CS130 - LAB - Pixel Traces

Name: _____ SID: _____

Pixel Tracing is a useful debugging technique that can help you check individual parts of your ray tracing program. Pixel Tracing, once setup properly, will allow you to print all relevant information about when your program is doing for a single pixel in your image. This output can be compared to the pixel trace files provided with your test cases, which can be a quick method to check that you are properly implementing your functions.

1 Setup

In order to set up an effective pixel trace, you will want to have output statements that only print when running a pixel trace. This will be important because printing these debug outputs for every pixel every time you run the program will make it difficult, if not impossible, to properly locate the output that you are interested in. When you have the pixel trace output statements properly set up, you will have strong foundation for debugging and testing future ray tracing assignments. The most effective way to set up the pixel trace outputs is to use the helper code that we have provided in the file `misc.h`:

```
// Useful for creating indentation in pixel traces
struct Debug_Scope
{
    static bool enable;
    static int level;

    Debug_Scope(){level++;}
    ~Debug_Scope(){level--;}
};

// This routine is useful for generating pixel traces.  It only prints when t
// desired pixel is being traced.
template<class... Args>
static void Pixel_Print(Args&&... args)
```

```
{
    if(!Debug_Scope::enable) return;
    for(int i=0;i<Debug_Scope::level;i++) std::cout<<"  ";
    (std::cout<<...<<std::forward<Args>(args))<<std::endl;
}

/*
Example usage of these routines:

void foo(const vec3& pt, const Ray& ray)
{
    Debug_Scope scope;
    Pixel_Print("calling foo; pt: ",pt,"; ray: ",ray);

    blah;
    blah;
    blah;
}
*/
```

As you can see, this file also contains an example usage.

The `Debug_Scope` object stores the static variable storing whether the pixel trace is enabled, and the declaration will temporarily increment the `level` variable, which will be used to indent your output to organize according to function call depth. You should add this scope delcaration to any function where you are calling `Pixel_Print`.

The `Pixel_Print` function will print any number of arguments that you give it, but it will only do so if the pixel trace has been enabled. Use this function to generate the output for your pixel traces.

The pixel trace sample files are text files included with the normal test cases, and end in `-t.txt`. For example, `00-t.txt` is a pixel trace sample corresponding with test file `00.txt`.

To get full credit for this portion of the assignment, you must implement pixel trace code that will output pixel trace debug information in the same way and the same format as both pixel trace samples included with the previous homework assignment.