Reconstruction

Dinesh K. Pai

Textbook Chapter 17

Several slides courtesy of M. Kim

Today

- Announcements
  - Next week will be devoted to course review (and brief discussion of optional topics)
  - Office hour today rescheduled due a conflict. New for this week: Thursday 11-12.
- Assignment 4 extension till Thursday midnight.
  - Note: if you still have grace days, the maximum allowed extension is till Sunday midnight.
- Reconstruction and Resampling
Shadow mapping recap for A4

Step 1: Compute depth image
Note: depth file.s is storing depth in a "packed" format (for WebGL 1)

Step 2: While rendering the final image
modify terrain shadow

→ Need to transform a terrain position to light and frame

→ Perspective divide (by w) yourself.

→ Find depth value using texture lookup of depth image

→ Render NDC sizes from -1 to 1
   Commit to 0 to 1
Alpha blending

- Brief recap
  - Three.js examples
    https://threejs.org/examples/#webgl_materials_transparency
  - Clarification of a subtle point: “Premultiplied” and “non-Premultiplied” alpha
  - Note: .png files store non-premultiplied

Recap: Alpha blending

- Associate with each pixel in each image layer, a value, $\alpha[i][j]$, that describes the overall opacity or coverage of the image layer at that pixel.
  - An alpha value of 1 represents a fully opaque/occupied pixel, while a value of 0 represents a fully transparent/empty one.
  - A fractional value represents a partially transparent (partially occupied) pixel.
- Alpha will be used during compositing.
Alpha definition

- More specifically, let \( I(x, y) \) be a continuous image, and let \( C(x, y) \) be a binary valued \((x, y)\) coverage function over the continuous domain, with a value of 1 at any point where the image is “occupied” and 0 where it is not.
- Let us store in our discrete image the values:

\[
I[i][j] \leftarrow \int_{\Omega_{i,j}} I(x, y)C(x, y)\,dx\,dy
\]

\[
\alpha[i][j] \leftarrow \int_{\Omega_{i,j}} C(x, y)\,dx\,dy
\]

Over operation

- To compose \( I^f[i][j] \) over \( I^b[i][j] \), we compute the composite image colors, \( I^c[i][j] \), using

\[
I^c[i][j] \leftarrow I^f[i][j] + I^b[i][j] (1 - \alpha^f[i][j])
\]

That is, the amount of observed background color at a pixel is proportional to the transparency of the foreground layer at that pixel.

- Likewise, alpha for the composite image can be computed as:

\[
\alpha^c[i][j] \leftarrow \alpha^f[i][j] + \alpha^b[i][j] (1 - \alpha^f[i][j])
\]

\(\theta\) More efficient \(\ominus\) Loss of precision at transparent points
Reconstruction

- Given a discrete image $I[i][j]$, how do we create a continuous image $I(x,y)$?
- Is central to resize images and to texture mapping.
  - How to get a texture colors that fall in between texels.
- This process is called reconstruction.
- We already know the key idea, from L23-L24: Interpolation! So we will go over this quickly.
**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)

```c
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);
}
```

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 pixel 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 \( (x,y) \).
Chapter 18

RESAMPLING
(RECONSTRUCTION + SAMPLING, DISCRETE $\rightarrow$ CONTINUOUS $\rightarrow$ DISCRETE)

Resampling
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 $\frac{1}{2}$ the number of pixels in both the horizontal and vertical directions.