CPSC 314
Computer Graphics

Dinesh K. Pai

Texture Filtering (Ch. 16, 17, 18)

Some slides courtesy of M. Kim, KAIST

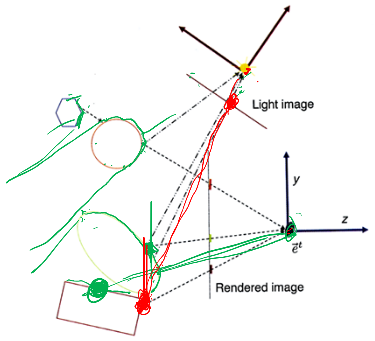
Preliminaries

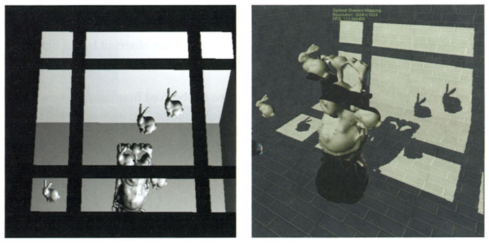
- Announcements and Reminders
- Today
  - Shadows and depth, wrap up
  - Texture filtering

2

Shadow mapping

- If a point observed by the eye is not observed by the light, then there must be some occluding object in between, and we should draw that point as if it were in shadow.





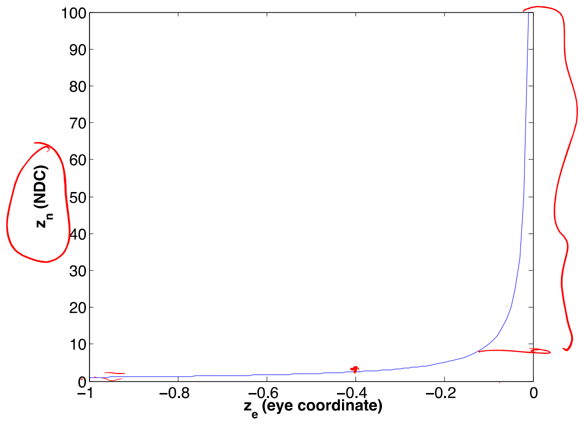
3

Numerics

- points very far from the eye have z_n values very close to zero

$$z_n = \frac{-1}{z_e}$$

```
ze=-1*[0.01:0.01:10];
zn=-1./ze;
plot(ze(1:100),zn(1:100))
```



4

Texture Sampling and Filtering

- This a big and subtle topic.
  - The textbook covers it well over 3 chapters!
  - We'll take a simplified and pragmatic approach. But read the book for theory
- Question: Why do we access texture values using things called "Sampler2D" "SamplerCube" etc.?

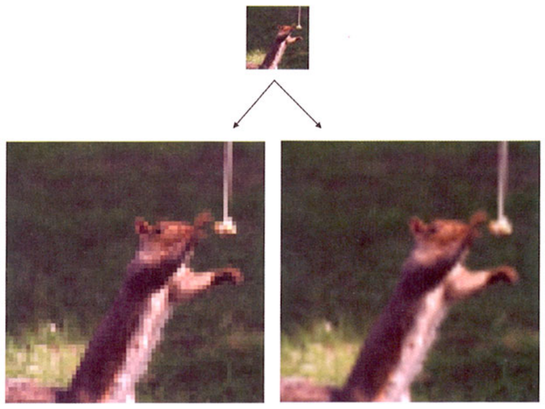
Texture Magnification

Chapter 17

RECONSTRUCTION

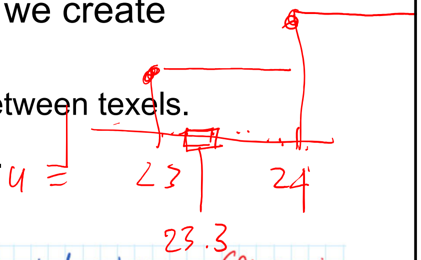
(DISCRETE → CONTINUOUS)

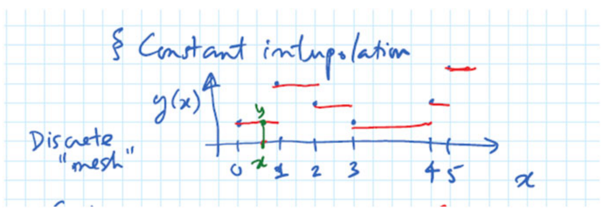
Example

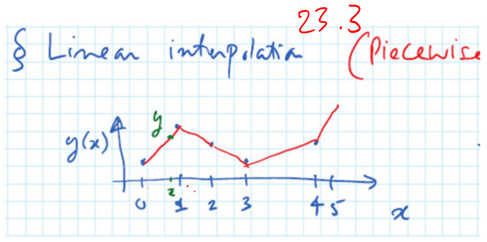


Reconstruction

- Given a discrete image I[i][j], how do we create a continuous image *I(x,y)*?
  - How to get a texture colors that fall in between texels.
- This process is called *reconstruction*.
- Key idea: Interpolation

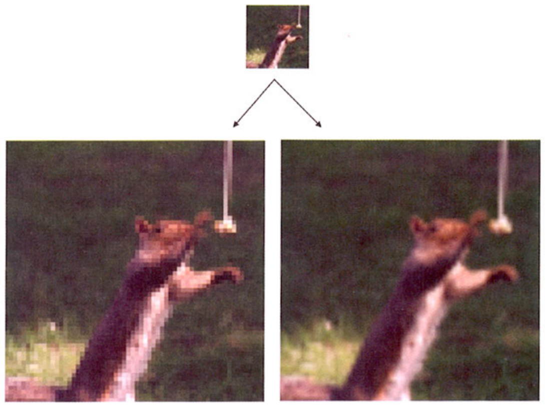






Constant reconstruction

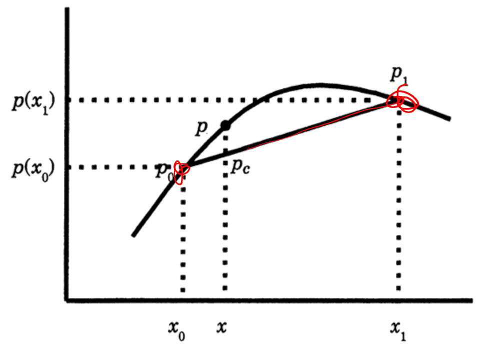
- The resulting continuous image is made up of little squares of constant color.
- Each pixel has an influence region of 1-by-1

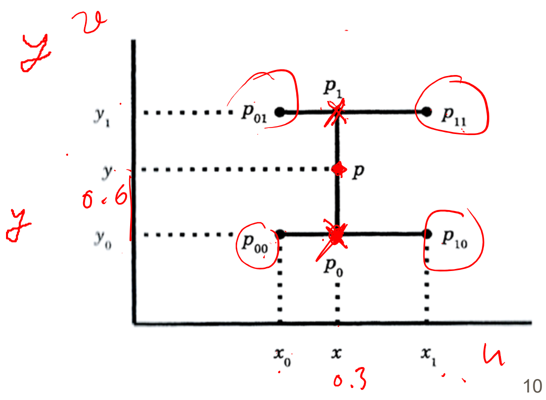


Linear and Bilinear interpolation

We already know how to interpolate in 1D

- Linear (1D) Bilinear (2D):





Bilinear reconstruction

- Can create a smoother looking reconstruction using *bilinear interpolation*.
- Bilinear interpolation is obtained by applying linear interpolation in both the horizontal and vertical directions. Pseudocode (not needed for WebGL)

```
color bilinearReconstruction(float x, float y, color
image[][]){
  int intx = (int) x;
  int inty = (int) y;
  float fracx = x - intx;
  float fracy = y - inty;

  color colorx1 = (1-fracx) * image[intx][inty] +
            (fracx) * image[intx+1][inty];
  color colorx2 = (1-fracx) * image[intx][inty+1] +
            (fracx) * image[intx+1][inty+1];
  color colorxy = (1-fracy) * colorx1 +
            (fracy) * colorx2;
  return(colorxy);
```

11

Bilinear properties

- At integer coordinates, we have *I(x,y)*=I[i][j]; the reconstructed continuous image *I* agrees with the discrete image I. => Interpolation
- In between integer coordinates, the color values are blended continuously.
- Each texel influences, to a varying degree, each point within a 2-by-2 square region of the continuous image. => Local Support
- The horizontal/vertical ordering is irrelevant.
- Color over a square is bilinear function of (x,y).

12

Texture Minification

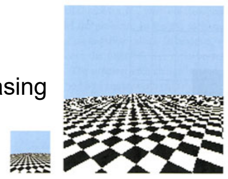
Chapter 18

RESAMPLING

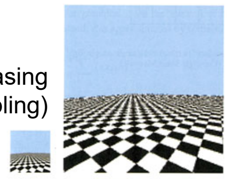
(RECONSTRUCTION+SAMPLING, DISCRETE→CONTINUOUS→DISCRETE)

Problem: Aliasing

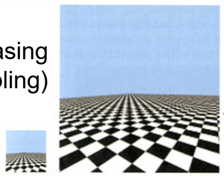
Aliasing



Anti-aliasing (multi-sampling)

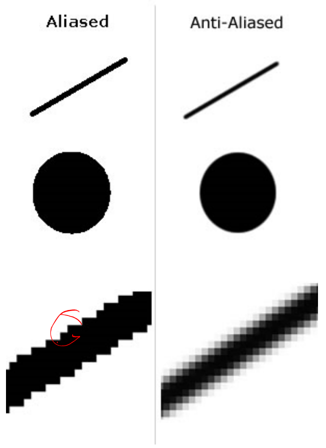


Anti-aliasing (super-sampling)



Aliasing

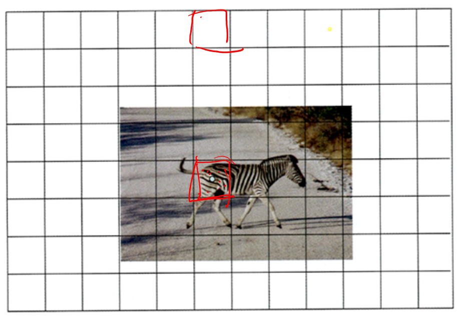
- Scene made up of black and white triangles: jaggies at boundaries
  - Jaggies will crawl during motion
- If triangles are small enough then we get random values or weird patterns.



15

Aliasing

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

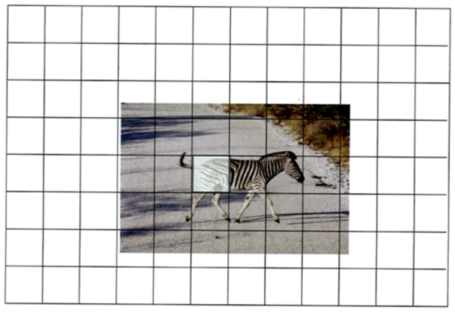


Soln: average or blur.

16

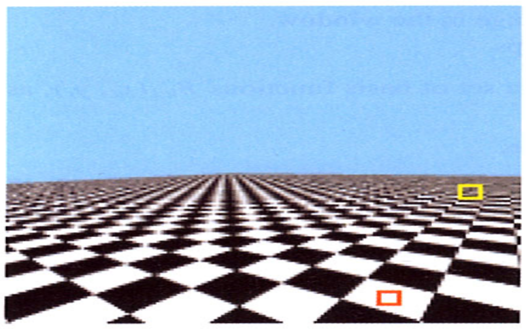
Anti-aliasing

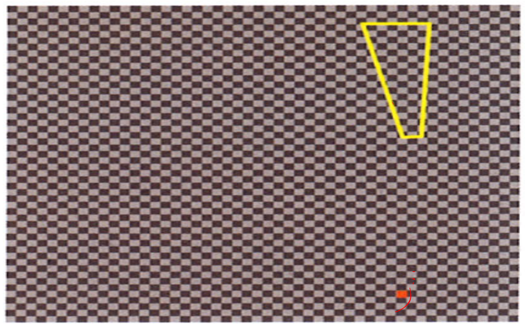
- Intuitively: the single sample is a bad value, we would be better off setting the pixel value using some kind of average value over some appropriate region.
- In the above examples, perhaps some gray value.



17

Resampling

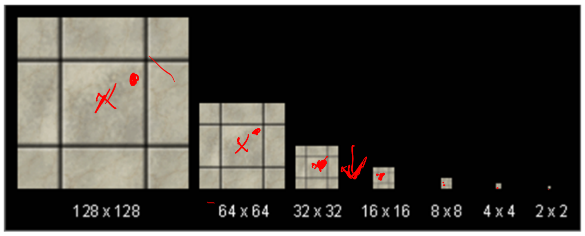




18

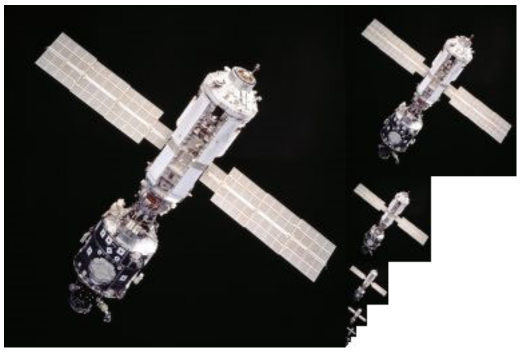
Mip mapping

- In mip mapping, one starts with an original texture T^0 and then creates a series of lower and lower resolution (blurrier) texture T^i.
- Each successive texture is twice as blurry. And because they have successively less detail, they can be represented with ½ the number of pixels in both the horizontal and vertical directions.



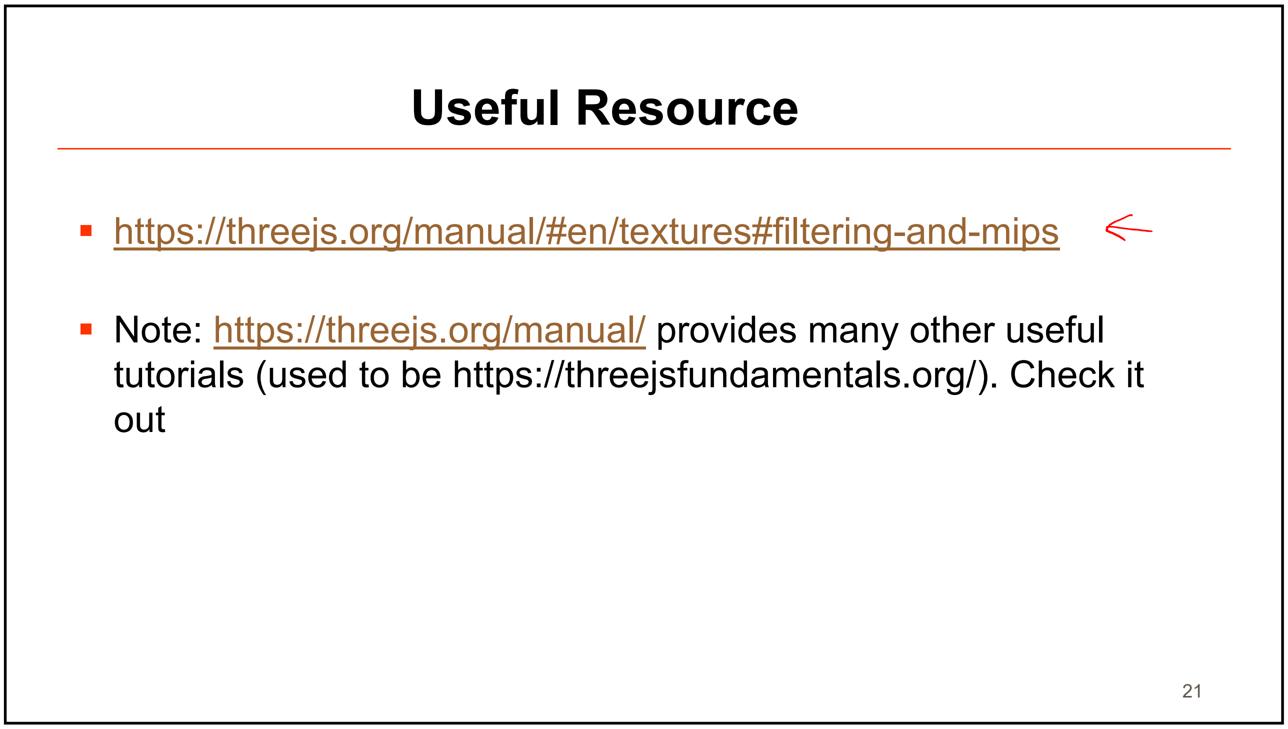
19

Mipmap example



Source: wikipedia

20



Useful Resource

- https://threejs.org/manual/#en/textures#filtering-and-mips ←
- Note: https://threejs.org/manual/ provides many other useful tutorials (used to be https://threejsfundamentals.org/). Check it out