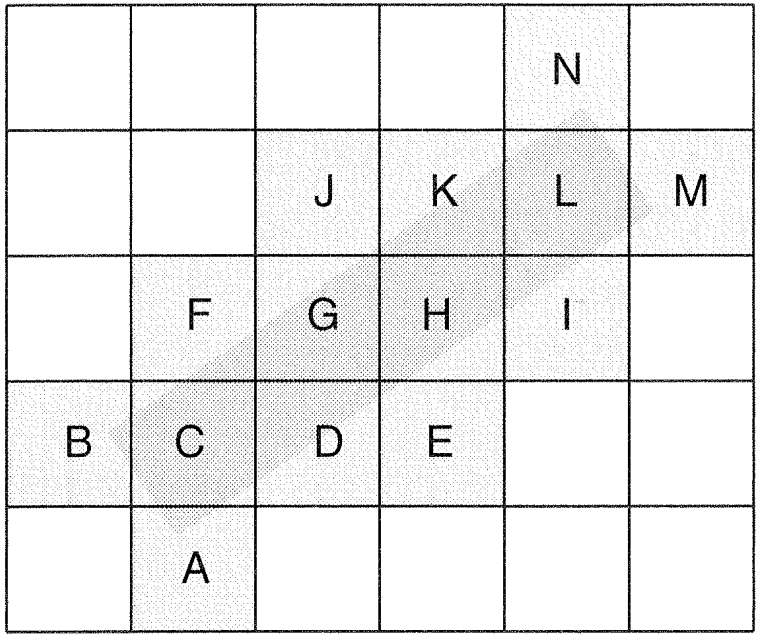
- The basic idea is that the value of the pixel should be proportional to the area of the pixel intersected by the primitive.
- Problems arise if there are multiple primitives intersecting a pixel.
- If the primitives occupy different area of the pixel, then the pixel value is the weighted sum of the primitive values.
- If the primitive overlap, then the visible portion of each primitive within the pixel must be determined before the area can be calculated.

Area Sampling

- So, to properly calculate the weight of the visible primitive for each pixel, area sampling anti-aliasing must be done at the same time as hidden surface removal.

- It is cumbersome, but can be applied for a more specific case. Lines and points.

Area Sampling



A .040510
B .040510
C .878469
D .434259
E .007639
F .141435
G .759952
H .759952
I .141435
J .007639
K .434259
L .878469
M .040510
N .040510

Pixel's coverage values from a line segment

Area Sampling

- In OpenGL, area sampling is used to render anti-aliased lines and points.
- The coverage value is stored as alpha for the pixel. It is up to the user to decide what to do with the alpha.
- Since it uses the alpha value, it means the line is treated as a translucent object. So, blending must be turned on.

Texture Mipmaps

- Other than jagged edges, under sampling of texture map is another major aliasing artifact in computer graphics. It results in texture flickering.
- Fortunately, the standard solution can practically eliminate it.
- The mipmaping process is in essence filtering the high frequency information before the texture is sampled.

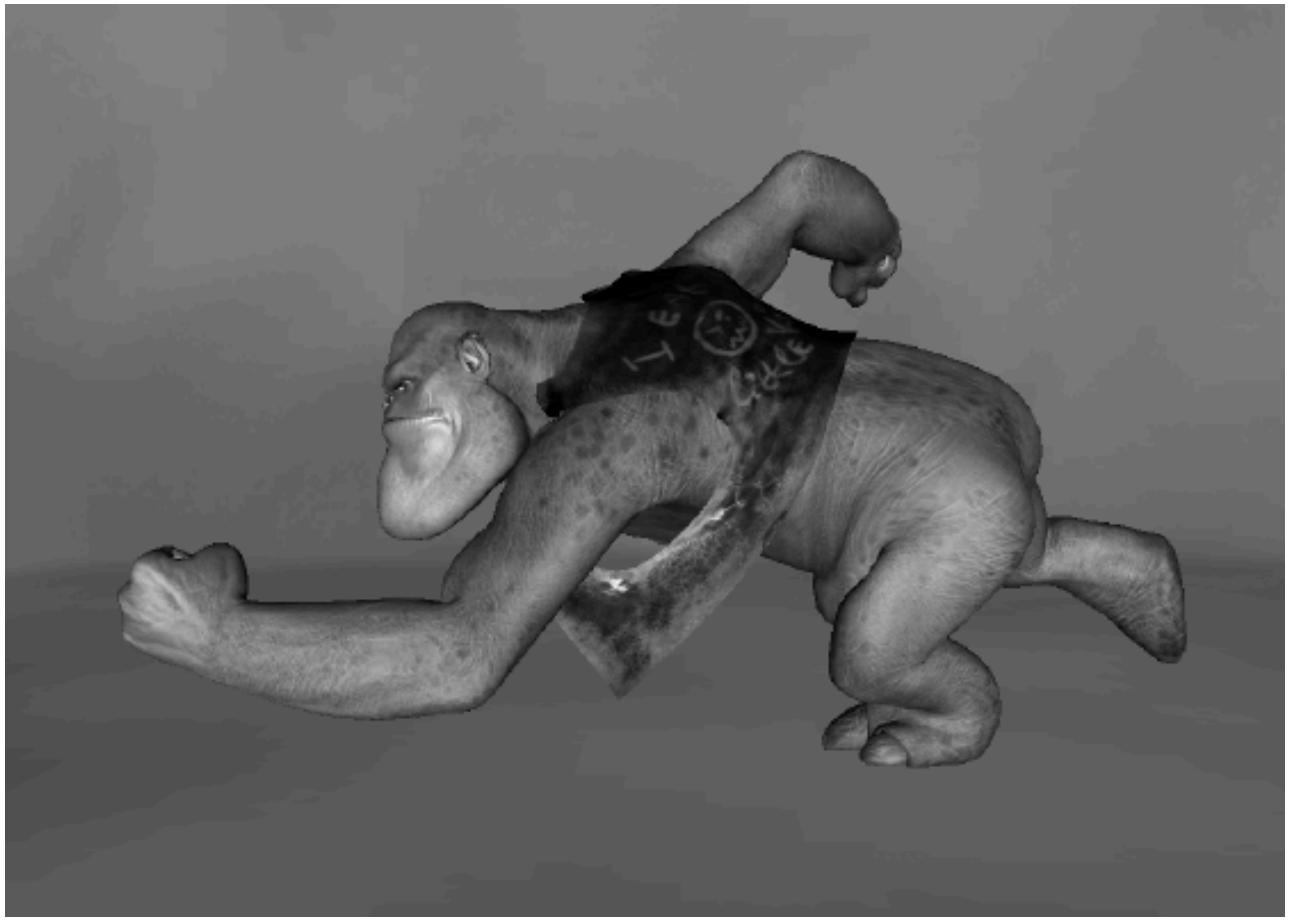
Motion Blur

- In addition to spatial aliasing artifact, we could also get the temporal aliasing artifact if there are animated objects in the scene.
- Do motion blur to fix the temporal aliasing artifact.
- In order to do motion blur, we need to re-render the scene multiple times. And use some post processing to combine the results together to one frame.
- It is very expensive. As we need to calculate object position, orientation, etc for each of the sub-frames and rendering them.

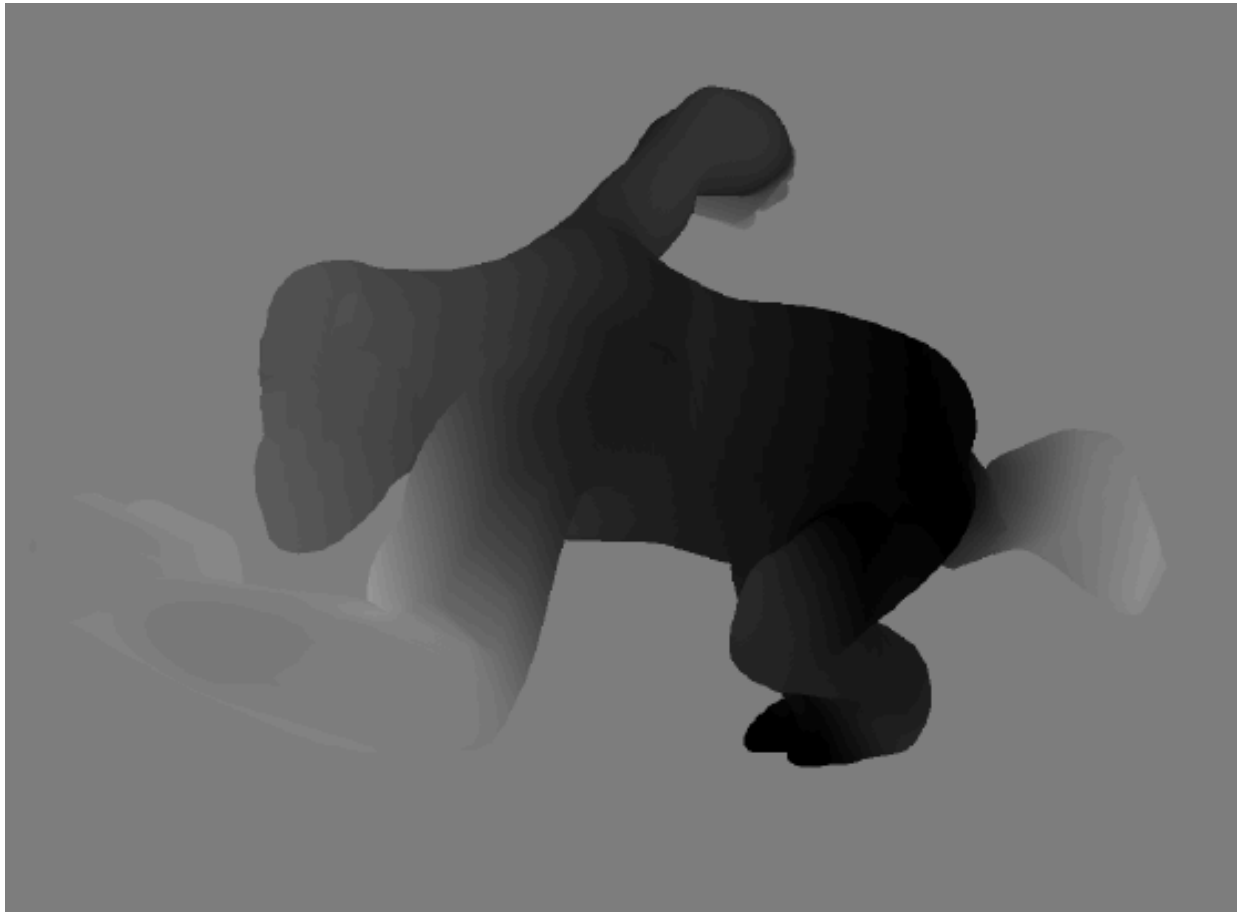
Motion Blur

- A quick way to do motion blur:
  1) At the end of the render function, copy the color buffer content to texture T.
  2) Render the next frame.
  3) Render texture T over the whole screen as a translucent quad.
  4) Go back to step 1.

Motion Blur



Velocity Visualization



Final Rendering

References

- Moller and Hanes, "Realtime Rendering," 3rd edition
- Shreiner, D., Woo, M., Neider, J. and Davis, T., "OpenGL Programming Guide", Fifth Edition
- McReynolds, T. and Blythe, D., "Advance Graphics Programming Using OpenGL"
- www.wikipedia.com
- Simon Green , "Stupid OpenGL Shader Tricks," NVIDIA GDC 2003 presentation