# SpectralData Documentation

## What This Code Does

This class uses the USGS\_SpectralDataReader class to open and load the splib06a library from USGS. The class then takes the records in that library (known as splib06a records or specpr records) and then concatenates the records that go together to create one “Spectral Record” with all the data in it for the material that has been characterized by the multiple specpr records. The end result is an ability to get all the spectral information for an sample material that was characterized in the splib06a dataset.

### Limitations

The class only reads and evaluates reflectance data. It is not designed to interpret the error or precission data in the splib06a dataset.

## Algorithm Walkthrough

This class is a straight forward read and concatenate processor that consolidates multiple records in splib06a into a single Spectral record. It does this by checking each USGS Data Record sequentially to determine if the current record is a binary continuation of the previous record. If it is a binary continuation of the previous record, the class concatenates the current USGS record data with the previous USGS record data.

## Class Style Guide

1. Readability was chosen over efficiency so that debuggers and porters could follow the algorithm easily
2. the class members and methods are grouped by overall function
3. All member variables are private.
4. All member variables have public get/set accessor functions.
5. When appropriate member variables have update accessor functions.
6. Static variables start with a lowercase "s".
7. Member variables start with a lowercase "m".
8. Local variables start with a lowercase "l".
9. Loop variables are a single lowercase letter. In our case we used "i"
10. Method return variables are all labeled "rvalue". They are used to insure that a method only has 1 exit point.
11. Return values (rvalue) are defaulted to indicate failure either through assignment to "false" or "NaN"
12. Accessor methods start with a lowercase "a".
13. Calculation methods start with a lowercase "c".
14. Void returns are discouraged.

## Class Structures

struct recordTuple{

unsigned char title[40];

int id;

int channels;

float reflectances[5000];

float wavelengths[5000];

};

The recordTuple structure is designed to hold a concatenated set of USGS specpr records to create a full spectral record.

title: the name of the material used in generating the record

id: specpr record id

channels: the number of channels used to collect the full spectral data

reflectances: array holding the spectral reflectance data

wavelengths: the wavelengths of the spectral reflectance data

\*Note\* the reflectance and wavelengths index are correlated. If the reflectance[5] = .8 and wavelengths[5] = .000002 then the sample at 2 microns would have a reflectance of .8.

struct WV3Spectra{

recordTuple USGSBase;

float SimulatedBands[numBands];

};

The WV3Spectra structure holds a recordTuple and a simulation of what a sensor with Digital Globe’s World View 3 spectral absorption characteristics would have seen if it had collected against the sample with World View 3’s 8 SWIR bands.

## Class Static Members

static const int numBands = 8;

numBands:

The number of SWIR bands that Digital Globe’s World View 3 sensor collects. At this point the value is 8.

## Class Instance Members

USGS\_SpectralDataReader mUSGSDataRecord;

map<int, recordTuple> mWavelengthsData;

map<int, int> mRecordList;

map<int, vector<int>> mReducedLibrary;

WV3Spectra mStatMoments[2];

mUSGSDataRecord:

Holds the splib06a library data for concatenation

mWavelengthsData:

Holds the wavelength reference data. The key is the splib06a index of the wavelength band’s first record

mRecordList:

Maps a splib06a library record index to the spectral record index

mRedcuedLibrary:

Holds a vector of index values that are utilized when you only want to search a subset of the library

mStatMoments:

Holds the mean and variance of each WV3 SWIR band for the entire library. Note that these values are obtained by simulating how each sample would look to WV3 and then recording the mean and variance for those simulated values.

## Class Accessor Methods

### Get Methods

recordTuple aGetSpectralRecord(int SpectralRecordIndex);

int aGetRecordListElement(int SpectralRecordIndex);

WV3Spectra aGetWV3Record(int SpectralRecordIndex);

WV3Spectra aGetStatNormedWV3Record(int SpectralRecordIndex);

int aGetNumSpectralRecords();

map<int, vector<int>> aGetReducedLibrary();

float \* aGetMoment(int index);

aGetSpectralRecord:

Returns a recordTuple corresponding to the SpectralRecordIndexth record.

aGetRecordListElement:

Converts the SpectralRecordIndex into its corresponding splib06a index.

aGetWV3Spectra:

Returns a WV3Spectra structure for the SpectralRecordIndex data.

aGetStatNormedWV3Record:

Returns a WV3Spectra structure for the SpectralRecordIndex data. Where the simulated bands have been mean subtracted and then variance divided utilizing mStatMoments.

aGetNumSpectralRecords:

Returns the total number of spectral records that this class made from the splib06a library.

aGetReducedLibrary:

Returns a copy of mReducedLibrary

aGetMoment:

Returns a pointer for the indexth statistical moment for all bands.

### Set Methods

bool aSetWavelengthData();

bool aSetRecordListMap();

bool aSetStatMoments();

aSetWavelengthData:

Loads data into the mWavelengthsData map

aSetRecordListMap:

Loads the mRecordList map

aSetStatMoments:

Cycles through the splib06a library and converts the reflectance data to simulated World View 3 reflectance data. It then calculates the mean and variance for the values in each band independently and returns all 16 of those values in an array.

mStatMoments[0] = means, mStatMoments[1] = variances

## Class Calculation Methods

This class has no calculation methods

## How To Use This Class

The constructor loads all the data and initializes everything behind the scenes. So all you have to do is instantiate the SpectraData and then make a call to one of the get methods to retrieve your data.

SpectralData lSpectralData = SpectralData();

SpectralData::WV3Spectra lWV3Data = lSpectralData.aGetStatNormedWV3Record(0);