GammaLib - Change request #1780

Simplify the FITS image interface

05/30/2016 10:05 PM - Knödlseder Jürgen

Status:	New	**Start date:**	05/30/2016
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

The FITS image interface is rather complex. For example, the GFitsImageDouble class has the following public methods:

```
// Constructors and destructors
GFitsImageDouble(void);
GFitsImageDouble(const int& nx, const double* pixels = NULL);
GFitsImageDouble(const int& nx, const int& ny, const double* pixels = NULL);
GFitsImageDouble(const int& nx, const int& ny, const int& nz, const double* pixels = NULL);
GFitsImageDouble(const int& nx, const int& ny, const int& nz, const int& nt, const double* pixels = NULL);
GFitsImageDouble(const std::vector<int>& naxes, const double* pixels = NULL);
GFitsImageDouble(const GFitsImageDouble& image);
virtual ~GFitsImageDouble(void);

// Operators
GFitsImageDouble& operator=(const GFitsImageDouble& image);
double&           operator()(const int& ix);
double&           operator()(const int& ix, const int& iy);
double&           operator()(const int& ix, const int& iy, const int& iz);
double&           operator()(const int& ix, const int& iy, const int& iz, const int& it);
const double&     operator()(const int& ix) const;
const double&     operator()(const int& ix, const int& iy) const;
const double&     operator()(const int& ix, const int& iy, const int& iz) const;
const double&     operator()(const int& ix, const int& iy, const int& iz, const int& it) const;

// Methods
void              clear(void);
GFitsImageDouble* clone(void) const;
std::string       classname(void) const;
double&           at(const int& ix);
double&           at(const int& ix, const int& iy);
double&           at(const int& ix, const int& iy, const int& iz);
double&           at(const int& ix, const int& iy, const int& iz, const int& it);
const double&     at(const int& ix) const;
const double&     at(const int& ix, const int& iy) const;
const double&     at(const int& ix, const int& iy, const int& iz) const;
const double&     at(const int& ix, const int& iy, const int& iz, const int& it) const;
double            pixel(const int& ix) const;
double            pixel(const int& ix, const int& iy) const;
double            pixel(const int& ix, const int& iy, const int& iz) const;
double            pixel(const int& ix, const int& iy, const int& iz, const int& it) const;
void*             pixels(void);
int               type(void) const;
```

This class should be simplified to

```
// Constructors and destructors
GFitsImageDouble(void);
GFitsImageDouble(const GIntTuple& naxes, const double* pixels = NULL);
GFitsImageDouble(const GFitsImageDouble& image);
virtual ~GFitsImageDouble(void);
```

```
    // Operators
    GFitsImageDouble& operator=(const GFitsImageDouble& image);
    double&           operator()(const GIntTuple& indices);
    const double&     operator()(const GIntTuple& indices) const;

    // Methods
    void              clear(void);
    GFitsImageDouble* clone(void) const;
    std::string       classname(void) const;
    double&           at(const GIntTuple& indices);
    const double&     at(const GIntTuple& indices) const;
    double            pixel(const GIntTuple& indices) const;
    void*             pixels(void);
    int               type(void) const;
```

by using the GIntTuple class to encode an integer tuples or arbitrary dimension. The GIntTuple should emulate a Python tuple, so that the following code would work

```
image = GFitsImageDouble(10,10,3)
image[0,0,0] = 1.0
value = image.pixel(0,0,0)
```