

# Modifying All-Silicon Tracker Prototype in EICroot

Reynier Cruz Torres

July 17, 2020

## Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Accessing EICroot image</b>                     | <b>2</b>  |
| <b>2</b> | <b>Modifying the geometry</b>                      | <b>2</b>  |
| 2.1      | Beampipe . . . . .                                 | 2         |
| 2.2      | Barrel . . . . .                                   | 3         |
| 2.3      | Disks . . . . .                                    | 4         |
| 2.4      | Support structure . . . . .                        | 5         |
| <b>3</b> | <b>Compiling the geometry into a TGeo file</b>     | <b>5</b>  |
| <b>4</b> | <b>Visualizing the geometry</b>                    | <b>6</b>  |
| <b>5</b> | <b>Detector updates (week of July 13-17, 2020)</b> | <b>7</b>  |
| 5.1      | Beampipe updates . . . . .                         | 7         |
| 5.2      | Barrel updates . . . . .                           | 7         |
| 5.3      | Disk updates (Under Construction) . . . . .        | 8         |
| <b>A</b> | <b>Appendices</b>                                  | <b>10</b> |
| A.1      | Original vst.C macro from Yue Shi Lai . . . . .    | 10        |
| A.2      | Original fbst.C file from Yue Shi Lai . . . . .    | 12        |
| A.3      | Help menu from event display GUI . . . . .         | 13        |

# 1 Accessing EICroot image

EICroot and necessary dependencies are already installed in a virtual machine. Install VMware. Double click on the file `vm-eic.vmx`. When the image loads, it will ask for a username (eic) and password (eiceic). This virtual machine was created with the Beast detector geometry. Thus, two files (`vst.C` and `fbst.C`) need to be updated so that the code generates the All-Silicon tracker geometry instead. The original codes I got from Yue Shi Lai are appended at the end of this document.

## 2 Modifying the geometry

The All-Si tracker geometry is defined in: `Home/eicroot/geometry/`.

### 2.1 Beampipe

The beam-pipe geometry is defined in:

```
Home/eicroot/geometry/BEAMPIPE/beampipe.C
```

The central part of the beampipe is made of Beryllium and is  $800\text{ }\mu\text{m}$  thick. It is 36 mm in diameter, and extends from -400 to 400 mm:

```
BeamPipeElement *ip = new BeamPipeElement("beryllium", 0.800);
ip->AddSection(-400.0, 36.00);
ip->AddSection( 400.0, 36.00);
```

Outside of this range, the beampipe becomes Aluminum and extends from where the Beryllium section ends and up until  $\pm 4.5\text{ m}$ . it increases in diameter from 36 mm to 40 mm:

```
BeamPipeElement *p1 = new BeamPipeElement("aluminum", 1.000);
p1->AddSection( 400.0, 36.00);
p1->AddSection(1000.0, 40.00);
p1->AddSection(4500.0, 40.00);
```

This is how the geometry is defined. Then it is assembled as follows:

```
BeamPipeGeoParData *bpipe = new BeamPipeGeoParData();
bpipe->AddElement (p1, BeamPipeElement::Swap);
bpipe->AddIpElement(ip);
bpipe->AddElement (p1);
```

Fig. 1 shows a diagram for the beampipe.

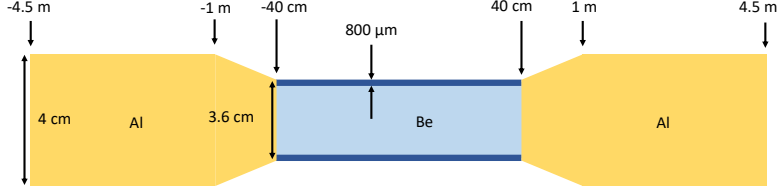


Figure 1: Beampipe schematic (not in scale).

## 2.2 Barrel

The barrel details are in:

```
Home/eicroot/geometry/MAPS/vst.C
```

The first step is to define the common cell that will be used to build up the barrel:

```
MapsMimosaAssembly *ibcell = (MapsMimosaAssembly*)ConfigureAliceCell();
```

Then, six layers are assembled, as illustrated below. The variables passed in `AddBarrelLayer` below correspond to: cell assembly type, number of staves in this layer, number of chips in a stave, chip center installation radius, additional stave slope around beam line direction in degrees, layer rotation around beam axis “as a whole” in degrees:

```
vst->AddBarrelLayer(ibcell, 12, 9, 23.4, 12.0, 0.0);
vst->AddBarrelLayer(ibcell, 2*12, 9, 2*23.4, 12.0, 0.0);
vst->AddBarrelLayer(ibcell, 6*12, 14, 6*23.4, 14.0, 0.0);
vst->AddBarrelLayer(ibcell, 4*20, 14, 4*39.3, 14.0, 0.0);
vst->AddBarrelLayer(ibcell, 4*20*10/4, 14*10/4, 10*39.3, 14.0, 0.0);
vst->AddBarrelLayer(ibcell, 4*20*11/4, 14*11/4, 11*39.3, 14.0, 0.0);
```

The information in this snippet of code is presented in Table 1. An overall overview of the barrel geometry is shown in Fig. 2 a).

Table 1: Barrel characteristics

| Layer | # staves<br>in layer | # chips<br>in stave | length along<br>z (cm) | chip center<br>installation radius (mm) | stave slope<br>(deg) |
|-------|----------------------|---------------------|------------------------|---|----------------------|
| 1     | 12                   | 9                   | 27                     | 23.4                                    | 12                   |
| 2     | 24                   | 9                   | 27                     | 46.8                                    | 12                   |
| 3     | 72                   | 14                  | 42                     | 140.4                                   | 14                   |
| 4     | 80                   | 14                  | 42                     | 157.2                                   | 14                   |
| 5     | 200                  | 35                  | 105                    | 393                                     | 14                   |
| 6     | 220                  | 38                  | 114                    | 432.3                                   | 14                   |

Here, a “chip” refers to a Mimosa chip assembly, and is the basic building block for the detector. The chip characteristics are defined in:

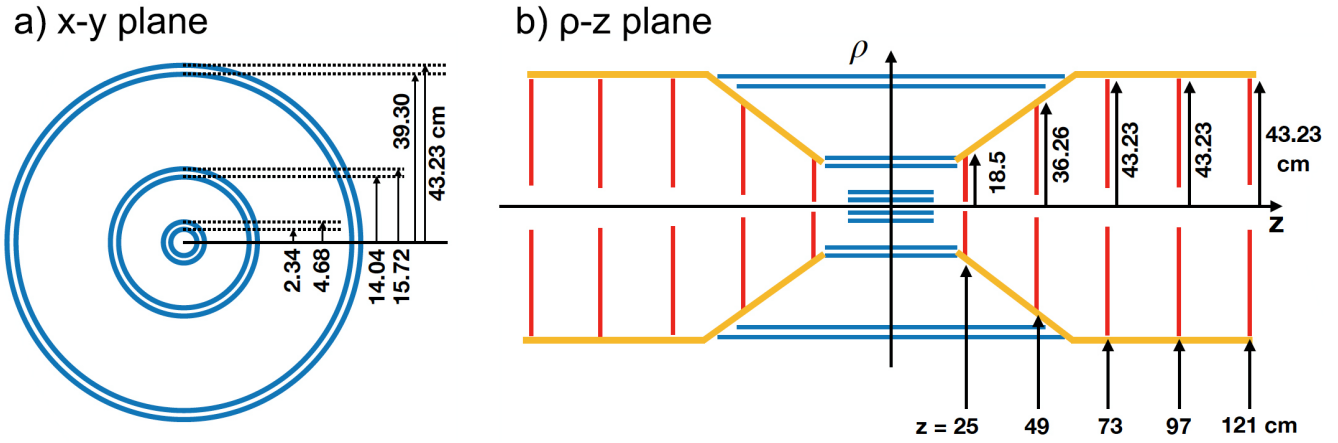


Figure 2: Schematic of the detector geometry. The barrel, disks, and support structure are drawn in blue, red, and yellow respectively. a) barrel as seen in the  $x - y$  plane. b) detector as seen in the  $\rho - z$  plane. This is a simplified diagram, as the barrel layers and disks are not smooth. The barrel is formed by staves, and the disks have jagged edges.

```
Home/eicroot/geometry/MAPS/maps-lib.C
```

The chips are defined with a width of 15 mm and a length (in  $z$ ) of 30 mm. Consequently, the length of the barrel layers in  $z$  is given by the number of chips per stave and corresponds to  $(\# \text{ chips}) \times (\text{chip length})$ .

## 2.3 Disks

The disks are in:

```
Home/eicroot/geometry/MAPS/fbst.C
```

At the top of this code we define the number of disks we will have in each  $z$  direction:

```
#define _DISK_NUM_ (5)
```

Then, the positions in  $z$  and radii of these disks are determined as follows:

```
Double_t Z[_DISK_NUM_];
Double_t R[_DISK_NUM_];
for (size_t i = 0; i < _DISK_NUM_; i++) {
    Z[i] = 250 + i * (1210.0 - 250.0) / (_DISK_NUM_ - 1);
    R[i] = 45.0 * (i % 2);
}
```

This corresponds to the values shown in Table 2 and in Fig. 2 b).

Table 2: Disk characteristics (in the forward direction). To get backward disks multiply the z-position column by -1.

| Disk number | z position [cm] | outer radius [cm] |
|-------------|-----------------|-------------------|
| 1           | 25              | 18.5              |
| 2           | 49              | 36.26             |
| 3           | 73              | 43.23             |
| 4           | 97              | 43.23             |
| 5           | 121             | 43.23             |

The disks are added the following way:

```
for(unsigned dc=0; dc<_DISK_NUM_; dc++) {
    FstDisc *disc = new FstDisc(ibcell, 18.0, TMath::Min(432.3, 185.0 * Z[dc] / Z[0]), 12.8);
    fbst->AddDisc(disc, (fb ? -1.0 : 1.0)*Z[dc], R[dc]);
}
```

## 2.4 Support structure

Note: EICroot uses millimeters as the standard length unit. All the lengths shown up to this point corresponded to mm. However, the support structure is defined directly using Geant classes, which use cm as the standard length unit. Thus, the lengths in this section are given in cm.

The support structure is also defined in:

```
Home/eicroot/geometry/MAPS/fbst.C
```

The parameters that go into `DefineSection` are: a numeric tag or label, z position, inner radius, and outer radius. The support structure is defined with three sections.

```
TGeoPcon *support = new TGeoPcon("AluStrips_support", 0.0, 360.0, 3);
support->DefineSection(0, (fb ? 1.0 : -1.0) * 20, 18.5 * 20.0 / 25.0, 18.5 * 20.0 / 25.0 + 0.5);
support->DefineSection(1, (fb ? 1.0 : -1.0) * 43.23 * 0.1 * Z[0] / 18.5, 43.23, 43.23 + 0.5);
support->DefineSection(2, (fb ? 1.0 : -1.0) * 121, 18.5 * 121.0 / 25.0, 18.5 * 121.0 / 25.0 + 0.5);
TGeoVolume *support_volume = new TGeoVolume("AluStrips_support_volume", support, fbst->GetMedium("aluminum"));
fbst->GetTopVolume()->AddNode(support_volume, 0, new TGeoCombiTrans(0.0, 0.0, 0.0, 0));
```

## 3 Compiling the geometry into a TGeo file

First of all, run the macros mentioned above. Go to the corresponding directories and run:

- `root -l beampipe.C`
- `root -l vst.C`
- `root -l fbst.C`

This will create root files that contain these pieces of the geometry to be stored in the combined TGeo file. Next, edit the function `void EicDetector::FinishRun()` in code:

```
Home/eicroot/eic/base/EicDetector.cxx
```

to include the following three lines (add as the first three lines of that function):

```
TFile *outfile = TFile::Open("genfitGeom.root","RECREATE");
gGeoManager->Write();
outfile->Close();
```

Afterwards, you have to recompile by `make -C build` or `cd Home/eicroot/eic/build; make`. Run the following root macro to compile the geometry:

```
{
    // Load basic libraries;
    gROOT->Macro("$VMCWORKDIR/gconfig/rootlogon.C");
    // Create simulation run manager; use GEANT3 for tracking exercises here;
    EicRunSim *fRun = new EicRunSim("TGeant3");
    fRun->SetOutputFile("simulation-uds.root");
    // Vertex tracker;
    fRun->AddModule(new EicMaps   ("VST", "/home/eic/eicroot/geometry/MAPS/vst-v01.0-fs.root", qVST));
    fRun->AddModule(new EicMaps   ("FST", "/home/eic/eicroot/geometry/MAPS/fst-v01.0-ns.root", qFST));
    fRun->AddModule(new EicMaps   ("BST", "/home/eic/eicroot/geometry/MAPS/bst-v01.0-ns.root", qBST));
    // Beam pipe; details of IR not known yet -> just a cylinder with thin walls;
    fRun->AddModule(new EicDummyDetector("BEAMPIPE", "BEAMPIPE/beampipe.root"));
    // Initialize and run the simulation; exit at the end;
    fRun->Run(-1);
}
```

## 4 Visualizing the geometry

```
{
    // Load basic libraries;
    gROOT->Macro("$VMCWORKDIR/gconfig/rootlogon.C");
    // Create simulation run manager; use GEANT3 for tracking exercises here;
    EicRunSim *fRun = new EicRunSim("TGeant3");
    fRun->SetOutputFile("simulation-uds.root");
    // Vertex tracker;
    fRun->AddModule(new EicMaps   ("VST", "/home/eic/eicroot/geometry/MAPS/vst-v02.0-fs.root", qVST));
    fRun->AddModule(new EicMaps   ("FST", "/home/eic/eicroot/geometry/MAPS/fst-v02.0-ns.root", qFST));
    fRun->AddModule(new EicMaps   ("BST", "/home/eic/eicroot/geometry/MAPS/bst-v02.0-ns.root", qBST));
    // Beam pipe; details of IR not known yet -> just a cylinder with thin walls;
    fRun->AddModule(new EicDummyDetector("BEAMPIPE", "/home/eic/eicroot/geometry/BEAMPIPE/beampipe.root"));
    // Initialize and run the simulation; exit at the end;
    fRun->Run(1);
}
```

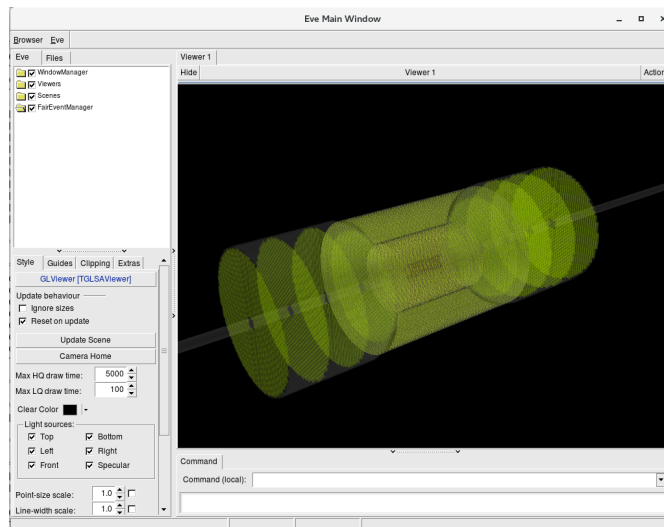


Figure 3: Event-display GUI.

```
{
  gROOT->Macro("$VMCWORKDIR/gconfig/rootlogon.C"); // Load basic libraries;
  // Create visualization manager;
  EicEventManager *fMan = new EicEventManager();
  fMan->SetInputFile("simulation-uds.root");
  fMan->Init(); // Initialize and run visualization manager; tune vis.properties a bit;
  fMan->Run();
}
```

## 5 Detector updates (week of July 13-17, 2020)

### 5.1 Beampipe updates

```
BeamPipeElement *ip = new BeamPipeElement("beryllium", 0.762);
ip->AddSection(-798.0, 62.00);
ip->AddSection( 798.0, 62.00);
BeamPipeElement *pf = new BeamPipeElement("aluminum", 1.000);
pf->AddSection( 798.0, 62.00);
pf->AddSection( 4500.0, 62.00);
bpipe->AddElement (pf, BeamPipeElement::Swap);
bpipe->AddIpElement(ip);
bpipe->AddElement (pf);
```

### 5.2 Barrel updates

Below are two possible upgrades for the barrel. Tho innermost layers are identical in these two upgrades, and differ from the original configuration in that their radii and number of staves have been increased to accommodate the beampipe.

Upgrade 1:

This version has the two middle and two outer layers configured such that the triangular staves are back-to-back. See Fig. 4 (center).

```
vst->AddBarrellLayer(ibcell, 17, 10, 32.000, 12.0, 0.0 );
vst->AddBarrellLayer(ibcell,2*13, 10, 40.000, 12.0, 0.0 );
vst->AddBarrellLayer(ibcell, 82, 20, 224.700, 0.5, 0.0 );
vst->AddBarrellLayer(ibcell, 82, 20, 228.980, 179.5, 180/82 );
vst->AddBarrellLayer(ibcell, 151, 14*10/4, 413.985, 0.5, 0.0);
vst->AddBarrellLayer(ibcell, 151, 14*10/4, 419.095, 179.5, 180/151+0.2);
```

Upgrade 2:

This version has the two middle and two outer layers configured such that the silicon part of each stave faces one another. See Fig. 4 (right). This makes the detector more hermetic.

```
vst->AddBarrellLayer(ibcell, 17, 10, 32.000, 12.0, 0.0 );
vst->AddBarrellLayer(ibcell,2*13, 10, 40.000, 12.0, 0.0 );
vst->AddBarrellLayer(ibcell, 82, 20, 224.700, 0.5, 0.0 );
vst->AddBarrellLayer(ibcell, 82, 20, 224.200, 179.5, 180/82 );
vst->AddBarrellLayer(ibcell, 151, 14*10/4, 419.510, 0.5, 0.0);
vst->AddBarrellLayer(ibcell, 151, 14*10/4, 419.095, 179.5, 180/151+0.2);
```

Additionally, with respect to the original barrel, some of the length of the staves in z has been changed to reflect the changes in the rest of the geometry.



Figure 4: Different configurations for the barrel layers: (left) original, (center) upgrade 1, and (right) upgrade 2.

### 5.3 Disk updates (Under Construction)

In order to avoid these gaps, a “hack” was implemented. The mimosa chips were extended in width.





# A Appendices

## A.1 Original vst.C macro from Yue Shi Lai

```
#define _VERSION_ 1
#define _SUBVERSION_ 0
//define _TEST_VERSION_ // Do not want to always overwrite "official" files; place "test" tag into the file name;
// All construction elements are smeared (so chip assembly is uniform in both beam line and asimuthal direction);
//define _NO_STRUCTURE_GEOMETRY_
// No tricky elements like inclined roof beams; still water pipes, etc are
// created (so chip assembly is uniform in beam line direction only);
//define _BEAM_LINE_UNIFORM_GEOMETRY_
#define _USE_TRIANGULAR_ASSEMBLIES_ // In case of VST no need to fall back to tricky TGeoCompositeShape cell assemblies;
//define _WITH_MOUNTING_RINGS_ // Comment out if no mounting rings wanted;
//define _WITH_ENFORCEMENT_BRACKETS_ // Comment out if no stave enforcement brackets wanted;
//define _WITH_EXTERNAL_PIPES_ // Comment out if no external water pipe pieces wanted;
#include <geometry/MAPS/maps-lib.C>
vst()
{
    // Load basic libraries;
    gROOT->Macro("$VMCWORKDIR/gconfig/rootlogon.C");
    // Prefer to think in [mm] and convert to [cm] when calling ROOT shape
    // definition routines only;
    VstGeoParData *vst = new VstGeoParData(_VERSION_, _SUBVERSION_);
    // Parse #define statements and make certain configuration calls accordingly;
    // NB: cast to the base MAPS geo class since void* interface used; FIXME!;
    DefinitionParser((MapsGeoParData*)vst);
    // For now assume ALICE Inner Barrel design only (but with more than 9 chips); no problem to introduce similar design with a bit different
    // parameter set later; construction a-la ALICE Outer Barrel looks like an overkill to the moment;
    // No offset per default;
    //vst->SetTopVolumeTransformation(new TGeoTranslation(0.0, 0.0, 0.0));
    // NB: all these numbers are sort of arbitrary; allow to get better visual representation, but do not reflect any reality of the final design;
    // Mounting ring construction; arbitrary numbers, same for all layers;
    if (vst->WithMountingRings()) {
        vst->mMountingRingBeamLineThickness = 5.00;
        vst->mMountingRingRadialThickness = 5.00;
        vst->mMountingRingRadialOffset = 3.00;
    } //if
    // Simplify the design -> just a triangular piece with a reasonable volume;
    if (vst->WithEnforcementBrackets())
        vst->mEnforcementBracketThickness = 1.00;
    // This is something for display purposes mostly;
    if (vst->WithExternalPipes())
        vst->mWaterPipeExtensionLength = 4.00;
    MapsMimosaAssembly *ibcell = (MapsMimosaAssembly*)ConfigureAliceCell();
    // If (much) longer staves needed, create an enforced configuration;
    //MapsMimosaAssembly *obcell = new MapsMimosaAssembly(ibcell);
    //obcell->mEnforcementBeamDiameter = 1.00; ... and so on ...
    // Now when basic building blocks are created, compose barrel layers;
    // a dirty part, but perhaps the easiest (and most readable) to do; parameters are:
    // - cell assembly type;
    // - number of stoves in this layer;
    // - number of chips in a stove;
    // - chip center installation radius;
    // - additional stove slope around beam line direction; [degree];
    // - layer rotation around beam axis "as a whole"; [degree];
    //
    vst->AddBarrelLayer(ibcell, 12, 9, 23.4, 12.0, 0.0);
    vst->AddBarrelLayer(ibcell, 2*12, 9, 2*23.4, 12.0, 0.0);
    vst->AddBarrelLayer(ibcell, 6*12, 14, 6*23.4, 14.0, 0.0);
    vst->AddBarrelLayer(ibcell, 4*20, 14, 4*39.3, 14.0, 0.0);
    vst->AddBarrelLayer(ibcell, 4*20*10/4, 14*10/4, 10*39.3, 14.0, 0.0);
    vst->AddBarrelLayer(ibcell, 4*20*11/4, 14*11/4, 11*39.3, 14.0, 0.0);
    vst->AttachSourceFile("vst.C");
    vst->AttachSourceFile("maps-lib.C");
    vst->AttachSourceFile("../eic/detectors/maps/VstGeoParData.cxx");
    // Specify color preferences; NB: void* interface, sorry;
    SetMapsColors((EicGeoParData*)vst);
    //
    // Fine, at this point structure is completely defined -> code it in ROOT;
    //
    vst->ConstructGeometry();
    // Yes, always exit;
    exit(0);
}
```



## A.2 Original fbst.C file from Yue Shi Lai

```
#define _VERSION_ 1
#define _SUBVERSION_ 0
// #define _TEST_VERSION_ // Do not want to always overwrite "official" files; place "test" tag into the file name;
// All construction elements are smeared (so chip assembly is uniform in both beam line and asimuthal direction);
#define _NO_STRUCTURE_GEOMETRY_
// No tricky elements like inclined roof beams; still water pipes, etc are created (so chip assembly is uniform in beam line direction only);
#define _BEAM_LINE_UNIFORM_GEOMETRY_
// FIXME: does not work; in case of FST/BST will have to fall back to tricky TGeoCompositeShape cell assemblies; also half of the cells should be
// Z-rotated to bring chips close to the beam pipe;
#define _USE_TRIANGULAR_ASSEMBLIES_
// #define _WITH_MOUNTING_RINGS_ // Comment out if no mounting rings wanted;
// #define _WITH_ENFORCEMENT_BRACKETS_ // Comment out if no stave enforcement brackets wanted;
// #define _WITH_EXTERNAL_PIPES_ // Comment out if no external water pipe pieces wanted;
#include <geometry/Maps/maps-lib.C>
#define _DISK_NUM_ (5)

fbst()
{
    // Load basic libraries;
    gROOT->Macro("$VMCWORKDIR/gconfig/rootlogon.C");
    Double_t Z[_DISK_NUM_];
    Double_t R[_DISK_NUM_];
    for (size_t i = 0; i < _DISK_NUM_; i++) {
        Z[i] = 250 + i * (1210.0 - 250.0) / (_DISK_NUM_ - 1);
        R[i] = 45.0 * (i % 2);
    }
    for(unsigned fb=0; fb<2; fb++) {
        FstGeoParData *fbst = new FstGeoParData(fb ? "BST" : "FST", _VERSION_, _SUBVERSION_);
        // Parse #define statements and make certain configuration calls accordingly;
        // NB: cast to the base MAPS geo class since void* interface used; FIXME!;
        DefinitionParser((MapsGeoParData*)fbst);
        // Prefer to think in [mm] and convert to [cm] when calling ROOT shape definition routines only;
        // Mounting ring construction; arbitrary numbers, same for all layers;
        if (fbst->WithMountingRings()) {
            fbst->mMountingRingBeamLineThickness = 3.00;
            fbst->mMountingRingRadialThickness = 5.00;
        } //if
        if (fbst->WithEnforcementBrackets()) // Simplify the design -> just a triangular piece with a reasonable volume;
            fbst->mEnforcementBracketThickness = 1.00;
        // This is something for display purposes mostly;
        if (fbst->WithExternalPipes())
            fbst->mWaterPipeExtensionLength = 4.00;
        // For now assume ALICE Inner Barrel design, composed in staves with varying number of Mimosa chips;
        MapsMimosaAssembly *ibcell = (MapsMimosaAssembly*)ConfigureAliceCell();
        // For now consider a single disc design (all the same); add staves by hand;
        // Parameters: cell assembly pointer, min available radius, max available radius, default neighboring stave overlap in "X" direction;
        // Declare discs; just put them by hand at hardcoded locations along the beam line;
        for(unsigned dc=0; dc<_DISK_NUM_; dc++) {
            FstDisc *disc = new FstDisc(ibcell, 18.0, TMath::Min(432.3, 185.0 * Z[dc] / Z[0]), 12.8);
            fbst->AddDisc(disc, (fb ? -1.0 : 1.0)*Z[dc], R[dc]);
        }
    }
    #if 0
        if (!fb) {
            FstDisc *hdisc = new FstDisc(ibcell, 18.0, 400.0, 12.8);
            fbst->AddDisc(hdisc, 1500.);
        } //if
    #endif
    fbst->AttachSourceFile("fbst.C");
    fbst->AttachSourceFile("maps-lib.C");
    fbst->AttachSourceFile("../eic/detectors/maps/FstGeoParData.cxx");
    #if 1
        TGeoPcon *support = new TGeoPcon("AluStrips_support", 0.0, 360.0, 3);
        support->DefineSection(0, (fb ? 1.0 : -1.0) * 20, 18.5 * 20.0 / 25.0, 18.5 * 20.0 / 25.0 + 0.5);
        support->DefineSection(1, (fb ? 1.0 : -1.0) * 43.23 * 0.1 * Z[0] / 18.5, 43.23, 43.23 + 0.5);
        support->DefineSection(2, (fb ? 1.0 : -1.0) * 121, 43.23, 43.23 + 0.5);
        TGeoVolume *support_volume = new TGeoVolume("AluStrips_support_volume", support, fbst->GetMedium("aluminum"));
        fbst->GetTopVolume()->AddNode(support_volume, 0, new TGeoCombiTrans(0.0, 0.0, 0.0, 0));
    #endif
    // Specify color preferences; NB: void* interface, sorry;
    SetMapsColors((EicGeoParData*)fbst);
    // Geometry is defined -> build it in ROOT;
    fbst->ConstructGeometry();
} //for fb
exit(0);
}
```

# A.3 Help menu from event display GUI

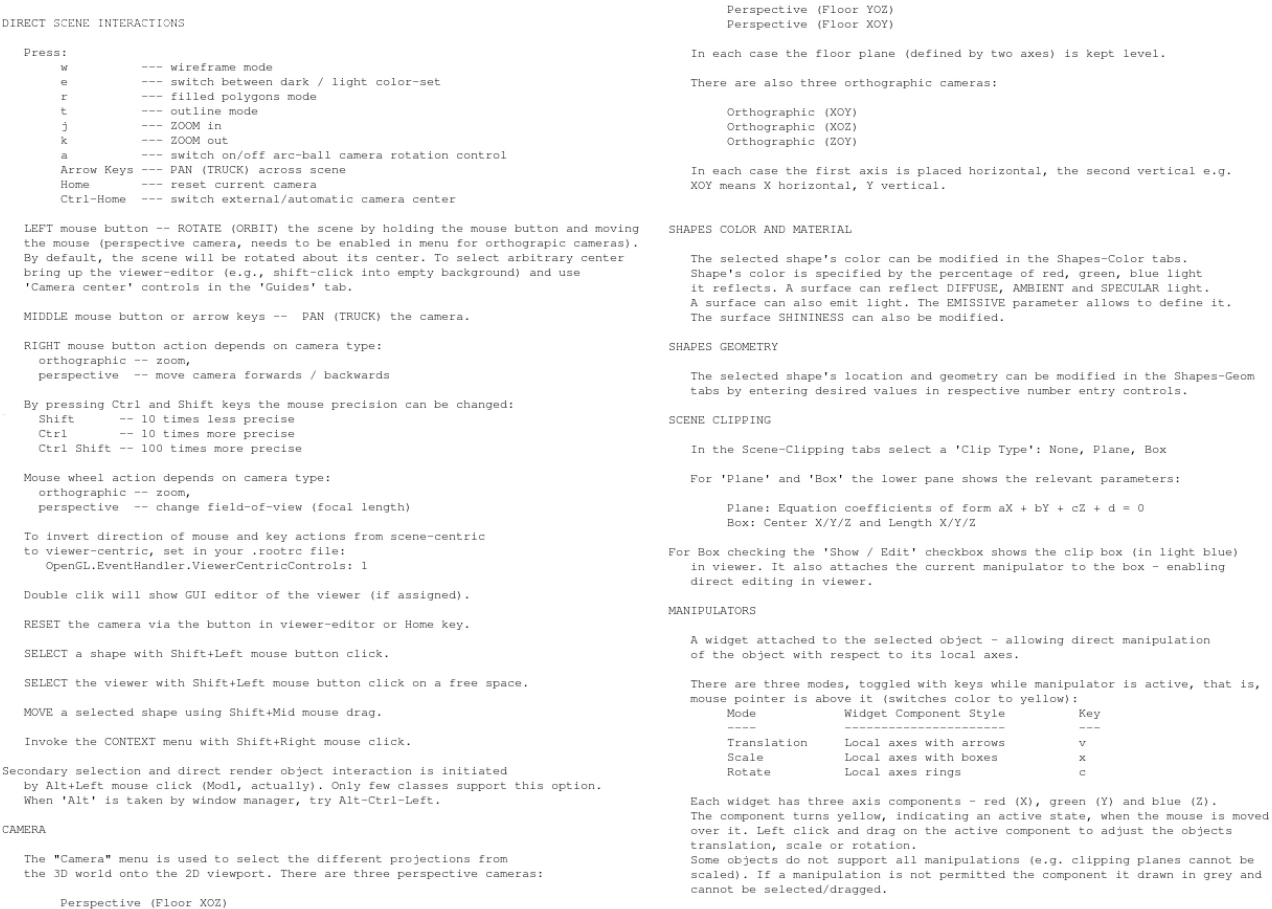


Figure 5: Event-display GUI help menu.