Resampling
Dinesh K. Pai
Textbook Chapter 18
Several slides courtesy of M. Kim

Announcements

- I will be out of town on Friday. We will have a regular lecture, taught by TAs
  - Topic: a high level intro to Computer Animation and Geometric Modeling (roughly corresponding to Chapters 22 and 23 of text)
- The last class will be Monday April 7. Will review course and exam preparations.
- Assignment 3 grades now uploaded. Rest by the weekend
- TA evaluations at the end of the class (on paper)
Recent example of using depth and compositing

- Depth cameras are now becoming available in mobile phones. Can use with compositing

Chapter 18

RESAMPLING
(RECONSTRUCTION+SAMPLING, DISCRETE→CONTINUOUS→DISCRETE)
Resampling

- Lets revisit texture mapping
- We start with a discrete image and end with a discrete image.
- The mapping technically involves both a reconstruction and sampling stage.
- In this context, we will explain the technique of mip mapping used for anti-aliased texture mapping.
Resampling equation

- Suppose we start with a texture image (discrete) $T[k][l]$ and apply some 2D warp to this image to obtain an output image $I[i][j]$.
- Reconstruct a continuous texture $T(x_t, y_t)$ using a set of basis functions $B_{k,l}(x_t, y_t)$.
- Apply the geometric wrap (at the view point) to the continuous image.
- Integrate against a set of filters $F_{k,l}(x_w, y_w)$ (a box filter) to obtain the discrete output image.

Resampling equation

- Let the geometric transform be described by a mapping $M(x_w, y_w)$ which maps from continuous window to texture coordinates.
- We obtain:
  $$I[i][j] \leftarrow \int \int_{\Omega} F_{i,j}(x_w, y_w) \left( \sum_{k,l} B_{k,l}[M(x_w, y_w)]T[k][l] \right) dx_w dy_w$$
  $$= \sum_{k,l} T[k][l] \left( \int \int_{\Omega} F_{i,j}(x_w, y_w) \left( B_{k,l}[M(x_w, y_w)] \right) dx_w dy_w \right)$$
  (we could obtain an output pixel as a linear combination of the input texture pixels.)
Resampling Intuition

\[ T_k \]

\[ k = 1, 2, 3, 4, 5, 6, 7 \]

\[ x_k \rightarrow \]

Reconstruction \( \equiv \) Interpolation

\[ T(x_k) = \sum B_k(x_c) T_k \]

\[ x_c \rightarrow \]

Viewpoint + perspective, etc.

\[ x_w = N(x_c) \text{ or } x_c = M(x_w) \]

could be complicated

\[ T(x_c) \]

Magnification

Interpolation

Fine sampling

Minification
Magnification

Interpolation

Constant Linear

Nearest neighbor

Minification

Interpolation

Filt & Sample

Mipmaps Pre Filter

the texture

§ Mip mappings
Magnification

- We tell OpenGL to do this using the call `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)`.
- For a single texture lookup in a fragment shader, the hardware needs to fetch 4 texture pixels and blend them appropriately.

Minification

- In the case that a texture is getting shrunk down, then, to avoid aliasing, the filter component should not be ignored.
- Unfortunately, there may be numerous texture pixels under the footprint of $M(\Omega_{i,j})$, and we may not be able to do our texture lookup in constant time.
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.

Mipmap example

Source: wikipedia
Mip mapping

- Mip mapping with trilinear interpolation is specified with the call
  `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR)`

- Trilinear interpolation requires OpenGL to fetch 8 texture pixels and blend them appropriately for every requested texture access.

Trilinear interpolation
Properties

- It is easy to see that mip mapping works reasonably well, but has limitations that can be addressed by more advanced methods.