Trace.js User Documentation

Version 0.0.2

Prologue: What is a Ray Tracer?

Ray tracing is a technique used in computer graphics typically used to draw three-dimensional scenes. At a high-level it works by shooting geometric rays from a virtual camera into a scene of objects you want to draw. If one of the rays intersects an object, it calculates the light at that intersection point and outputs a color to the final image. It then continues to bounce rays throughout the scene until all the pixels in the final image are colored.

Index

Quick Reference

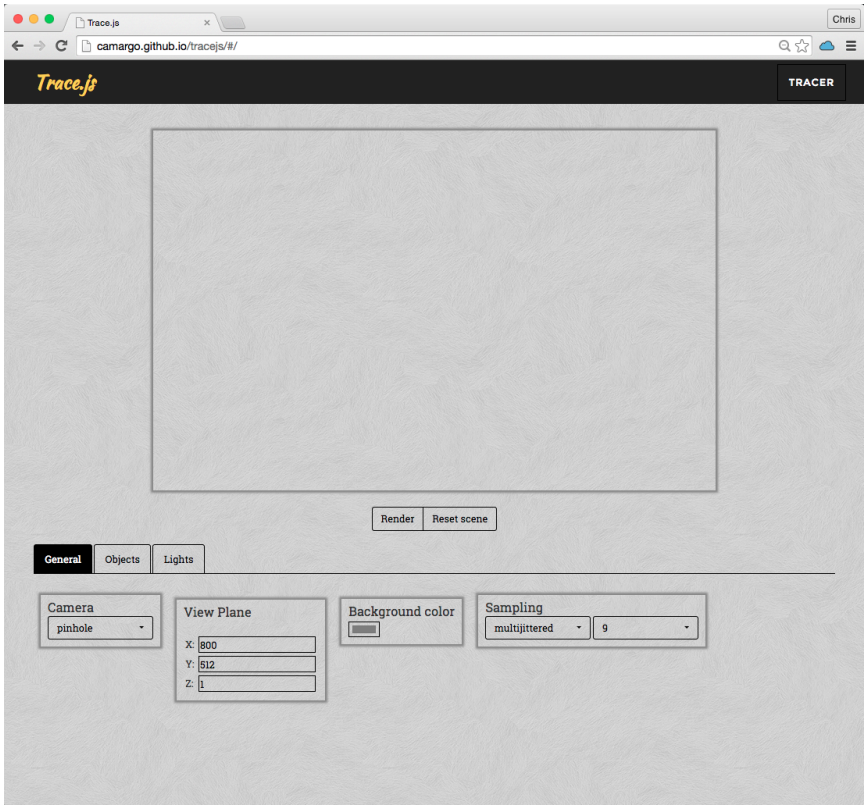
This section provides a list of available operations in Trace.js that can be used to specify a scene.

1. **BRDFs** - The *Bidirectional Reflectance Distribution Functions*. These describe how light is reflected or scattered on a given material.
2. **Camera** - Provides the user with a virtual camera to view the scene. Options include two-dimensional (2D) orthographic projection, and three-dimensional (3D) perspective projection. The 3D camera is referred to as a pinhole camera for brevity.
3. **Geometric Objects** - Objects that the user can place directly in scene. These currently include Planes, Spheres, Triangles, and a Torus.
4. **Lights** - Provides the user with virtual lights to light the scene. Options include:
    a. Directional Lights - These light the scene uniformly in every direction no matter where they are placed.
    b. Point Lights - These light the scene in a specific direction specified by the user.
    c. Ambient Light - This is extra light added to the scene that is used if the user wants to brighten the scene a little.
5. **Materials** - These are the materials specified by the user that the Geometric Objects are made out of. Options include Matte (a chalk-like material), Phong (a plastic-like material) , and Reflective (a reflective material).
6. **Samplers** - Allows the user to specify how many rays are traced for each pixel. This allows for elimination of Anti-Aliasing artifacts (typically referred to as jaggies), which makes for a smoother image overall. Trace.js implements Regular and Multi-Jittered sampling. Outlining how these techniques work is outside the scope of this document.

7. **Tracers** - Provides the user with a choice of ray-tracing technique. Currently we support Ray Casting (non-recursive ray propagation) and Whitted Ray Tracing (recursive ray propagation).

Tutorial

This section includes a full walkthrough for using Trace.js. To access the application visit http://camargo.github.io/tracejs/#/. Your screen should look something like Figure 1 below.

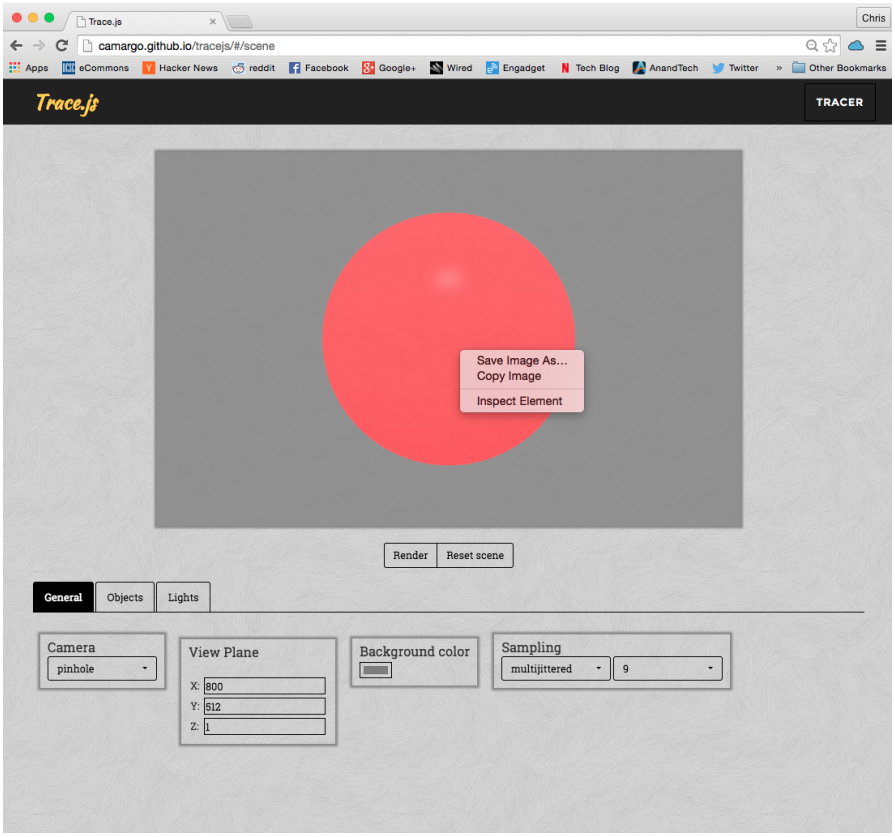


(Figure 1 - Trace.js Main Page)

There are a couple of important concepts to notice from this figure. The first is the **render canvas**, which is the large blank rectangle in the center of the page. This is where the final rendered image will appear.

Below the render canvas are two buttons labeled **Render** and **Reset Scene**. The Render button renders the scene with the configurations given in the control panel below - the control panel will be explained in detail shortly. The Reset Scene button clears the canvas to it's initial state like in Figure 1.

It should be noted here that after an image is rendered you can right click on the render canvas to save the image to your machine (see Figure 2 below).



(Figure 2 - Saving a Rendered Image)

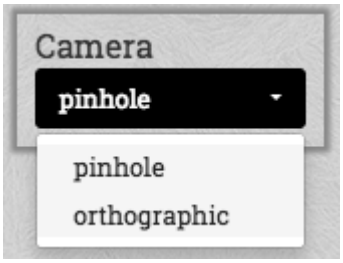
Next we will describe the **control panel**, which allows you to customize how the scene is rendered.

The control panel is broken down into three tabs: General, Objects, and Lights. See Figure 3 below.

(Figure 3 - The Control Panel - General Tab)

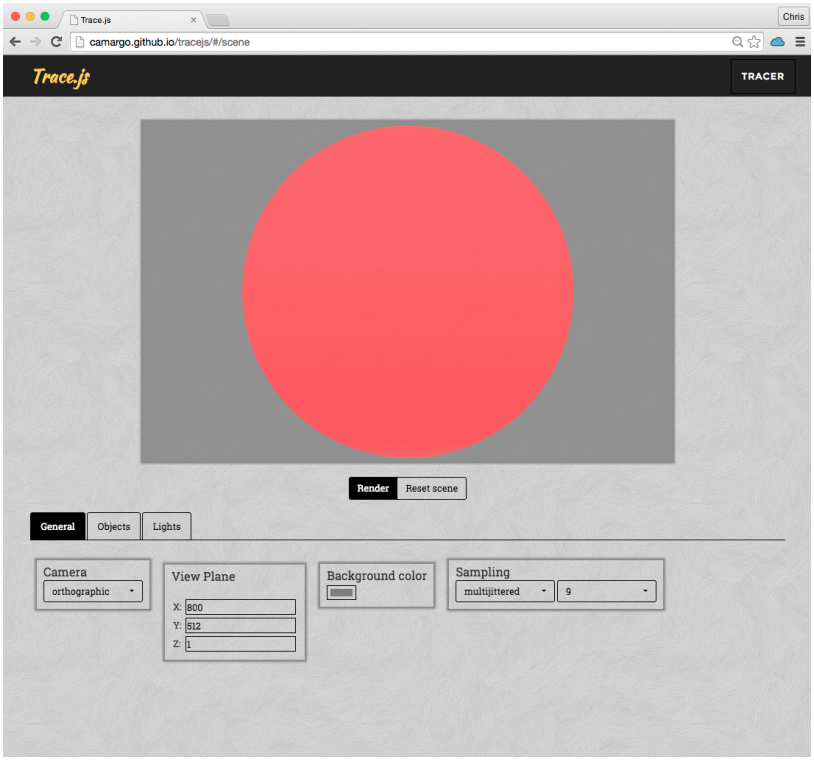
The **General** tab contains controls for customizing the camera, view plane, background color, and type of sampling.

There are two types of **Cameras** supported by Trace.js: **Orthographic** and **Pinhole**. To change the camera type use the drop down in the control panel (See Figure 4).



(Figure 4 - Camera Selection)

The default camera used is the pinhole camera. This gives you a 3D perspective of the scene much like how like humans perceive the world in real life. The orthographic camera on the other hand gives you a 2D - or flat - view of the scene (see Figure 5).

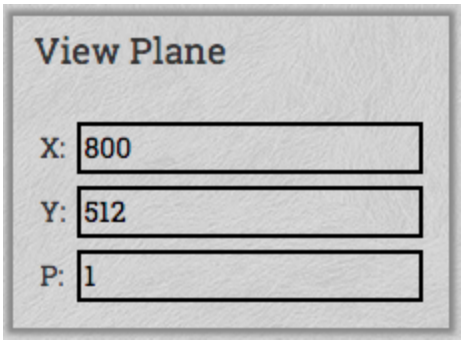


(Figure 5 - Orthographic Projection)

Notice how the 3D perspective has been lost from Figure 2 to Figure 5. Both these scenes render exactly the same sphere, the only difference being the camera used to render them.

It is important to note here that every time you change a configuration in the control panel you much hit the Render button again to re-render the scene and see the changes.

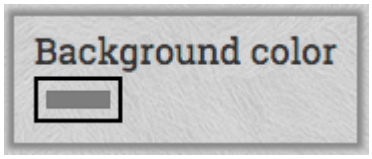
The next portion of the control panel gives you control over the **View Plane**, which describes the size of your rendered image (see Figure 6 below).



(Figure 6 - View Plane Control)

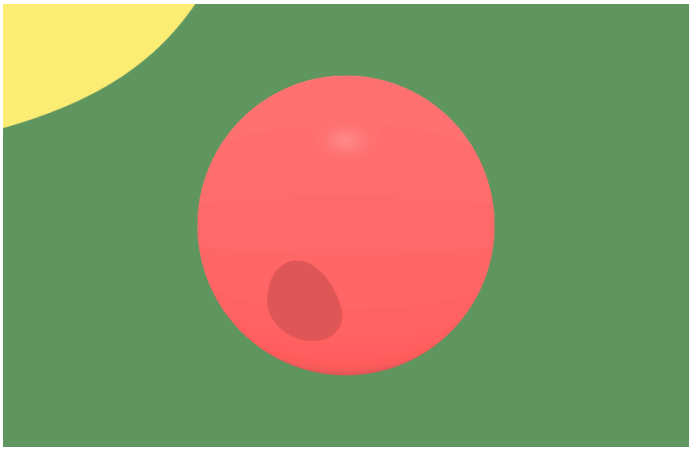
The **X** and **Y** values change the width and height of the render canvas respectively.The **P** value changes the pixel size (this is typically left at 1 unless you are going for a specific effect). Changing the pixel size has the same effect as zooming the image in or out.

The next control panel piece controls the **Background Color** of the rendered image - the default being grey (see Figure 7 below).



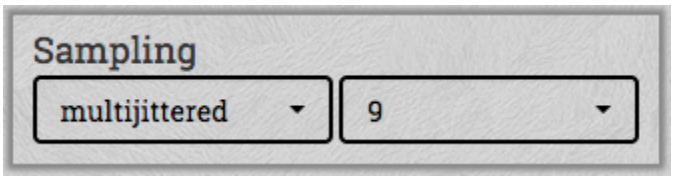
(Figure 7 - Background Color Control)

Changing the background to green for example would give an image like in Figure 8:



(Figure 8 - Green Background)

Next in the control panel are the **Sampling** controls. These help in anti-aliasing (http://en.wikipedia.org/wiki/Aliasing) the image - sometimes referred to as removing “jaggies.” The default settings fully anti-alias the image using a technique called multijittered sampling (See Figure 9 below). Note that the details of how sampling works is beyond the scope of this tutorial.



(Figure 9 - Default Sampling Settings)

The drop-down on the left gives the sampling technique used, and the drop-down on the right gives how many samples you want per pixel. As an example let’s use regular sampling with 1 sample per pixel. This gives an image like in Figure 10:

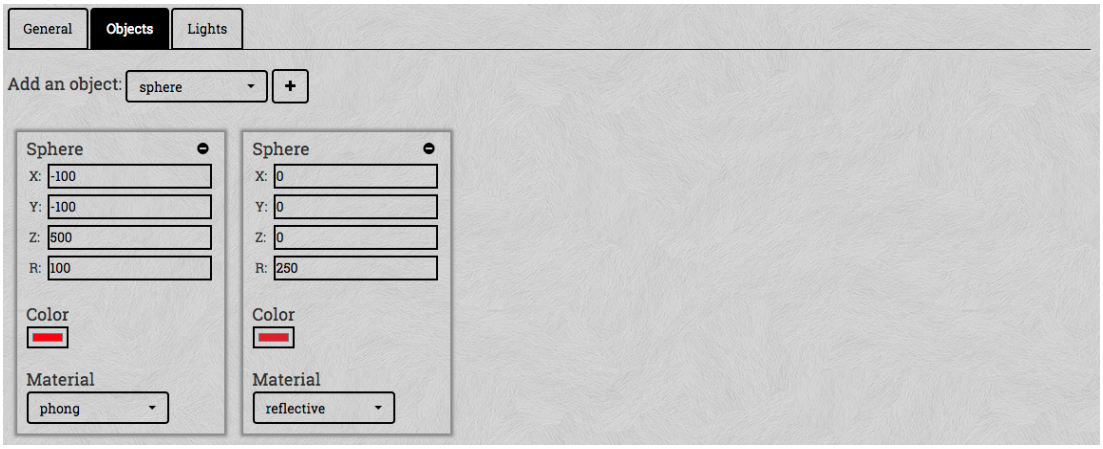
(Figure 10 - Regular Sampling, 1 Sample Per Pixel)

Here you can see the very rough, jagged, curve in the sphere. If we move back to the default settings of multijittered sampling with 9 samples per pixel we see results like in Figure 11:

(Figure 11 - Multijittered Sampling, 9 Samples Per Pixel)

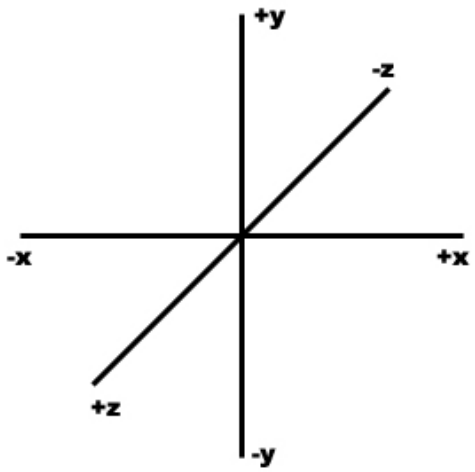
You can see that even zoomed in, most of the jagged detail from Figure 10 are gone. This is the benefit we get from advanced sampling.

The next tab of the control panel is the **Objects** tab. This tab allows you to add new objects to the scene and changes properties of those objects like their **position**, **color**, and **material**. See Figure 12 below:



(Figure 12 - The Control Panel - Objects Tab)

It should be noted here for placing objects that Trace.js uses a right-handed coordinate system (see Figure 13). In this system the +Z axis comes out of the screen and the -Z axis goes into the screen. The +X axis goes to the right and the +Y goes directly up.



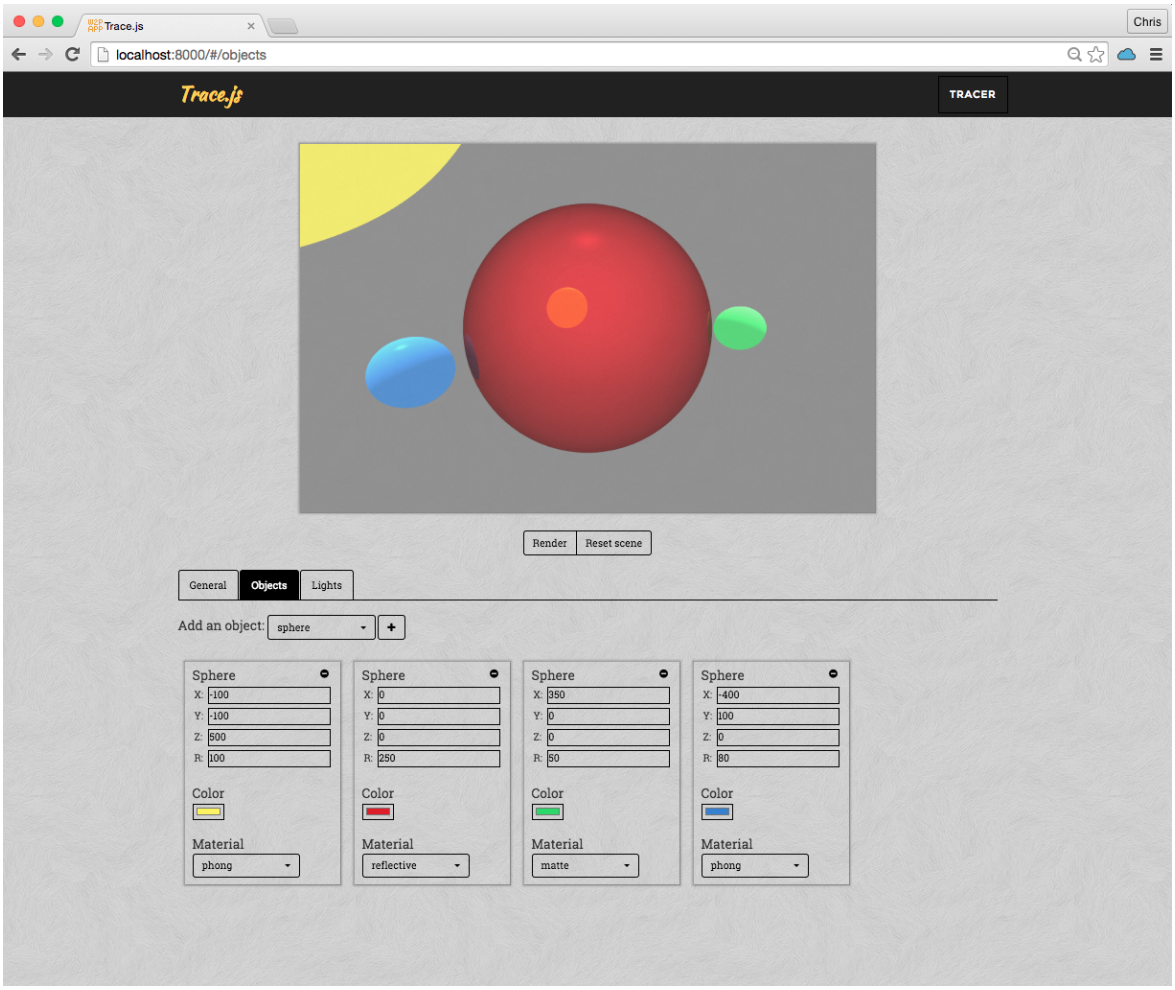
(Figure 13 - Right Handed Coordinate System)

When playing with the positions of objects it is important to keep these conventions in mind. You can see that by default two spheres are placed in the scene, one at the origin and one in the top left of the coordinate system.

In addition to changing the objects color in this panel, Trace.js also allows you to change the objects material. There are three materials currently supported: **Matte**, **Phong**, and **Reflective** (see Figure 14).

(Figure 14 - Object Color and Materials)

Matte does not have any sharp reflections - kind of like chalk. Phong gives a plastic look. And reflective reflects its surroundings on its surface. Try playing around with different materials and colors to see what you can come up with. Figure 15 gives a look at spheres with a number of different colors and materials.



(Figure 15 - Sphere Color and Material Experiments)

It should be noted here that to add a new object to the scene simply use the ‘Add an Object’ dropdown.

The final tab of the control panel is the **Lights** tab, which allows you to control the types of lights in your scene, where those lights are places, and their color (see Figure 16).

To add a new light to the scene use the ‘Add a light’ dropdown. You can specify the light location in the same way you specified the object location. The point light is the default light placed in the scene. This would be the equivalent to a light close by like when you’re in an artificially lit room. A directional light is a light that is simulated to be infinitely far away. You can think of it as standing outside and being lit from the sun.

(Figure 16 - The Control Panel - Lights Tab)

Try playing around with different lighting options to see what you can come up with.

You can check out development progress on our Github here:
https://github.com/camargo/tracejs

Happy Tracing!