

Writing Single-Line Parsers in C++

Niels Joubert, CS184 TA, UC Berkeley

2008-09-30

You've just been asked to write a parser to pull in some ascii-text format into your raytracer. Luckily enough, the standard template library for C++ has some *really good* text parsing functionality that makes parsing files a breeze. Since the OBJ file format is a line-by-line format, this document will focus on writing parsers for such a case.¹

0.1 The Standard Template Library's string and stream functionality

Silicon Graphics did not only contribute fantastic advances in computer graphics to the developer community, they also wrote a large bundle of C++ classes that became the equivalent of Java's built-in API for C++. We will use the `ifstream` class to parse an input file into `stringstream` objects. Notice the abundance of the `stream` class' subclasses.

We use streams since they have two very useful operators defined on them:

```
operator>>    Extracts formatted data
operator<<    Insert data with format
```

By default, the `>>` extract operator considers whitespace to be a separator, and will return something of the type of variable it is used on. In other words, you can do the following:

```
// instream contains a string of the format "1.0 1.0 1.0 1 2 3"
void parseCommand(stream & instream) {}
    double x,y,z;
    int i,j,k;
    instream >> x >> y >> z >> i >> j >> k;
}
```

The above will populate `x`, `y` and `z` with the correct double values of 1.0 and `i`, `j` and `k` with the expected 1, 2 and 3 respectively. The opposite is also valid - using `<<` to pipe out different data types to file.

0.2 The OBJ file format²

The obj file format defines geometry on a line-by-line basis of commands. Each line consists of an **operator** followed by one or more **operands**. The subset of the obj file format that is very applicable to the raytracer project is as follows:

```
FILE STRUCTURE:
[operator] [operands]\n

LINE STRUCTURE:
[operator]      [operands]
v              x y z
f              i j k
```

¹This document does not describe the only way to write parsers, and not even the best way, but it is easy!

²See <http://local.wasp.uwa.edu.au/~pbourke/dataformats/obj/> for a full discussion of the OBJ format.

(x, y, z) gives the world-space coordinates of a vertex. i, j and k gives the indices of three vertices that make up a triangle, where indices are counted from **1** to **n** where 1 is the first vertex read in from the file and n is the last. You can easily extend this format to support everything your raytracer does.

0.3 Using ifstream

ifstream allows us to represent a file on disk as a stream. Thus we can do the following (Please note the ... are to indicate that you can (and probably will) be passing around other arguments as well.):

Listing 1: Using ifstream

```
void parseScene(string filename, ...) {
    char line[1024]; //We create some temporary storage
    ifstream inFile(filename.c_str(), ifstream::in); //Open as stream
    if (!inFile) {
        cout << "Could not open given file" << filename;
        exit(1);
    }
    while (inFile.good()) {
        inFile.getline(line, 1023); //Read line into temporary storage
        if (!parseLine(string(line), ...)) //Do something with line
            exit(1); //An error occurred?
    }
    inFile.close();
}
```

0.4 Using stringstream

We could run some regex expression on each line, but regex is much uglier than streams, so we can define each line as a stream, and parse that line with the operators explained in section 0.1.

Listing 2: Using stringstream

```
bool parseLine(string line, ...) {
    string operator;

    if (line.empty())
        return true;
    stringstream ss(stringstream::in | stringstream::out);
    ss.str(line);
    ss >> operator;
    if (operator[0] == '#') { //access strings as arrays.
        return true;
    } else if (operator.compare("v") == 0) {
        double x, y, z;
        ss >>x >>y >>z; //Now you have an x,y,z as doubles.
        ...
    }
    if (ss.fail())
        return false;
    return true;
}
```