CS 4620 Homework 6

out: Wednesday 29 October 2008
due: 2pm Friday 7 November 2008

Problem 1: Image resampling

Write code to resample an image using a separable cubic filter to achieve nearly artifact-free scaling.

This homework comes with a framework that displays two windows and lets you select a source and destination rectangles for resampling. It calls a method `resample` in the `AxisAlignedResampler` class to perform the resampling operation, given the two corners on the input image, and two corners on the output image. The input image and output image can be accessed via the fields `inputImage` and `outputImage` in the `Resampler` class. The corners on the two images define a mapping between these two images, and your code should write a scaled copy of the content in the source rectangle to the target rectangle.

To keep things simple, you should use wraparound boundary conditions, which are already implemented by the `Image` class, to avoid having to treat the edges separately in any way—you can just go ahead and access pixels past the edge of the image. A field `clipToDestination` specifies whether the content should only fill the destination rectangle or should just be allowed to tile itself across the whole output image.

The framework comes with code that does this job by point sampling, which produces severe aliasing artifacts. Your code should instead use the Mitchell-Netravali bicubic filter presented in class and in the book, taking advantage of separability to improve performance. The filter is already typed in for you in `Resampler.filter`.

Your code will run slower than point sampling, but it should still require only a second or so to resample a 1000-pixel-wide image.

Problem 2: *Projective warping

Write code to similarly warp a rectangular image to an arbitrary quadrilateral using projective warping, using a mipmap for antialiasing.

If you choose the "non-axis-aligned" mode in the framework, you can drag the four corners of the source and destination areas independently. This creates a situation identical to view-

ing a texture-mapped rectangle in perspective. Just as in the previous part, turning off "clip to destination" treats the input image like a tiled texture, creating a situation analogous to viewing a texture-mapped infinite plane in perspective.

1. Implement `PerspectiveResampler.resample` by point sampling.
2. Upgrade your implementation to linear interpolation.
3. Modify the image class so that it keeps a mipmap pyramid.
4. Compute the derivative of the projective warp, to measure the size of the region that should be averaged. Use this derivative to choose a mipmap level.

The solution to Homework 4 Problem 4 has been coded up in the method `computeTransformation` in `PerspectiveResampler`; you can use it to set up the warping transformation. You should compute the source point incrementally, though, to avoid the expense of a matrix multiply per pixel.