6.837 Intro to Computer Graphics
Assignment 7: Supersampling and Antialiasing

In this assignment, you will add antialiasing to your ray-tracer. You will use super-sampling and filtering to alleviate jaggies and Moire patterns. You will experiment with different sampling patterns and filters. To implement antialiasing, you will use a new image representation that stores multiple samples per pixel. A filter class will then use this representation to compute the final image. For extra credit, you can implement a simple pre-filtering technique for Perlin noise in order to band-limit the signal before sampling.

Tasks

- First you'll need to hack up your raytracer a bit. Instead of directly storing the colors computed per pixel with `RayTracer::TraceRay` into the `Image` class, you'll store lots of color samples (computed with `RayTracer::TraceRay`) into a `Film` class. We've provided the `Film` and `Sample` classes:
  - sample.h
  - film.h
  - film.C

  The `Sample` class stores a 2D coordinate offset from (0,0)->(1,1) within the pixel and the color of the sample. The `Film` stores `width*height*num_samples` samples. We will use a constant number of samples per pixel, which will be specified on the command line (see below). The `Film` constructor initializes all the sample colors black:

  ```
  Film(int _width, int _height, int _num_samples);
  ```

- You will implement various sampling strategies, which you will encapsulate in an abstract `Sampler` class. The constructor takes a single integer argument, the number of samples per pixel. The key function of the `Sampler` class is:

  ```
  virtual Vec2f getSamplePosition(int n) = 0;
  ```

  which returns the 2D offset for the specified sample.

- Derive a sub-class of Sampler called `RandomSampler` which assigns random locations inside the pixel to each sample. You'll need to add a number of command line options to your raytracer for this assignment. First add `-random_samples <num_samples>` that specifies the number of samples per pixel. Adapt your main rendering loop to use the sample positions specified by the `Sampler`. Cast your rays from these sample positions and store the resulting color into the `Film` instance with the function:

  ```
  void Film::setSample(int x, int y, int i, Vec2f offset, Vec3f color);
  ```

  `x` and `y` are the integer pixel coordinates, `i` is the sample number, `offset` is the sample position within the pixel, ranging from (0,0) -> (1,1), and `color` is the value returned by `RayTracer::TraceRay` for that sample.

- To visualize your sampling pattern add the command line option: `-render_samples <filename> <zoom_factor>`. If this option is specified, you should call the provided function:

  ```
  void Film::renderSamples(char *samples_file, int sample_zoom);
  ```

  This function creates a zoomed-in version of the image plane with a pixel at each sample location. Verify that your `RandomSampler` creates a nice random distribution across the pixels. Do not try it with a large image resolution, as the zoom factor will result in a huge image. Now you can create interesting "pointillist" versions of your image. Verify that with a small zoom factor this visualization looks like the scene.

- Implement uniform and jittered sampling patterns by deriving the corresponding `UniformSampler` and `JitteredSampler` classes. These patterns are specified on the command line with `-uniform_samples <num_samples>` and `-jittered_samples <num_samples>`. If none of the 3 sampling patterns are specified on the command line, your raytracer should default to a single centered sample per pixel, as in previous assignments. This can be done with the `UniformSampler`, with a single sample.

- Next, we need to filter those samples to compute the final image. Implement an abstract `Filter` class, whose constructor takes no arguments. The key function for this class is:

  ```
  Vec3f Filter::getColor(int i, int j, Film *film);
  ```

  which computes the appropriately filtered color for pixel `(i,j)` from the samples stored in the `Film` instance. An important helper function for the `Filter::getColor` method is:

  ```
  virtual float Filter::getWeight(float x, float y) = 0;
  ```

  which returns the weight for point (x + pixel center, y + pixel center). In general, this function will have maximum value at the pixel center (when x=0 & y=0). `getWeight` is a pure virtual function which will be defined in the `Filter` subclasses.

  Some filters span multiple pixels, but we do not want to access all samples in the `Film`. For each filter we can have a conservative integer `supportRadius` which will indicate which pixel samples might contribute to the final color. The second helper function for the `Filter::getColor` method is:

  ```
  virtual int Filter::getSupportRadius() = 0;
  ```

  If the filter only relies on samples in the center pixel, this function returns zero. If it relies on the center pixel and the 8 neighboring pixels (9 pixels total) this function returns 1, etc. In other words, to compute the output at pixel (*i*, *j*) of the `Film`, we will take the weighted average of the samples from pixels *i-supportRadius* through *i+supportRadius* included, and from *j-supportRadius* through *j+supportRadius* included.

- Derive from `Filter` the subclasses `BoxFilter`, `TentFilter` and `GaussianFilter`. The choice of filter is specified with one of the following command line options:

  ```
  -box_filter <radius>
  -tent_filter <radius>
  -gaussian_filter <sigma>
  ```

  For the box filter, the `radius` specifies the orthogonal distance from pixel center to the boundary edges. The tent filter is a linear function with value 1 at the pixel center and 0 for points with distance further than the `radius` from the pixel center. Finally the Gaussian filter uses the normal distribution to weight pixel samples:

  $e^{-d^2/2*sigma^2}$

  You may clamp this function to zero for points further than than 2*sigma from the pixel center.

- To visualize your filter weights, add the command line option: `-render_filter <filename> <zoom_factor>`. If this option is specified, you should call the provided function:

```
void Film::renderFilter(char *filter_file, int filter_zoom, Filter *filter);
```

- Finally, finish off your raytracer by outputting the filtered image. The filtering stage will happen after you've collected & stored all the samples. It should be a simple loop which calls `Filter::getColor(int i, int j, Film *film)` for all the pixels in the final image.

  Please think about good code design as you restructure your main rendering loop. We will be grading you on coding style, as usual.

Hints

- To avoid a segmentation fault, make sure you don't try to access samples in pixels beyond the image height & width. Pixels on the boundary will have a cropped support area.

Ideas for Extra Credit

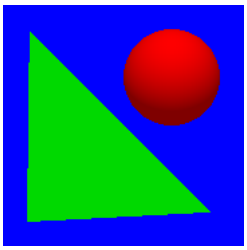
- Poisson or n-rook sampling patterns
- Adaptive sampling (careful with normalization of filters)
- Mitchell filter
- Perlin frequency cutoff: Given the image resolution & distance to the object (hit.t) you use fewer octaves of noise when computing the pixel color. This is basically a view-dependent low-pass pre-filtering of the texture. Note: there are some problems for indirect rays and high curvature.

Input Files

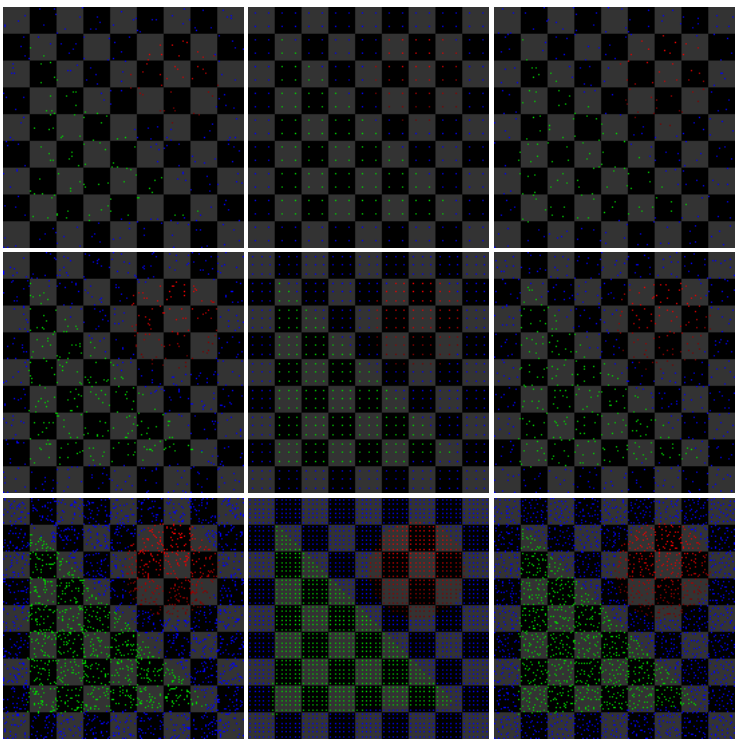
- scene7_01_sphere_triangle.txt
- scene7_02_checkerboard.txt
- scene7_03_marble_vase.txt
- scene7_04_6.837_logo.txt
- scene7_05_glass_sphere.txt
- scene7_06_faceted_gem.txt

Sample Results

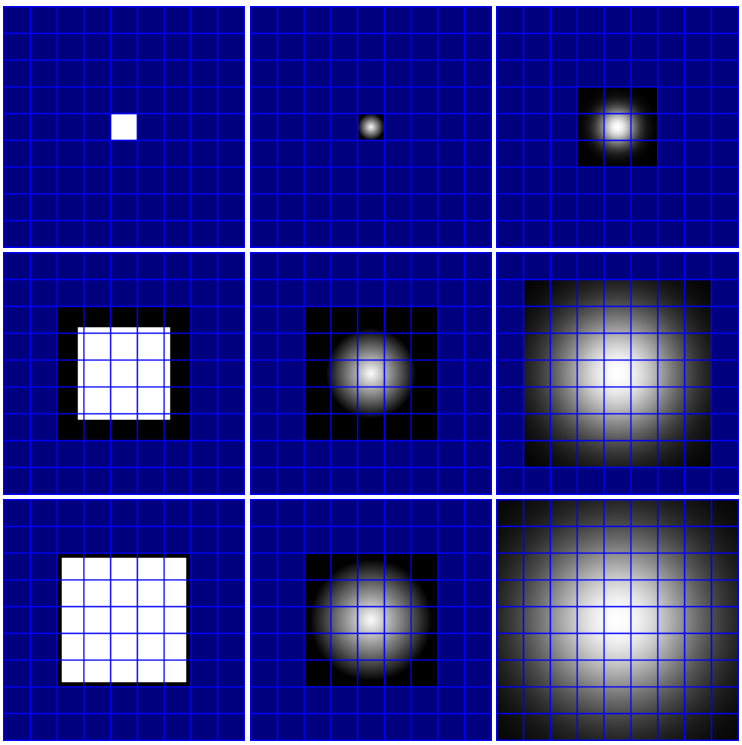
```
raytracer -input scene7_01_sphere_triangle.txt -size 180 180 -output output7_01.tga
```



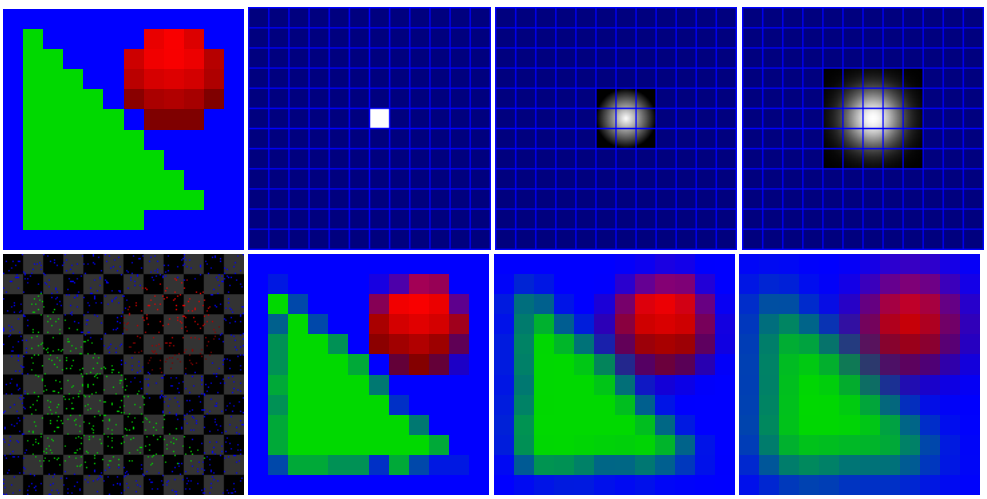
```
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_samples samples7_01a.tga 20 -random_samples 4
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_samples samples7_01b.tga 20 -uniform_samples 4
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_samples samples7_01c.tga 20 -jittered_samples 4
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_samples samples7_01d.tga 20 -random_samples 9
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_samples samples7_01e.tga 20 -uniform_samples 9
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_samples samples7_01f.tga 20 -jittered_samples 9
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_samples samples7_01g.tga 20 -random_samples 36
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_samples samples7_01h.tga 20 -uniform_samples 36
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_samples samples7_01i.tga 20 -jittered_samples 36
```

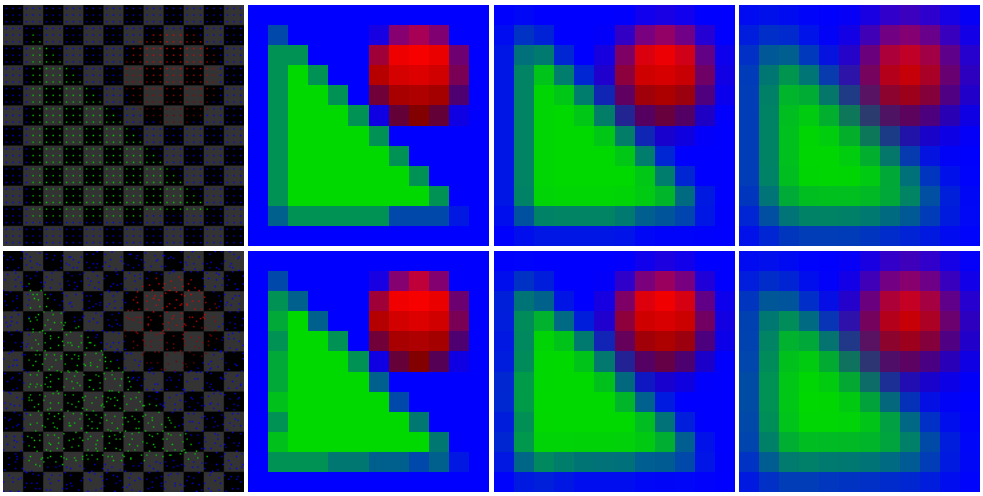


```
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_filter filter7_01a.tga 20 -box_filter 0.5
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_filter filter7_01b.tga 20 -tent_filter 0.5
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_filter filter7_01c.tga 20 -gaussian_filter 0.5
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_filter filter7_01d.tga 20 -box_filter 1.7
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_filter filter7_01e.tga 20 -tent_filter 1.7
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_filter filter7_01f.tga 20 -gaussian_filter 1.7
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_filter filter7_01g.tga 20 -box_filter 2.3
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_filter filter7_01h.tga 20 -tent_filter 2.3
raytracer -input scene7_01_sphere_triangle.txt -size 9 9 -render_filter filter7_01i.tga 20 -gaussian_filter 2.3
```

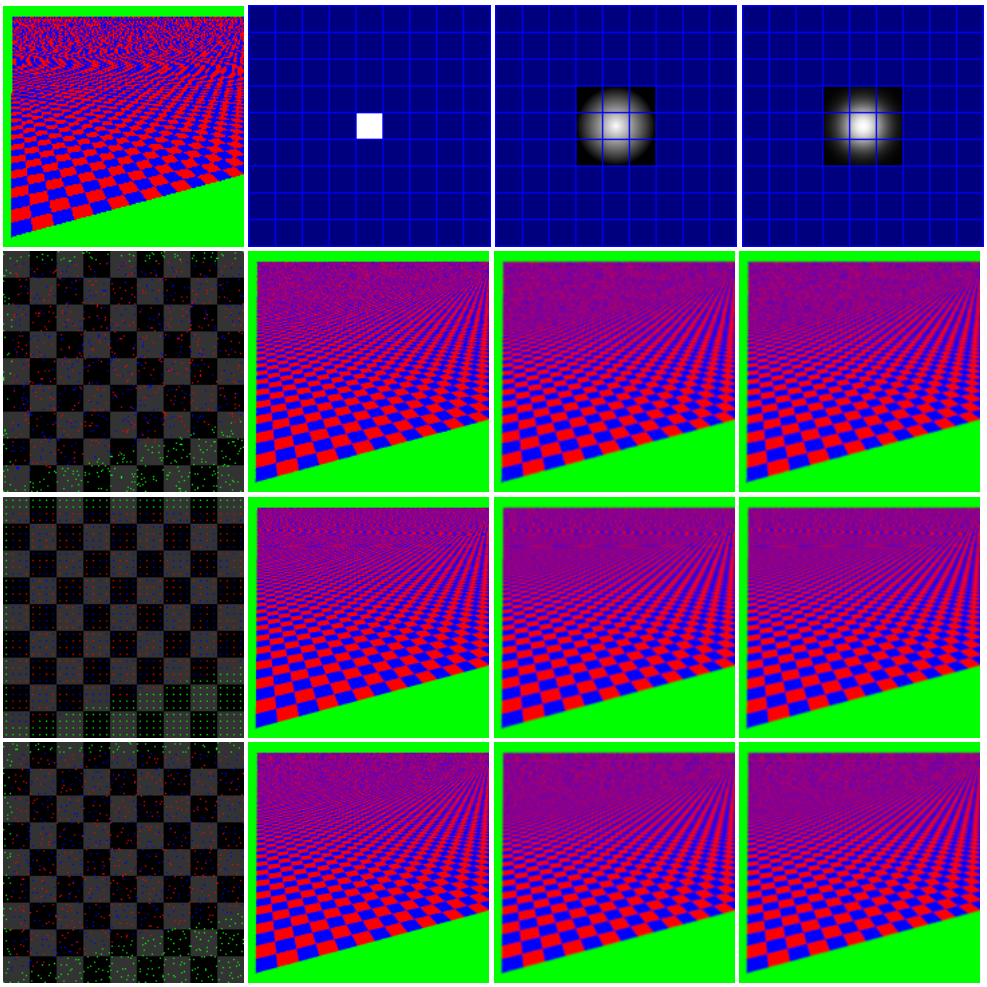


```
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -output output7_01_low_res.tga
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -render_samples samples7_01a_low_res.tga 15 -random_samples 9
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -render_samples samples7_01b_low_res.tga 15 -uniform_samples 9
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -render_samples samples7_01c_low_res.tga 15 -jittered_samples 9
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -render_filter filter7_01a_low_res.tga 15 -box_filter 0.5
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -render_filter filter7_01b_low_res.tga 15 -tent_filter 1.5
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -render_filter filter7_01c_low_res.tga 15 -gaussian_filter 1.0
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -output output7_01a_low_res.tga -random_samples 9 -box_filter 0.5
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -output output7_01b_low_res.tga -random_samples 9 -tent_filter 1.5
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -output output7_01c_low_res.tga -random_samples 9 -gaussian_filter 1.0
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -output output7_01d_low_res.tga -uniform_samples 9 -box_filter 0.5
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -output output7_01e_low_res.tga -uniform_samples 9 -tent_filter 1.5
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -output output7_01f_low_res.tga -uniform_samples 9 -gaussian_filter 1.0
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -output output7_01g_low_res.tga -jittered_samples 9 -box_filter 0.5
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -output output7_01h_low_res.tga -jittered_samples 9 -tent_filter 1.5
raytracer -input scene7_01_sphere_triangle.txt -size 12 12 -output output7_01i_low_res.tga -jittered_samples 9 -gaussian_filter 1.0
```

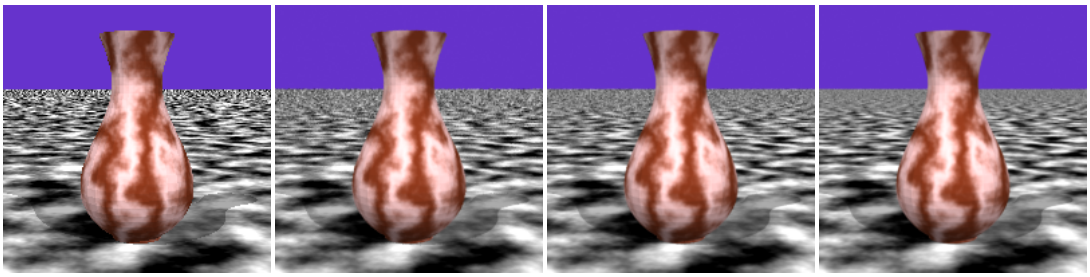




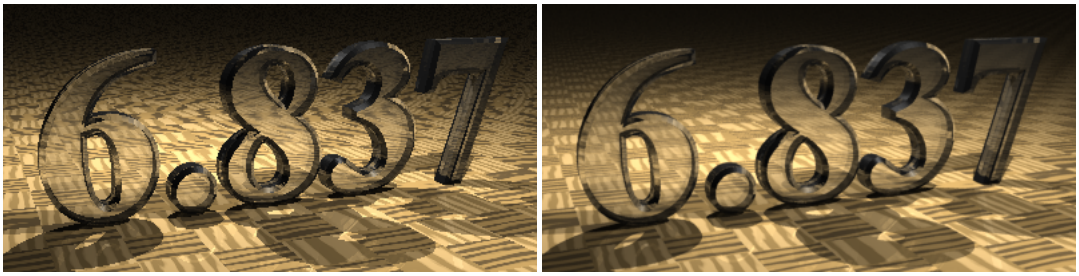
```
raytracer -input scene7_02_checkerboard.txt -size 180 180 -output output7_02.tga
raytracer -input scene7_02_checkerboard.txt -size 9 9 -render_samples samples7_02a.tga 20 -random_samples 16
raytracer -input scene7_02_checkerboard.txt -size 9 9 -render_samples samples7_02b.tga 20 -uniform_samples 16
raytracer -input scene7_02_checkerboard.txt -size 9 9 -render_samples samples7_02c.tga 20 -jittered_samples 16
raytracer -input scene7_02_checkerboard.txt -size 9 9 -render_filter filter7_02a.tga 20 -box_filter 0.5
raytracer -input scene7_02_checkerboard.txt -size 9 9 -render_filter filter7_02b.tga 20 -tent_filter 1.5
raytracer -input scene7_02_checkerboard.txt -size 9 9 -render_filter filter7_02c.tga 20 -gaussian_filter 0.6
raytracer -input scene7_02_checkerboard.txt -size 180 180 -output output7_02a.tga -random_samples 16 -box_filter 0.5
raytracer -input scene7_02_checkerboard.txt -size 180 180 -output output7_02b.tga -random_samples 16 -tent_filter 1.5
raytracer -input scene7_02_checkerboard.txt -size 180 180 -output output7_02c.tga -random_samples 16 -gaussian_filter 0.6
raytracer -input scene7_02_checkerboard.txt -size 180 180 -output output7_02d.tga -uniform_samples 16 -box_filter 0.5
raytracer -input scene7_02_checkerboard.txt -size 180 180 -output output7_02e.tga -uniform_samples 16 -tent_filter 1.5
raytracer -input scene7_02_checkerboard.txt -size 180 180 -output output7_02f.tga -uniform_samples 16 -gaussian_filter 0.6
raytracer -input scene7_02_checkerboard.txt -size 180 180 -output output7_02g.tga -jittered_samples 16 -box_filter 0.5
raytracer -input scene7_02_checkerboard.txt -size 180 180 -output output7_02h.tga -jittered_samples 16 -tent_filter 1.5
raytracer -input scene7_02_checkerboard.txt -size 180 180 -output output7_02i.tga -jittered_samples 16 -gaussian_filter 0.6
```



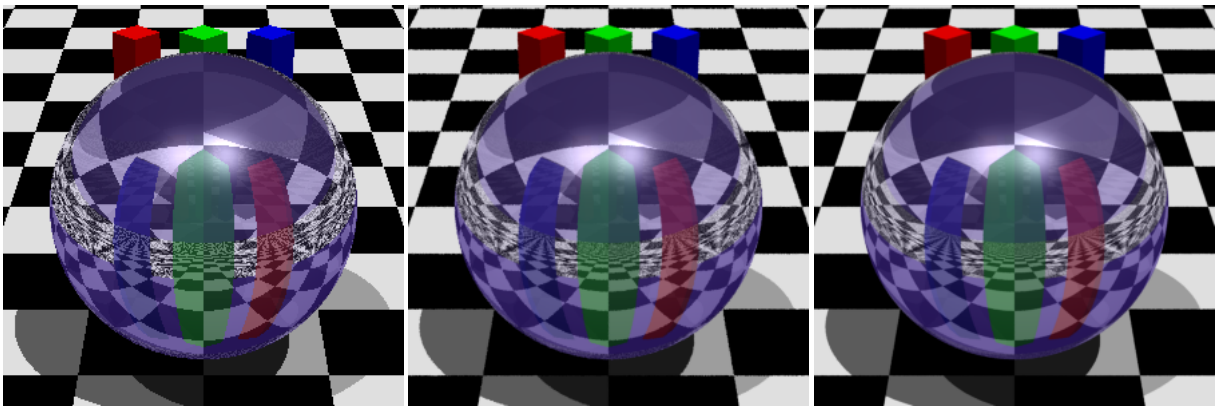
```
raytracer -input scene7_03_marble_vase.txt  -size 200 200 -output output7_03a.tga -grid 15 30 15 -shadows
raytracer -input scene7_03_marble_vase.txt  -size 200 200 -output output7_03b.tga -grid 15 30 15 -shadows -jittered_samples 4 -gaussian_fi
raytracer -input scene7_03_marble_vase.txt  -size 200 200 -output output7_03c.tga -grid 15 30 15 -shadows -jittered_samples 9 -gaussian_fi
raytracer -input scene7_03_marble_vase.txt  -size 200 200 -output output7_03d.tga -grid 15 30 15 -shadows -jittered_samples 36 -gaussian_f
```



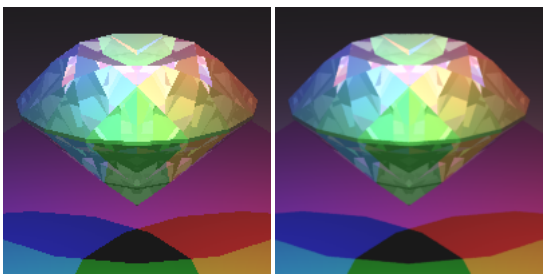
```
raytracer -input scene7_04_6.837_logo.txt   -size 400 200 -output output7_04a.tga -shadows -shade_back -bounces 5 -weight 0.01 -grid 80 3(
raytracer -input scene7_04_6.837_logo.txt   -size 400 200 -output output7_04b.tga -shadows -shade_back -bounces 5 -weight 0.01 -grid 80 3(
```



```
raytracer -input scene7_05_glass_sphere.txt -size 300 300 -output output7_05a.tga -shadows -shade_back -bounces 5 -weight 0.01 -grid 20 2(
raytracer -input scene7_05_glass_sphere.txt -size 300 300 -output output7_05b.tga -shadows -shade_back -bounces 5 -weight 0.01 -grid 20 2(
raytracer -input scene7_05_glass_sphere.txt -size 300 300 -output output7_05c.tga -shadows -shade_back -bounces 5 -weight 0.01 -grid 20 2(
```



```
raytracer -input scene7_06_faceted_gem.txt -size 200 200 -output output7_06a.tga -shadows -shade_back -bounces 5 -weight 0.01 -grid 20 20
raytracer -input scene7_06_faceted_gem.txt -size 200 200 -output output7_06b.tga -shadows -shade_back -bounces 5 -weight 0.01 -grid 20 20
```



See the main Assignments Page for submission information.
