Contents

1 Introduction.

HEPInventor is a graphical library built on top of the MasterSuite package from TGS. It provides an interface between the user data (arrays, histogram data structures defined by HistOO) and graphics. The aim of this package is to provide an **easy to understand** and **easy to use** way to present physics data.

The package focuses on :

- Present 2D and 3D data in a "page".
- Define the "page layout".
- Provide several ways to represent data (curve, errors bars etc ...).
- Specify graphics attributes and options.
- Display a page on screen or generated output on file (PS, IV ..).

For that three basic classes are provided:

HIPage Contains HIZones. Can be displayed on the screen (via the window class), printed via PostScript files or output as an Inventor file. The page size is defined in centimeters.

HIZone Contains Datas. Can be placed anywhere on a Page. This class is used to define the layout of the page. The Zone position in the page is given in centimeters.

HIData Does the interface between user data and graphics. User data can come from various sources. The class Data allows to disconnect the data from the graphics representation. If a new type of data has to be drawn by the HEPInventor package it is enough to create a new constructor which accepts this new data type keeping unchanged the other parts of the package.

2 Quick start.

Names

This section shows examples using the HEPInventor prototype. To run these examples, some environment variables need to be defined for OpenGl, OpenInventor and MasterSuite. The HistOO package should be also set up (it implies some setting for objectivity). We don't explain here how to set up this environment we just assume it has been done.

2.1 Simple example

The following example is the simplest one can do with HEPInventor. It opens a window then builds a page containing one empty HIZone and displays it in the window.

```
#include "HInventor.h"

int main()
{
 HIWindow W;              // create a HIWindow W (500x500 pixels)
 HIPage P;                // create a HIPage P (20x20 cm)
 HIZone Z(2.,18.,2.,18.); // create a HIZone Z
 P.AddZone(&Z);           // put the Zone Z in the HIPage P
 W.ShowPage(&P);          // build the scene graph corresponding to P
 W.Flush();               // display the HIPage P in the HIWindow W

 char c = getchar();      // <CR> to exit
 return 0;
}
```

2.2

Data from array

This example is an extension of the previous one. Now the HIZone Z contains data (sin and cos curves). Instead displaying the result on the screen it generate a PostScript file.

```
#include "HInventor.h"

int main()
{
#define N  50

 HIPage P;                        // create a HIPage P (20x20 cm)
 HIZone Z(2.,18.,2.,18.);         // create a HIZone Z
 P.AddZone(&Z);                   // put the HIZone Z in the HIPage P

 float X[N],SINX[N],COSX[N];      //
 float ANGLE = 0;                 //
 float DX    = 2*PI/N;            //
 int i;                           // generate 2 floating points
 for ( i=0; i < N; i++) {         // arrays one containing a sin and
    X[i]    = ANGLE;              // this other one a cos.
    SINX[i] = sin(ANGLE);         //
    COSX[i] = cos(ANGLE);         //
    ANGLE   = ANGLE+DX;           //
 }                                //

 HIData SINDATA(N,X,SINX);        // create a Data set SINDATA
 HIData COSDATA(N,X,COSX);        // create a Data set COSDATA

 Z.AddData(&SINDATA);             // put SINDATA in the HIZone Z
 Z.AddData(&COSDATA);             // put COSDATA in the HIZone Z

 HIPostScript PS("Ex002.ps");     //
 PS.put(&P);                      // generate the PS file Ex002.ps from P
```

```
 PS.close();                    //
 return 0;
}
```

2.3

Data from histogram

In this example the data are taken from a Histo1D. The only difference, from the graphics part point of view, is the HIData constructor used.

```
#include "HInventor.h"
#include "HepODBMS/goodies/ooSession.h"
#include "HTL/P_Histograms_1D.h"
#include "HTL/P_Histograms_2D.h"

int main()
{
#define N  100

 HIPage P;                              // create a HIPage P (20x20 cm)
 HIZone Z(2.,18.,2.,18.);               // create a HIZone Z
 P.AddZone(&Z);                         // put the HIZone Z in the HIPage P

 ooSession mySession;                   //
 mySession.Init();                      // initialise objectivity

 mySession.setUpdate();                 // set transaction mode to update
 mySession.TransStart();                // start stransaction
 // handle to the database.
 ooHandle(ooDBObj) dbH = mySession.db("SimpleTestDatabase");

 d_Double X1 = -5.;
 d_Double X2 = 5.;

 H_P_REF(P_Histo1D) H2 = new (dbH)      // create a 1D histogram
     P_Histo1D( "Two gaussians",N,X1,X2);
```

```
 float STEP = (X2-X1)/N;
 float X    = X1+STEP/2.;
 float Y;
 for ( int k=0; k < N; k++) {
    Y = 10*exp(-(X-2)*(X-2)) + 6*exp(-(X+1)*(X+1));
    H2->fill(X,Y);
    X = X+STEP;
 }


 Z.set(HIZone::titleSize,0.5);          // change the HIZone title size (cm)

 HIData D2(*H2);                         // create a HIData set D2
 Z.AddData(&D2);                        // put D2 in the HIZone Z

 mySession.TransCommit();               // commit changes to database

 HIPostScript PS("Ex003.ps");   //
 PS.put(&P);                    // generate the PS file Ex003.ps from P
 PS.close();                    //
 return 0;
}
```

2.4 More than one page in the same program

A program can create more than one page. When a page is ready, it can be displayed on the screen.

```
#include "HInventor.h"

int main()
{

 HIWindow W(600,800);      // create a window
```

```
 HIPage P1(11,18);          // create the first page
 HIPage P2(11,11);          // create the second page
 HIZone Z(2, 10, 2, 9);     // create one zone
 P1.AddZone(&Z);            // put the zone Z in the page P1
 P2.AddZone(&Z);            // put the zone Z in the page P2

 W.ShowPage(&P1);           // display the page P1
 W.Flush();                 //

 cout << "Type <CR> to continue ... " << endl; char c = getchar();

 W.ShowPage(&P2);           // display the page P2
 W.Flush();                 //

 cout << "Type <CR> to continue ... " << endl; c = getchar();
 return 0;
}
```

3 Reference Manual.

Names

A complete description of all the classes provided by HEPInventor.

3.1 Callable user classes.

Names

Classes directly usable from a user program.

3.1.1 class **HIWindow**

Names

This class allows to create and manage windows.

3.1.1.1

Public methods

Names

The public methods of the class HIWindow allow to:

- create a HIWindow:

```
HIWindow w (400, 600);
```

- render an existing HIPage:

```
HIPage p;
   ...
w.ShowPage(&p);
```

- display the graphics corresponding to the rendered HIPage:

```
w.Flush();
```

3.1.1.1.1

HIWindow (int WW=500, int WH=500)

HIWindow constructor. It takes two arguments: the HIWindow width and the HIWindow height, both in pixels. These arguments can be omitted, in that case both the HIWindow width and the HIWindow height are equal to 500 pixels. These method creates a SoXtRenderArea called "viewer".

Parameters: `WW` — HIWindow width (pixel).
`WH` — HIWindow height (pixel).

3.1.1.1.2

void **ShowPage** (HIPage *page)

This method create the scene graph for "page" by calling the Draw method of the HIPage class which call the Draw method of each HIZone contained in page, and finally each Draw method of the HIZone(s) calls the Draw method of all the Data object contained in each HIZone. Once the page scene graph is created, it is asign to the SoXtRenderArea created in the HIWindow constructor.

3.1.1.1.3

void **Flush** ()

In most OpenInventor applications, once a scene graph is created and asign to a viewer, the program usually enter a loop which allows to manipulate the scene. This is usually done with:

```
SoXt::mainLoop();
```

This is fine in some cases when the goal of the application is to manipulate a graphics scene, but it put the program in an endless loop which in some case is not good. The Flush method allows to display a scene graph and then continue the execution of the program. In the following example, Flush display the scene and then the getchar() is executed.

```
w.Flush();
char c = getchar() ;
```

3.1.2

class **HIPage**

Names

This class allows to create and manage Pages. A HIPage can be displayed in a HIWindow or translated in a output file for example in PostScript format. A HIPage define a space (cm coordinates) in which HIZones will be placed. A HIPage contains a collection of HIZone.

3.1.2.1

Public methods

Names

One can found two kind of methods in this section. Methods design to be called from the user program and methods design to be called from other classes methods. If not specified, public methods are design to be called from the user program.

3.1.2.1.1

HIPage (float PW=20., float PH=20.)

HIPage constructor. It creates a new HIPage. It takes two arguments: the page width (cm) and the page height (cm). These arguments can be omitted, in that case both the HIPage width and the HIPage height are equal to 20cm.

Parameters: `PW` — HIPage width.
`PW` — HIPage height.

3.1.2.1.2

void **AddZone** (HIZone *zone)

This method allows to add a zone in the page. It takes one argument: a pointer to the zone to be added. A HIPage can contains many HIZones. This method check also if the zone to be added is valid and if not refuse to add it. Then, if the HIZone is valid, it checked if the HIZone position fits into the page dimension if yes the HIZone is added in the page. A zone can be added in several pages. The following example add the HIZone z in the HIPage p (page size is 20x20 cm):

```
HIPage p;
HIZone z(2.,10.,3.,11);
p.addZone(&z);
```

Parameters: `zone` — Pointer to the HIZone to be added.

3.1.2.1.3

SoSeparator* **Draw** (int IX, int IY)

Even if it is declared public, this method is not design to be called by the user program. It is called automatically by the functions doing graphics output like the ShowPage method of the HIWindow classe or the HIPostScript and HIiv functions. This method returns a pointer to Inventor scene graph.

Return Value: PageSceneGraph Pointer to the page scene graph.
Parameters: `IX` — Paper or HIWindow width.
`IY` — Paper or HIWindow height.

3.1.2.1.4

float **XCm2View** (float XCM)

This method convert a X cm value into a X normalized value.

Return Value: Converted value (normalized coordinates).
Parameters: `XCM` — X cm value to be converted.

3.1.2.1.5

float **YCm2View** (float YCM)

This method convert a Y cm value into a Y normalized value.

Return Value: Converted value (normalized coordinates).
Parameters: `YCM` — Y cm value to be converted.

3.1.3

class **HIZone**

Names

This class allows to create and manage HIZones. A HIZone can be placed into a HIPage at creation time by specifying its position coordinates in cm. A HIZone defined a coordinate system which will be used at drawing time to draw the data. A HIZone contains a collection of HIData. The coordinate system is computed by the HIZone in order to contain (see) all the data set known by the zone.

3.1.3.1

Public data members

Names

The appearance of a zone can be controlled via settings and options. Two methods **set** and **option** are provided in the HIZone class to change settings and options.

3.1.3.1.1

enum **zoneSettings**

The possible settings are:

XaxisColor X Axis body color.

XmainGradTickColor X axis main tick marks color.

XsubGradTickColor X axis Secondary tick marks color.

YaxisColor Y Axis body color.

YmainGradTickColor Y axis main tick marks color.

YsubGradTickColor Y axis Secondary tick marks color.

backgroundColor HIZone backgound color.

XGradLabelSize X axis labels size (cm).

YGradLabelSize Y axis labels size (cm).

XGradAddString X axis string added to each label (units).

YGradAddString Y axis string added to each label (units).

XGradDistAxis Distance (cm) between X axis body and labels.

YGradDistAxis Distance (cm) between Y axis body and labels.

XTickMainLength X axis main tick marks length (cm).

YTickMainLength Y axis main tick marks length (cm).

XTickSubLength X axis Secondary tick marks length (cm).

YTickSubLength Y axis Secondary tick marks length (cm).

titleSize HIZone title size (cm).

titleDist Distance (cm) between X axis body and the HIZone title.

dataMaximum Scale factor apply to the maximum of the data [0,1].

dataMinimum Scale factor apply to the minimum of the data [0,1].

statsXPos X statistics position.

statsYPos Y statistics position.

statsSize statistics text size.

XAxisTitle X axis title.

YAxisTitle Y axis title.

XAxisTitleSize X axis title size (cm).

XAxisTitleDist Distance from the X axis title to the X axis (cm).

XAxisPos X axis position between 0 and 1.

YAxisTitleSize Y axis title size (cm).

YAxisTitleDist Distance from the Y axis title to the Y axis (cm).

YAxisPos Y axis position between 0 and 1.

3.1.3.1.2

enum **zoneOptions**

The possible options are:

XAxis The X axis will be drawn.

noXAxis The X axis will not be drawn.

XAxisGrid The X axis grid will be drawn.

noXAxisGrid The X axis grid will not be drawn.

noXAxisArrow No arrow at the en of X axis.

XAxisArrow An arrow is drawn at the end of the X axis.

XAxisLinear X axis is linear.

XAxisLog X axis is logarithmic.

YAxis The Y axis will be drawn.

noYAxis The Y axis will not be drawn.

YAxisGrid The Y axis grid will be drawn.

noYAxisGrid The Y axis grid will not be drawn.

noYAxisArrow No arrow at the en of Y axis.

YAxisArrow An arrow is drawn at the end of the Y axis.

YAxisLinear Y axis is linear.

YAxisLog Y axis is logarithmic.

Stats Statistics are drawn.

noStats Statistics are not drawn.

3.1.3.1.3

enum **AxisType**

Axis type. Can be LINEAR or LOGARITHMIC.

3.1.3.1.4

short **XAxisType**

Current X axis type. Can be LINEAR or LOGARITHMIC.

3.1.3.1.5

short **YAxisType**

Current Y axis type. Can be LINEAR or LOGARITHMIC.

3.1.3.2

Public methods

Names

3.1.3.2.1

HIZone (float X1P=2., float X2P=18., float Y1P=2., float Y2P=18., float X1A=0., float X2A=0., float Y1A=0., float Y2A=0., float Z1A=0., float Z2A=0.)

Constructor. It takes ten arguments. The first for arguments are the zone position (cm) on the page. If not specified the position (cm) is (2.,18.,2.,18). The other arguments speficy the zone coordinates. Most of the time these values should not be specified because by default they are computed from the data sets included in the Zone. Example:

```
HIZone z1(2.,10.,3.,11);
```

creates a zone at the position (cm) (2.,10.) on X and (3.,11.) on Y.

Parameters:
`X1P` — X lower zone position value.
`X2P` — X higher zone position value.
`Y1P` — Y lower zone position value.
`Y2P` — Y higher zone position value.
`X1A` — X lower zone coordinates value.
`X1A` — X higher zone coordinates value.
`Y1A` — Y lower zone coordinates value.
`Y2A` — Y higher zone coordinates value.
`Z1A` — Z lower zone coordinates value.
`Z2A` — Z higher zone coordinates value.

3.1.3.2.2

int **AddData** (HIData *data)

This method allows to add a data set in the zone. It takes one argument: a pointer to the data set to be added. A `HIZone` can contains many data sets. A data set can be added in several zones. The following example add the data set `d` in the `HIZone z`:

```
HIZone z(2.,10.,3.,11);
HIData d(h);
z.AddData(&d);
```

h is a pointer to a Histo1D;

Return Value: Data identifier in the Zone.
Parameters: `data` — Pointer to the data set to be added.

3.1.3.2.3

void **AddComment** (HIComment *comment)

This method allows to add a comment in the zone. It takes one argument: a pointer to the comment to be added. A `HIZone` can contains many comments. A comment can be added in several zones. The following example add the comment `c` in the `HIZone z`:

```
HIZone z(2.,10.,3.,11);
HIComment c(10.,10.,"Hello");
z.AddComment(&c);
```

Parameters: `comment` — Pointer to the comment to be added.

3.1.3.2.4

void **AddSceneGraph** (SoSeparator *sceneGraph)

This method allows to add a SceneGraph in the zone. It takes one argument: a pointer to the SceneGraph to be added. A `HIZone` can contains many SceneGraphs. A SceneGraph can be added in several zones. The following example add the SceneGraph `sg` in the `HIZone z`:

```
HIZone z(2.,10.,3.,11);
z.AddSceneGraph(&sg);
```

Parameters: `comment` — Pointer to the comment to be added.

3.1.3.2.5

void **DeleteData** (int Id)

This method allows to delete a data set in the zone. It takes one argument: a Data identifier in the Zone.

Parameters: `Id` — identifier in the Zone.

3.1.3.2.6

void **resetSettings** (void)

Reset the default HIZone settings.

XGradLabelSize = 0.32

YGradLabelSize = 0.32

XGradDistAxis = 0.40

YGradDistAxis = 0.20

XTickMainLength = 0.30

YTickMainLength = 0.30

XTickSubLength = 0.15

YTickSubLength = 0.15

titleSize = 0.35

titleDist = 1.20

Colors = "Black"

XGradAddString = " "

YGradAddString = " "

XAxisTitle = " "

YAxisTitle = " "

statsXPos = 0.15

statsYPos = 0.15

XAxisTitleSize = 0.28

XAxisTitleDist = 0.80

YAxisTitleSize = 0.28

YAxisTitleDist = 1.40

3.1.3.2.7

void **set** (zoneSettings S, char *C)

This method allows to modify the HIZone attributes which are of type `char`. Example:

```
z.set(HIZone::backgroundColor,Red);
```

Note that the `zoneSettings` enum is part of the class `HIZone` so to access the members of the enum the prefix `HIZone::` should be used.

Parameters: `S` — Value to be change.
`C` — New value.

3.1.3.2.8

void **set** (zoneSettings S, float F)

This method allows to modify the HIZone attributes which are of type `float`. Example:

```
z.set(HIZone::titleSize,0.5);
```

Note that the `zoneSettings` enum is part of the class `HIZone` so to access the members of the enum the prefix `HIZone::` should be used.

Parameters: `S` — Value to be change.
`F` — New value.

3.1.3.2.9

void **resetOptions** (void)

Reset the default HIZone settings.

X Axis = Yes;

Y Axis = Yes;

X Axis Arrow = No;

Y Axis Arrow = No;

X Axis Type = LINEAR;

Y Axis Type = LINEAR;

X Axis Grid = No;

Y Axis Grid = No;

3.1.3.2.10

void **option** (zoneOptions O)

This method allows to modify the HIZone options. Example:

```
z.option(HIZone::YAxisLog);
```

Note that the `zoneOptions` enum is part of the class `HIZone` so to access the members of the enum the prefix `HIZone::` should be used.

Parameters: `O` — Option to be set on.

3.1.3.2.11

void **setTitle** (char *T)

Set the HIZone set title. The HIZone title is automatically set equal to the title of the first data set added in the HIZone.

Parameters: `T` — New title.

3.1.3.2.12

void **setAngles** (float AX=30., float AY=0., float AZ=30.)

For 3D representation. Set the rotations angles around the X, Y and Z axis.

Parameters:
`AX` — Rotation angle around the X axis;
`AY` — Rotation angle around the Y axis;
`AZ` — Rotation angle around the Z axis;

3.1.3.2.13

void **GetZonePosition** (float , float , float , float)

Return the zone position in the page.

3.1.3.2.14

void **GetZoneLimits** (float , float , float , float)

Return the zone limits.

3.1.3.2.15

void **SetZonePosition** (float , float , float , float)

Set the zone position.

3.1.3.2.16

int **isValid** ()

TRUE if the zone position and limits are valid.

3.1.3.2.17

SoSeparator* **Draw** (HIPage *page)

Even if it is declared public, this method is not design to be called by the user program. It is called automatically by the `Draw` method of the `HIPage` class. This method returns an Inventor scene graph.

Return Value:	ZoneSceneGraph Pointer to the zone scene graph.
Parameters:	`page` — Pointer to the HIPage in which the HIZone is drawn.

3.1.3.2.18

float **XZone2Cm** (float X)

Convert a X zone coordinate into centimeters.

3.1.3.2.19

float **YZone2Cm** (float Y)

Convert a Y zone coordinate into centimeters.

3.1.3.2.20

float **XCm2Zone** (float X)

Convert a centimeter X coordinate into zone a coordinate.

3.1.3.2.21

float **YCm2Zone** (float Y)

Convert a centimeter Y coordinate into zone a coordinate.

3.1.4

class **HIData**

Names

This class allows to create and manage HIData(s) (vectors or histograms). The HIData class is the interface between the user data and the graphics. User data can be stored in simple floating points array or in histograms. Depending how the user data are stored, one constructor or an other is used. Having this class allows to disconnect the data from the graphics representation. If a new type of data has to be drawn by the HEPInventor package it is enough to create a new constructor which accept this new data type keeping unchanged all the other parts of the package.

3.1.4.1

Public data members

Names

The appearance of a data set can be controlled via data settings and data drawing modes. One method (**set**) is provided with 2 different signatures to change these settings.

3.1.4.1.1

enum **dataSettings**

The possible settings are:

FillColor Color used to fill the curve.

BorderColor Color used to draw the border of the curve.

ErrorBarsColor Color used to draw the error bars.

LineWidth Line width used to draw the curve.

LinePattern Line Pattern used to draw the curve.

3.1.4.1.2

enum **dataDrawingMode1D**

The possible drawing modes for 1D data sets are:

Line Data are drawn as a simple curve (default for arrays).

Smooth Data are drawn as a smooth curve.

Histogram Data are drawn as a histogram (default for Histo1D)

Errors Data are drawn as error bars.

3.1.4.1.3

enum **dataDrawingMode2D**

The possible drawing modes for 2D data sets are:

Box Data are drawn as a set of boxes.

Ellipse Data are drawn as a set of boxes.

Colour Data are mapped on a colour table.

3.1.4.2

Public methods

Names

3.1.4.2.1

HIData (int N, float *X, float *Y)

Constructor from floating point 1D arrays. This constructor allows to create graphics data from user data stored in two floating point arrays. It takes three arguments: the number of points in the arrays and the two arrays X and Y. The following example shows how this constructor should be used:

```
#define N  20
#define PI 3.1415
float X[N],sinX[N];
float angle = 0;
float dx    = 2*PI/N;
int i;
for ( i=0; i < N; i++) {
   X[i]    = angle;
   sinX[i] = sin(angle);
   angle   = angle+dx;
}
HIData SinData(N,X,sinX);
```

Other constuctors using double instead of float are also available:

```
HIData (int N, float *X, double *Y);
HIData (int N, double *X, double *Y);
```

Parameters:
`N` — Number of points in X and Y arrays.
`X` — Data value along the X axis.
`Y` — Data value along the Y axis.

3.1.4.2.2

HIData (int n, float* x, float* y, float *e)

Constructor from floating point 1D arrays. This constructor allows to create graphics data from user data stored in three floating point arrays. It takes four arguments: the number of points in the arrays, the two arrays X and Y and the array of errors along the Y axis. Some constuctors using double are also available:

```
HIData(int n,float* x,double* y,double *e) ;
HIData(int n,double* x,double* y,double* e) ;
```

Parameters:
`N` — Number of points in X and Y arrays.
`X` — Data value along the X axis.
`Y` — Data value along the Y axis.
`E` — Errors along the Y axis.

3.1.4.2.3

HIData (int NX, int NY, float *X, float *Y, double *Z)

Constructor from 2D arrays. This constructor allows to create graphics data from user data stored in floating point arrays. It takes five arguments: the number of points along the X axis, the number of points along the Y axis, two arrays X and Y defining the coordinates along the X and Y axis and finally an array containing the data. Some constuctors using double are also available:

```
HIData (int NX, int NY, float *X, float *Y, float *Z );
HIData (int NX, int NY, double *X, double *Y, float *Z );
HIData (int NX, int NY, double *X, double *Y, double *Z );
```

Parameters:
`NX` — Number of points along X.
`NY` — Number of points along Y.
`X` — Coordinate value along the X axis.
`Y` — Coordinate value along the Y axis.
`Z` — Data value.

```
H_P_REF(P_Histo1D) h1=new(dbH) P_Histo1D("Histo",10,0.,20.);
H_P_REF(P_Histo2D) h2=new(dbH) P_Histo2D("2D",9,0.,9.,9,0.,9.);

                ... histograms filling ...

        HIData d1(*h1);
        HIData d2(*h2);
```

3.1.4.2.5

void **resetSettings** (void)

Reset the default settings for a data set.

- Drawing mode = "Line"
- Fill color = "Black"
- Border color = "Black"
- Line Width = "Black"

3.1.4.2.6

void **set** (dataSettings S, char *C)

This method allows to modify the attributes used to draw a data set. For example it is possible to change the color used to fill the curve:

```
d.set(HIData::FillColor,Blue);
```

The user can also specify his own color:

```
#define C1   "diffuseColor 0.8 1. 0.8"
d.set(HIData::FillColor,C1);
```

Note that the `dataSettings` enum is part of the class `HIData` so to access the members of the enum the prefix `HIData::` should be used.

Parameters: `S` — Value to be change.
`C` — New value.

3.1.4.2.7

void **set** (int D)

This method allows to set the "HIData drawing mode". One data set can have several drawing modes. In the following example, the data set `d` will be drawn as a line and as error bars.

```
d.set(HIData::Errors | HIData::Line );
```

Note that the `dataDrawingMode` enum is part of the class `HIData` so to access the members of the enum the prefix `HIData::` should be used.

Parameters: `D` — Drawing mode(s) to be used.

3.1.4.2.8

void **set** (HIPalette *P)

This method allows to set the colour palette to be used (for example by the option Colour.

Parameters: `P` — Palette to be used.

3.1.4.2.9

void **setMinMax** (float MIN, float MAX)

Set the HIData minimum and maximum.

Parameters: `MIN` — Minimum.
`MAX` — Maximum.

3.1.4.2.10

void **computeMinMax** ()

Compute the HIData minimum and maximum.

3.1.4.2.11

void **setTitle** (char *T)

Set the HIData set title. For data coming from and Histo1D/Histo1DVar, the title is set by default to the histogram title.

Parameters: `T` — New title.

3.1.4.2.12

char* **getTitle** ()

Returns the data set title. This method is used by the `HIZone` class.

Return Value: title;

3.1.4.2.13

void **getDataLimits** (float , float , float , float)

Return the limits of the data set. The method returns four float Xmin, Xmax, Ymin and Ymax.

3.1.4.2.14

void **Draw** (HIZone *zone, PoDomain *Datadomain, SoSeparator *DataSceneGraph)

Even if it is declared public, this method is not design to be called by the user program. It is called automatically by the `Draw` method of the `HIZone` class. This method creates an Inventor scene graph and attach it on the `DataSceneGraph` pointer.

Parameters:
`zone` — Pointer to the HIZone in which the data set is drawn.
`Datadomain` — Data domain (define the data set coordinates).
`PageSceneGraph` — Pointer to the data scene graph.

3.1.4.2.15

void **addRef** ()

This method is used by the `HIZone` to increment the reference counter.

3.1.4.2.16

void **delRef** ()

This method is used by the `HIZone` to decrement the reference counter.

3.1.4.2.17

int **getRef** ()

This method is used by the `HIZone` to get the reference counter value.

3.1.5

class **HIComment**

Names

This class allows to create a comment (some text) which can be then put into a HIZone at the position specified at creation time. The position can be define in the HIZone coordinates or in centimeters (default is centimeters). The size of the comment is also specified at creation time and is always in centimeters.

3.1.5.1

Public data members

Names

A comment can be drawwn in the centimeters of zone coordinates. This is controlled wia the `option` method and the `PosCM` and `PosZONE` option in the `commentOptions` enum.

3.1.5.1.1

enum **commentOptions**

The possible options are:

PosCM The comment position is defined in centimeters.

PosZONE The comment position is defined in zone coordinates.

3.1.5.2

Public methods

Names

3.1.5.2.1

HIComment (float x, float y, char *string, float height=0.5)

Constructor. Create a comment which can be then put in a `HIZone`:

```
HIComment *com = new HIComment(0.,18.,"Hello");
        com->option(HIComment::PosZONE);
```

Parameters: `x` — X position of the text to be drawn.
`y` — Y position of the text to be drawn.
`string` — Text to be drawn.
`height` — Height of the text to be drawn.

3.1.5.2.2

void **option** (commentOptions option)

This method allows to modify the HIComment options. Example:

```
c.option(HIComment::PosZONE);
```

Note that the `commentOptions` enum is part of the class `HIComment` so to access the members of the enum the prefix `HIComment::` should be used.

Parameters: `option` — Option to be set on.

3.1.5.2.3

void **Draw** (PoDomain *domain, SoSeparator *separator, HIZone *zone, HIPage *page)

Even if it is declared public, this method is not design to be called by the user program. It is called automatically by the `Draw` method of the `HIZone` class. This method creates an Inventor scene graph and attach it on the `SoSeparator` given as input parameter.

Parameters: `domain` — Domain.
`separator` — Scene graph.
`zone` — HIZone in which the comment is drawn.
`page` — HIPage containing the zone.

3.1.6 class **HIPageManager**

This class allows to manage several zone inside a page.

```
#include "HInventor.h"
int main()
{
 int Nx = 2;
 int Ny = 3;
 HIZone *Zone;
 HIPageManager PM(Nx,Ny);

 float X[10]  = { 0., 1., 2., 3., 4., 5., 6., 7., 8., 9.} ;
 float Y[10]  = { 1., 2., 3., 4., 5., 5., 4., 3., 2., 1.} ;

 HIData DATA(10,X,Y);

 for (int i=1; i <= Nx*Ny; i++) {
    Zone = PM.GetNextZone();
    Zone->AddData(&DATA);
 }

 HIPostScript PS("Ex009.ps");   //
 PS.put(PM.Page);               // generate the PS file Ex009.ps from PAGE
 PS.close();                    //
 return 0;
}
```

3.1.7 class **HIPalette**

Names

This class allows to define color palette. Palettes are used to draw 2D data set with the color option.

3.1.7.1

Public methods

Names

The public methods of the class HIPalette allow to:

- create a HIPalette:

```
HIPalette *P;
P = new HIPalette (8);
```

- retrieve the content of the HIPalette:

```
SbColor C = P->getColour(i);
```

3.1.7.1.1

HIPalette (int NC)

HIPalette constructor. It takes one argument.

Parameters: `NC` — Number of colors in the palette.

3.1.7.1.2

void **setColour** (int I, float R, float G, float B)

Set the color number "I". It takes four arguments.

Parameters: `I` — Color index in the palette.
`R` — Red value.
`G` — Green value.
`B` — Blue value.

3.1.7.1.3

SbColor **getColour** (int I)

Return the color number "I". It takes one argument.

Return Value: Colour number I
Parameters: `I` — Color index in the palette.

3.1.7.1.4

void **getRGB** (int I, float R, float G, float B)

Return R G B components of the color number "I". It takes 4 argument.

Parameters: `I` — Color index in the palette.
`R` — Red.
`G` — Green.
`B` — Blue.

3.1.8

class **HIPostScript**

Names

This class allows to translate scenegraphs into a PostScript files. The constructor open a new PostScript file. The put methods allow to put several scenegraphs or HIPages in the same PostScript file. The close method finishes the PostScript output in a file.

3.1.8.1 Public methods

Names

3.1.8.1.1 HIPostScript (char *filename, int xstart=5, int ystart=30)

Constructor. Open a new PostScript file.

Parameters: `FileName` — X position of the statistics.
`xstart` — X starting position.
`ystart` — Y starting position.

3.1.8.1.2

void **put** (SoSeparator *S)

This method put a SoSeparator into a PostScript file.

Parameters: `S` — SceneGraph to be converted in PostScript.

3.1.8.1.3

void **put** (HIPage *P)

This method put a HIPage into a PostScript file.

Parameters: `P` — HIPage to be converted in PostScript.

3.1.8.1.4

void **close** ()

This method closes the PostScript file.

3.2 Internal utility classes.

Names

Classes used by the user classes. Normally should not be used from the user program.

3.2.1

class **HIDataItem**

This class encapsulate a HIData object to be managed in a list. This utility class is used by the HIZone class. HIZone is a "friend class" of HIDataItem.

3.2.2

class **HIZoneItem**

This class encapsulate a HIZone object to be managed in a list. This utility class is used by the Page class. Page is a "friend class" of HIZoneItem.

3.2.3

class **HICommentItem**

This class encapsulate a HIComment object to be managed in a list. This utility class is used by the HIZone class. HIZone is a "friend class" of HICommentItem.

3.3

User functions

Names

These functions are callable from a user program. They are not object methods but provide some general funtionnalities.

3.3.1

void **HIiv** (HIPage *page, char *filename)

Generate an IV file. It takes two arguments: a pointer to a HIPage and a file name. Example:

```
#include "HInventor.h"
int main()
{
   HIPage p;
   HIZone z1(2.,10.,3.,11);
   p.AddZone(&z1);
   HIiv(&p, "example.iv");
   return 0;
}
```

This example generate the Inventor file called `example.iv`.

Parameters: `page` — Pointer to a HIPage.
`filename` — Inventor file name.

3.4

Internal utility functions.

Names

These functions are not callable from a user program. They are used internaly in the package. They are not object methods but provide some general funtionnalities needed by the package.

3.4.1

void **HIInit** (int)

Initialize HEP Inventor. This function is automatically called when needed. Returns 0 if the initialization was valid, 1 otherwise.

3.4.2

PoCurve* **HIDrawLineBox** (PoDomain *, float, float, float, float, char *)

Draw a box with line. It takes six arguments: a pointer to a Domain, four floating points for the box position and one character string for the box color. It returns a PoCurve.

3.4.3

PoRectangle* **HIDrawFilledBox** (PoDomain *, float, float, float, float, char *)

Draw a box with filled polygon. It takes six arguments: a pointer to a Domain, four floating points for the boxe position and one character string for the box color. It returns a PoRectangle.

3.4.4

PoCurve* **HIDrawLineEllipse** (PoDomain *, float, float, float, float, char*)

Draw an ellipse. It takes six arguments: a pointer to a Domain, two floating points for the ellipse centre, one float for the X radius, one flot for the Y radius and one character string for the color. It returns a PoCurve.

3.4.5

PoCurve* **HIDrawFilledEllipse** (PoDomain *, float, float, float, float, char*)

Draw an ellipse. It takes six arguments: a pointer to a Domain, two floating points for the ellipse centre, one float for the X radius, one flot for the Y radius and one character string for the color. It returns a PoCurve.

3.4.6

PoSceneView* **HIDefineView** (float, float, float, float, HIPage *, SoSeparator *)

Define a view in cm. Returns a PoSceneView.

3.4.7

PoDomain* **HIDefineDomain** (float, float, float, float, float, float, float, float)

Define a domain according to its position in cm and its axis coordinates Returns a PoDomain.

3.5 Iris Explorer utility functions.

Names

These functions are not callable from a user program. They allow to send MasterSuite geometry to the Iris Explorer Render module.

3.5.1

int **HIEOpenGeometry** (char *portname, cxGeometry **geo, int *port)

Open the named geometry buffer and start the geometry If unsuccessful storage is reclaimed and 1 is returned, else 0

3.5.2

void **HIECloseGeometry** (cxGeometry *geo, int port)

Close the named geometry buffer and send the geometry

3.6 Iris Explorer utility functions.

Names

These functions are callable from a user program (module). They allow to send a HEPInventor HIPage the Iris Explorer Render module.

3.6.1 void **HIEInit** (void)

Initialise the package.

3.6.2 void **HIEDisplayPage** (HIPage *page)

Send a HIPage to the Render window

Class Graph