Sampling

Dinesh K. Pai

Textbook Chapter 16

Several slides courtesy of M. Kim

1

Announcements

- Final exam resources
  - It's not too early to start studying!
  - Extra office hours just before exam on 22nd, 23rd, 24th and 25th. 1-2pm in room 005
  - Review during last two classes
- Industry talk on VFX pipeline today at 3pm
- A personal note

2

Visual Effects Industry talk

- By Andrew Kaufman, my former M.Sc. student, now at Image Engine in Vancouver
- "The VFX Pipeline", Fri March 28, 3pm, DMP 301
- Guest lecture in CPSC 426 Computer Animation class.

3

C^3 Survey

- How far along are you with Assignment 4
  a) Not started
  b) Can run template code
  c) Finished at least one part
  d) Finished all specified parts (1,2,3)
  e) Finished everything

5

C³ Review

- The z-buffer stores
  a) A zipped version of the rendered image
  b) Distance of a fragment in the eye frame
  c) Nearness (1/depth) of a fragment in the eye frame
  d) Shadows cast by the light source

6

Two views of images

- A *continuous image*, $I(x_w, y_w)$, is a bivariate function.
  - range is a linear color space.
- A *discrete image* I[i][j] is a two dimensional array of color values.
- We associate each pair of integers i, j, with the continuous image coordinates $x_w = i$ and $y_w = j$

7

Sampling

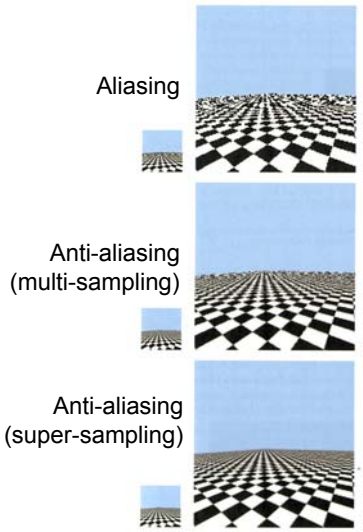
- The simplest and most obvious method to go from a continuous to a discrete image is by *point sampling.*
- To obtain the value of a pixel i, j, we sample the continuous image function at a single integer valued domain location:

$$\mathrm{I[i][j]} \leftarrow I(i, j)$$

- This can results in unwanted artifacts.

8

Aliasing and anti-aliasing



These are polygons, not Textures!

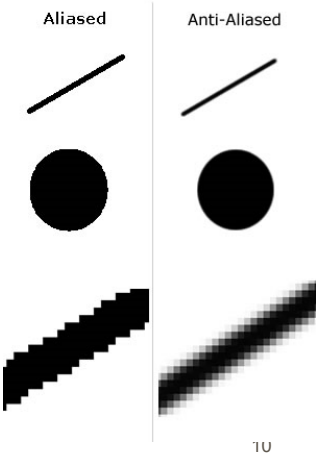
← still have some problems

↖ good here

9

Aliasing

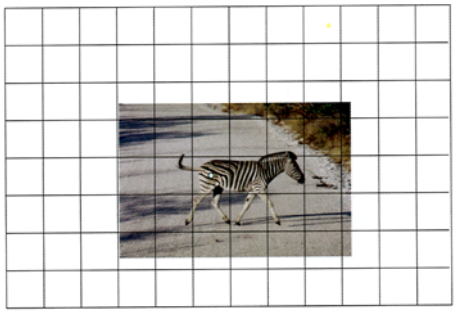
- Scene made up of black and white triangles: jaggies at boundaries
  - Jaggies will crawl during motion
- If triangles are small enough then we get random values or weird patterns.
  - Jaggies will crawl during motion



10

Aliasing

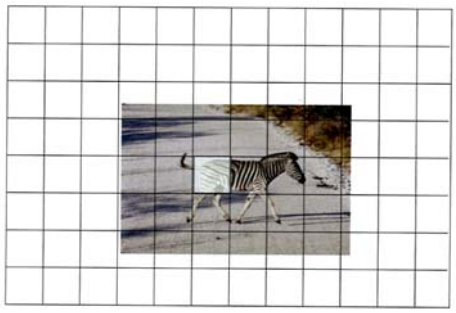
- The heart of the problem: too much information in one pixel



11

Anti-aliasing

- Intuitively: the single sample is a bad value, we would be better off setting the pixel value using some kind of average value over some appropriate region.
- In the above examples, perhaps some gray value.



12

Anti-aliasing

- Mathematically this can be modeled using *Fourier analysis*.
  - Breaks up the data by “frequencies” and figures out what to do with the un-representable high frequencies.

13

Anti-aliasing

- We can also model this as an optimization problem.
- These approaches lead to:

$$\mathrm{I[i][j]} \leftarrow \iint_{\Omega} I(x,y)F_{i,j}(x,y)dxdy$$

- where $F_{i,j}(x,y)$ is some function that tells us how strongly the continuous image value at $[x,y]^t$ should influence the pixel value i, j

14

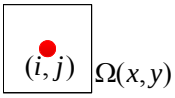
Anti-aliasing

- In this setting, the function $F_{i,j}(x,y)$ is called *a filter*.
  - In other words, the best pixel value is determined by performing some continuous weighted averaging near the pixel's location.
  - Effectively, this is like blurring the continuous image before point sampling it.

15

Box filter

- We often choose the filters $F_{i,j}(x,y)$ to be something non-optimal, but that can more easily computed with.
- The simplest such choice is a *box filter*, where $F_{i,j}(x,y)$ is zero everywhere except over the 1-by-1 square center at $x = i, y = j$.



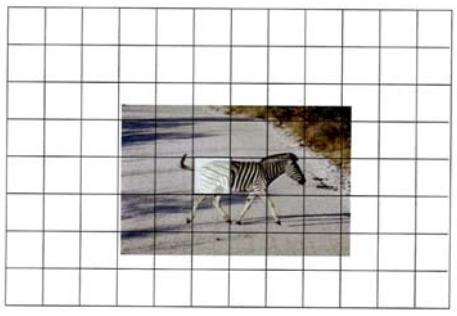
- Calling this square $\Omega_{i,j}$, we arrive at

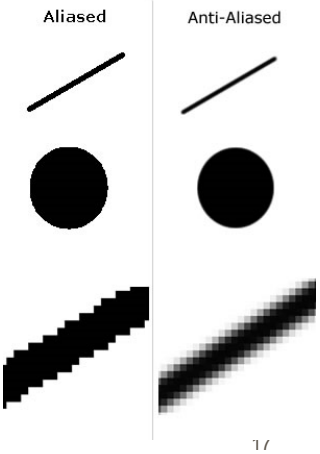
$$\mathrm{I[i][j]} \leftarrow \iint_{\Omega_{i,j}} I(x,y)dxdy$$

16

Box filter

- In this case, the desired pixel value is simply the average of the continuous image over the pixel's square domain.





17

Over-sampling

- Even that integral is not really easy to compute
- Instead, it is approximated by some sum of the form:

$$\mathrm{I[i][j]} \leftarrow \frac{1}{n}\sum_{k=1}^{n} I(x_k, y_k)$$

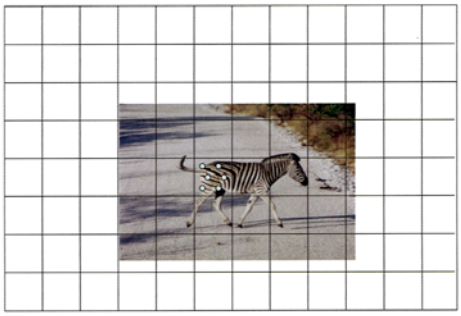
where *k* indexes some set of locations (x_k, y_k) called the sample locations.

- The renderer first produces a "high resolution" color and z-buffer "image",
  - where we will use the term *sample* to refer to each of these high resolution pixels.

18

Over-sampling

- Then, once rasterization is complete, groups of these samples are averaged together, to create the final lower resolution image.



19

Super-sampling

- If the sample locations for the high resolution image form a regular, high resolution grid, then this is called *super sampling.*
- We can also choose other sampling patterns for the high resolution "image",
  - Such less regular patterns can help us avoid systematic errors that can arise when using the sum to replace the integral.

20

Multi-sampling

- Render to a "high resolution" color and z-buffer
- *During the rasterization* of each triangle, "coverage" and z-values are computed at this sample level.
- But for efficiency, the fragment shader is only called **only once per final resolution pixel.**
  - This color data is shared between all of the samples hit by the triangle in a single (final resolution) pixel.
- Once rasterization is complete, groups of these high resolution samples are averaged together.

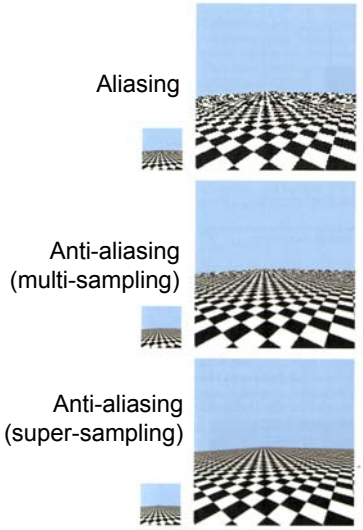
21

Multi-sampling

- Multisampling can be an effective anti-aliasing method since, without texture mapping, colors tend to vary quite slowly over each triangle, and thus they do not need to be computed at high spatial resolution.
- To deal with aliasing that occurs during texture mapping, we have the advantage of possessing the texture image in hand at the outset of the rendering process.
- This leads to specialized techniques such as *mip mapping*.

22

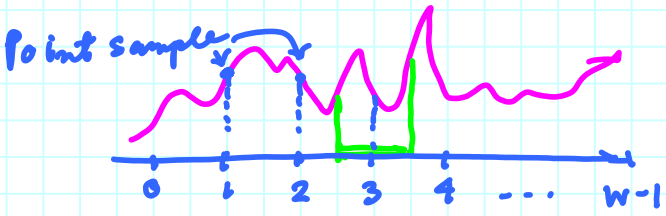
Aliasing and anti-aliasing



23

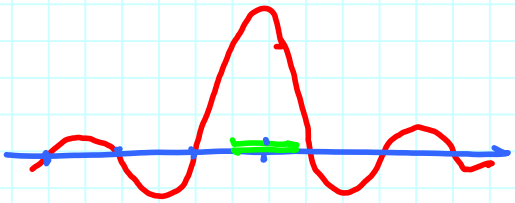
Images in 1D (for illustration)

Note Title 2014-03-28



$I(x)$ continuous image

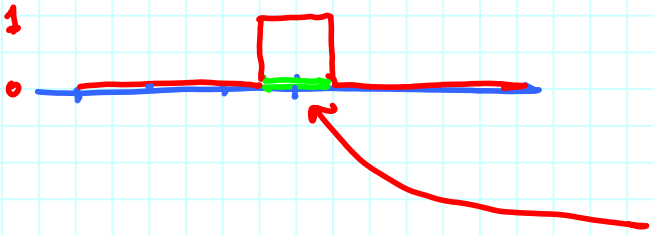
$I[i]$ discrete image



★ Pixels are not points!

Have finite extent

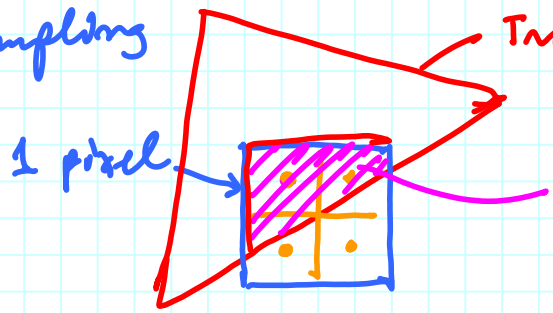
$F(x)$ in general (e.g. $\mathrm{sinc}(x)$ has good properties, but expensive to compute with)



$F(x)$ = Box function

Think of this as an "if" that only looks at function in this interval

§ Oversampling



Supersampling with n samples

$n \times$ memory

$n \times$ computation

$\Rightarrow$ Can reduce with multipass rendering + use of "accumulation buffer"