CO 417 Spring 2017 Homework 1

due Thursday Feb 9, 2017, 9am
(please submit results on CATE before class)

- **Part 1: Assemble an HDR Image**
  - Given seven exposures of the Memorial church scene two stops apart, assemble an HDR image. Images are provided in both a regular 8-bit format (.ppm) and floating point format (.pfm). The 8-bit versions of the images are only provided to enable viewing of the images in a regular image viewer. For HDR assembly, use the .pfm versions instead.

  - Use sample **Python/C code** provided for loading images in floating point .pfm file format and assemble the .pfm image sequence using your own weighting function for combining values from multiple exposures into a radiance value. Assume the response curve is linear for the .pfm images.

    - The algorithm should run as follows:
      - Load all the images in the sequence Zi(x,y).
      - Create a new blank float image buffer F the same size.
      - Compute the amount of relative exposure Δt_i for each image. For the given seven images taken with the same f/stop and ISO, two stops apart in shutter speed, these relative exposures Δt_i would be 1, 4, 16, 64, 256, 1024, 4096.
      - Implement a center-weighting function w(z) defined over [0..1] such that w(0)=0, w'(0)=0 (first derivative is 0 at 0), w(1)=0, w'(1)=0, and w(0.5)=1. Please describe the function you implement in your report.
      - Compute F(x,y) = exp(SUM_over_i (log((1/ Δt_i) * Zi(x,y)) * w(Zi(x,y))) / SUM(w(Zi(x,y)))) to create the assembled HDR image. This computes a weighted average of the pixel values in the sequence, inversely scaled by the amount of exposure to produce radiance values. It achieves better conditioning by taking a weighted average of the log of the radiance values, and then exponentiating the result
      - Ignore values < 0.005 and values > 0.92 in any image when combining values into HDR. Also watch out for any bad pixels in the input LDR images!
      - Save as a new HDR image file **f** in .pfm format.

  - See Debevec & Malik, SIGGRAPH 97, section 2.2, debevec-siggraph97.pdf. Also at: http://www.debevec.org/Research/HDR/

  - Report on the total dynamic range of the scene – the ratio of the brightest pixels to the dimmest reliably measured pixels.

- Tone map the HDR image **f** and save in .ppm format. Note that you will need to convert the floating point range of pixel values to the range 0 – 255 (unsigned char) to save as .ppm.

    - Implement the simplest tone mapper –> a linear scaling from 0.0 – Max (**f**) to 0.0 – 1.0 (**n**) and save **n** as .ppm (0 - 255).
    - Implement an exposure function that scales the values of **n** 0.0 – 1.0 by a factor of 2 (+ 1 stop) at a time to obtain **n'**. Clamp values of **n'** above 1.0 to 1.0 to save as .ppm. Record the exposure setting (number of stops) that you needed to make the image look good, i.e. see information in the bright and dark part at the same time.
    - Implement a gamma function **g** = pow(**n''**, 1/gamma) and save **g** as .ppm (0 - 255). Here **n''** represents some exposure setting (+ stops) applied to **n** to brighten the image. Try gamma = 1.5, 1.8, 2.2, 2.5 & 3.0 and choose the one that gives you the best looking result. Report the choice of exposure setting (stops) and gamma for the best looking result! For example, stops = 0 and gamma = 1.0 means **g** = **n**.

- **Part 2: Implement simple Image Based Lighting**
    - Given a light probe, the Grace cathedral lighting environment in latlong format (.pfm), create an image **m** of 511x511 resolution containing a mirror ball sphere lit by the latlong map. The corresponding .ppm of Grace is provided simply for viewing the latlong map in a regular image viewer.
    - Assume the reflectivity of the sphere to be 1.0 for the rendering and assume an orthographic view vector v = [0, 0, 1], i.e, + Z axis.
    - Note that you do not need to do any actual ray-tracing, just pixel indexing into the latlong map to create the rendering!
    - The algorithm should proceed as follows:
        - First trace out a circle of diameter 511 pixels inside the square image. The pixel coordinates of every pixel enclosed inside this circle denotes a specific surface normal n. Note that n needs to be normalized. Set pixels outside the circle in the image to black.
        - Use the equation of reflection to calculate the reflection vector r on each pixel of the sphere as $r = 2*(n \text{ dot } v)*n - v$. Note that ($n$ dot v) is a scalar but that r, n and v are vectors!
        - Write out .pfm images encoding the normal n and reflection vector r of every pixel of the sphere as follows XYZ -> RGB. Also write out the n and r images respectively as .ppm images by mapping the -1.0 to +1.0 range of n and r to the 0 – 255 range of .ppm.
        - Use r to index into the latlong map (Cartesian to polar coordinates conversion) and shade the pixel on the sphere with the corresponding pixel in the latlong map.

- For Cartesian coordinates Y is up, X is pointing right and Z is pointing out of the XY plane. For spherical coordinates of the latlong map, Width varies along phi = 0 – 2*PI, and Height varies along theta = 0 – PI.
- Save the rendered 511x511 image **m** in both .pfm format and .ppm formats. For .ppm format, simply clamp values above 1.0 to 1.0, and then map 0.0 - 1.0 to the 0 – 255 range.

- **Part 3: Latitude-longitude map extraction**
  - Given a light probe of an outdoor scene captured under a clear sunny sky (mirror_ball.pfm) in mirror ball format (511x511), create a 768x384 resolution latitude-longitude representation **l** of the environment. The corresponding .ppm file is once again given for visualization purposes only in a regular image viewer.
  - Assume the reflectivity of the sphere to be 1.0 and orthographic camera with view direction v = [0,0,1]. Once again, you do not need to do any actual ray tracing, only pixel indexing from the mirror ball.
  - Everything you did in Part 2 should help you here, except you are now doing the inverse problem!
  - You will likely find "holes" in your latitude-longitude map which is fine. Write out **l** in .pfm and .ppm formats.

- Prepare the assignment report as a WORD or .pdf document and include all LDR images (.ppm) generated for the assignment in the report along with the settings used. Submit your report document including the .ppm images and all the generated images (.pfm and the .ppm versions) of the results as a zipped file on the CATE system.

IMPORTANT: Please write a good report with sufficient explanation of your implementation and the results obtained for each part as the TA will grade your assignment based only on the report and the data submitted but will not examine the code! We reserve 10% of the marks for the quality of the report.