DataAction

Summary:

Serializing objects to byte streams and back can be error prone, and difficult to maintain. Without a helper class it can be difficult to keep the read, and write procedures matched over time. However, operating the helper class can be as much work in programming as just writing the code. Thus, the goal of this project is to make a framework that is quick and simple to implement for the programmer, and will make read-sequence and write-sequence into a single sequence so that any change will keep the read & write consistent. As a bonus side note, since the framework can also be used as an undo helper. The object can be transferred to a uint8[] (“byte[]”) held in memory, and then used to restore an objects primary data-types to the stored state.

Examples:

Example 1 – A basic data type to file and read back from file.
Example 2 – Mixed basic data types to file and read back from file.
Example 3 – A class object with mixed primary data inside& write/read from file.
Example3b – A class object with mixed primary data undo example.

Example10 – {Todo} GTK3 Example (Open, Save & Undo)

Example1 Usage: (Primary Data – Write to file)

```
.. // Code Fragment

   double d1 = 23; ///Double is 64bits – 8Byte

   dataAction myout = new dataAction.forSave("test_1.dat", 8);

   myout.as_double64 ( ref d1 );

   myout.writeToFile();
```

Example1 Usage: (Primary Data – Read from file)

```
.. // Code Fragment

   double d2 = 0;

   dataAction inFile = new dataAction.OpenFile("test_1.dat");

   inFile.as_double64( ref  d2);
```

Example2 Usage: (Mixed Primary Data – Write to file)

```
.. // Code Fragment

    ///Sample Variable
    double d1 = 23;
    int32 i1 = 11;
    string name1 = "its working";

    dataAction myout = new dataAction.forSave("test_2.dat", 8+4+4+name1.length );

    myout.as_double64 ( ref d1 );
    myout.as_int32 ( ref i1 );
    myout.as_string32 ( ref name1 );

    myout.writeToFile();
```

Example2 Usage: (Mixed Primary Data – Read from file)

```
.. // Code Fragment

    //Sample to read back into
    double d2 = 0;
    int32 i2 = 0;
    string name2 = "none";

    dataAction inFile = new dataAction.OpenFile("test_2.dat");

    inFile.as_double64( ref  d2);
    inFile.as_int32( ref  i2);
    inFile.as_string32( ref  name2);

    print ("Read back is %g, %d, %s\n", d2, i2, name2);
```

Example3 Usage: (A Class with Mixed Primary Data – Write to file)

.. // Code Fragment

```
///Sample Variable
myObject o1 = new myObject();
o1.d1 = 37.1;
o1.i1  = 376;
o1.s1 = "its working";

dataAction myout = new dataAction.forSave("test_3.dat", o1.dataSize() );

o1.withData( ref myout );

myout.writeToFile();
```

Example3 Usage: (A Class with Mixed Primary Data – Read from file)

.. // Code Fragment

```
//Sample to read back into
myObject o2 = new myObject();

 dataAction inFile = new dataAction.OpenFile("test_3.dat");

o2.withData( ref inFile);

print ("Read back is %g, %d, %s\n", o2.d1, o2.i1, o2.s1);
```

Example3b Usage: (A Class with Mixed Primary Data – Save object state)

.. // Code Fragment

```
  ///Sample Variable
 myObject o1 = new myObject();
 o1.d1 = 37.1;
 o1.i1  = 376;
 o1.s1 = "its working";

 print ("Object state is %g, %d, %s\n", o1.d1, o1.i1, o1.s1);
 dataAction undo1 = new dataAction( o1.dataSize() );
 undo1.mode = dataMode.SAVE;

 o1.withData( ref undo1 );

 //User Edit - Object
 o1.d1 = 1234;
 print ("changed d1 = %g\n", o1.d1);
 o1.prt();
```

Example3b Usage: (A Class with Mixed Primary Data – Restore object state “undo”)

.. // Code Fragment

```
 print("Action.Undo()\n");
 undo1.mode = dataMode.OPEN;
 o1.withData( ref undo1 );

 o1.prt();
```