Geant4 for EIC (g4e)

Yulia Furletova

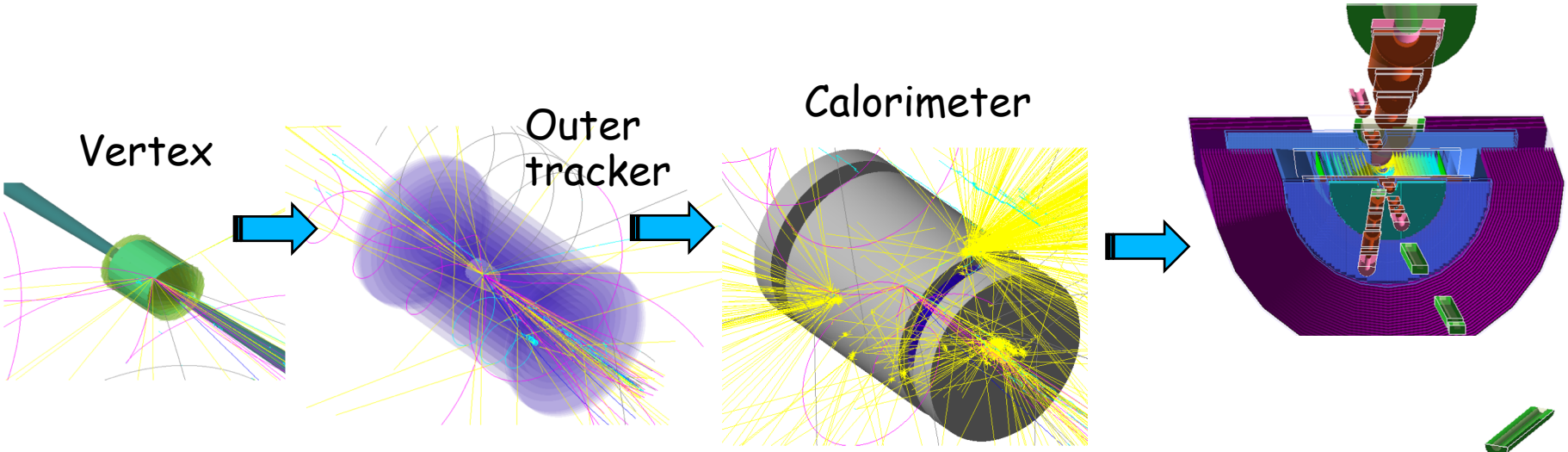
U.S. DEPARTMENT OF ENERGY
JSA
Jefferson Lab

From sub-detectors to a global detector system.

- Almost all sub-detectors within eRD have their own standalone GEANT4 simulation (detector geometry descriptions, material, sensitive detector description, digitization, etc… as well as a reconstruction code….
- A lot of discussions with EICUG and eRD20 (software consortia) about a common tools for EIC global detector system, like JLEIC, BEAST, etc and HOW to link all activities within Detector R&D program to this global system.

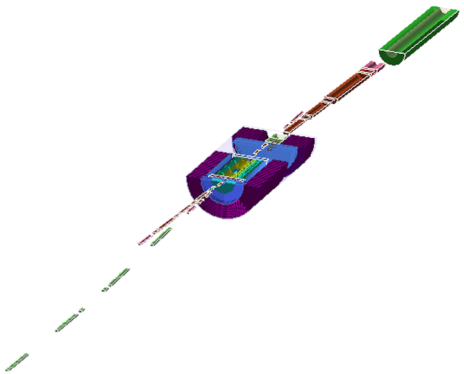
History of g4e

- Historically started as a standalone simulation for vertex detector for EIC (for open charm search and vertex resolution validation), and then has been extended by including other sub-detectors (straw –tube tracker, EMCAL, beam elements, etc….)



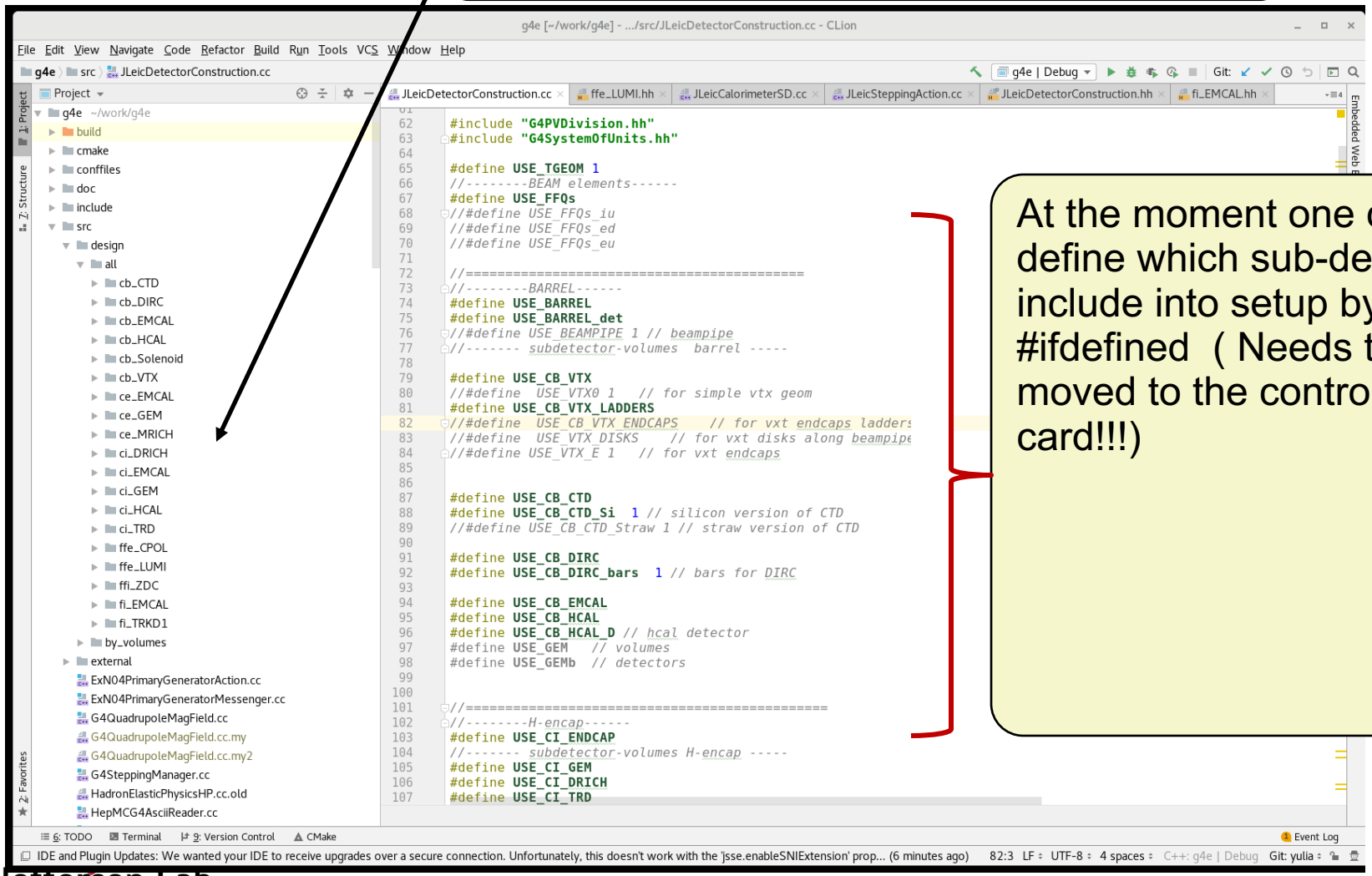
C++, standalone GEANT4

This is not ideal and need some improvements, but has all necessary features



Structure

All sub-detectors have their own class/ structure (and sub-directory for better navigation)



At the moment one could define which sub-detector include into setup by #ifdefined (Needs to be moved to the control card!!!)

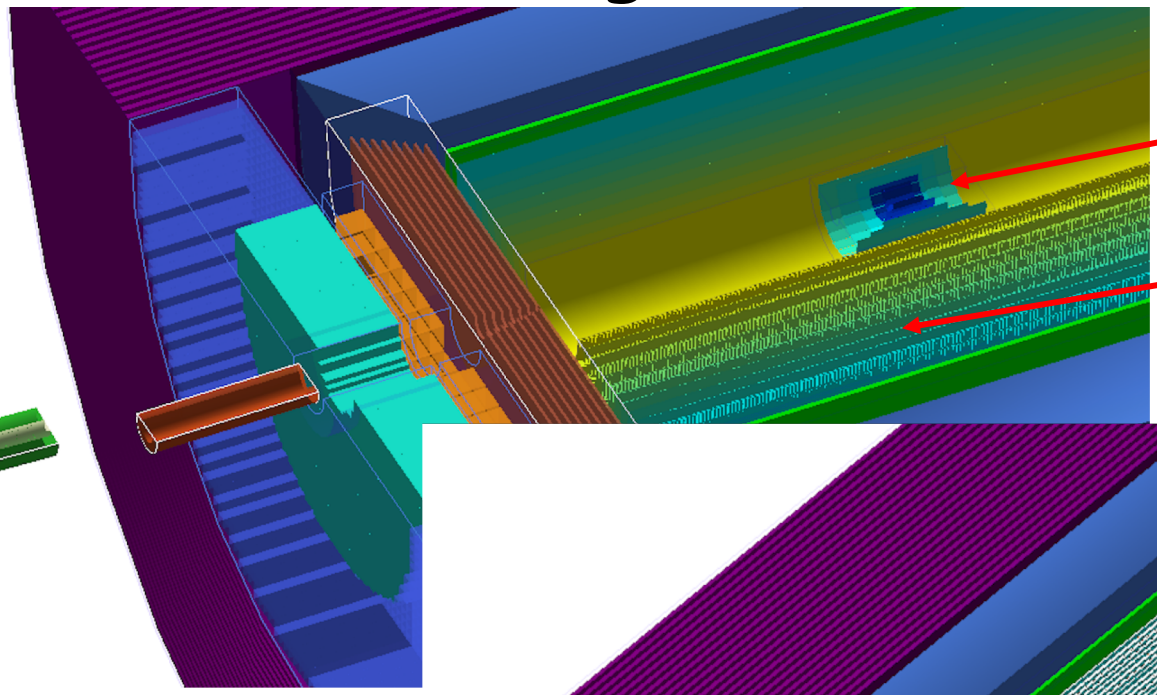
Structure

```
#ifdef  USE_BARREL_det
    //============================================================================
    //==                          VERTEX DETECTOR VOLUME                        ==
    //============================================================================

#ifdef  USE_CB_VTX

    fConfig.cb_VTX.ShiftZ = -fConfig.World.ShiftVTX;
    cb_VTX.Construct(fConfig.cb_VTX, World_Material, cb_Solenoid.Phys);

#ifdef  USE_CB_VTX_LADDERS
    //----------vtx barrel ladder geometry--------------
    cb_VTX.ConstructLaddersCentral();
#endif
#ifdef  USE_CB_VTX_ENDCAPS
    cb_VTX.ConstructLaddersEndcaps();
//            if (fLogicVTXEndE[lay]) { fLogicVTXEndE[lay]->SetSensitiveDetector(fCalorimeterSD); }
//             if (fLogicVTXEndH[lay]) { fLogicVTXEndH[lay]->SetSensitiveDetector(fCalorimeterSD); }

#endif

#endif  // end VTX

    //============================================================================
    //==                          CTD DETECTOR                                  ==
    //============================================================================
#ifdef  USE_CB_CTD

    fConfig.cb_CTD.SizeZ = fConfig.cb_Solenoid.SizeZ - fConfig.cb_CTD.SizeZCut;

    cb_CTD.Construct(fConfig.cb_CTD, World_Material, cb_Solenoid.Phys);

#ifdef USE_CB_CTD_Si
    cb_CTD.ConstructLadders();
#endif
#ifdef  USE_CB_CTD_Straw
    cb_CTD.ConstructStraws();
#endif
#endif  // end CTD
```

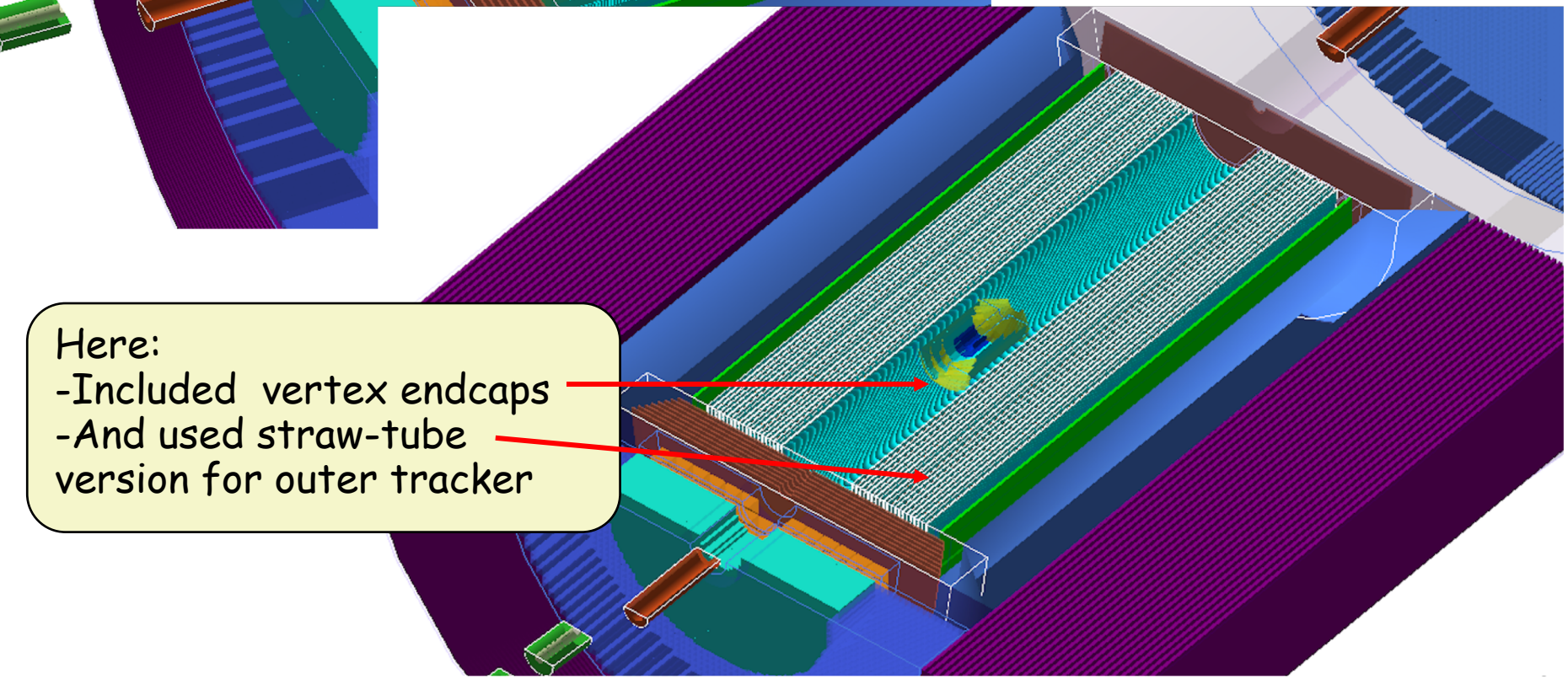
One could include parts of detector (here example, vertex end-caps)

One could choose different Versions of detectors (straw, TPC, all-silicon, etc)

Different designs of sub-detector

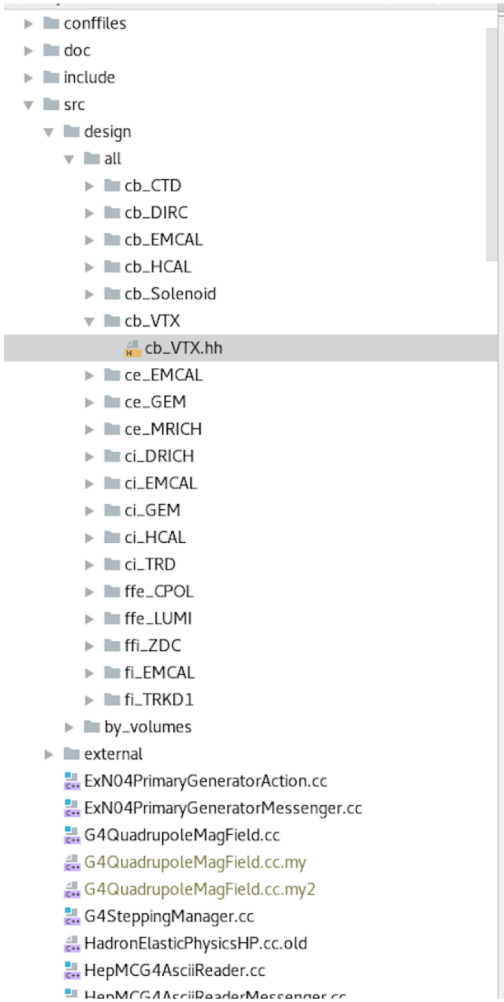


Here:
-Only barrel part of vertex det.
-And used also all-Si for outer tracker (10 layers)



Here:
-Included vertex endcaps
-And used straw-tube version for outer tracker

Sub-detector configuration/description




```
10  #include "G4Material.hh"
11  #include "G4Color.hh"
12  #include "G4VisAttributes.hh"
13  #include "G4SystemOfUnits.hh"
14
15  #include "JLeicDetectorConfig.hh"
16
17  typedef struct {
18      double Dx;
19      double Dy;
20      double Dz;
21      double Rin;
22  } cb_VTX_ladder_LayParam;
23
24  struct cb_VTX_Config {
25
26      double RIn = 3.3 * cm;      /// Inner radius
27      double ROut = 20 * cm;      /// Outer radius
28      double SizeZ = 50 * cm;     /// Guess what
29      double ShiftZ;
30      double ladder_deltashi = -7. * deg;
31
32
33  };
34
35
36  class cb_VTX_Design {
37  public:
38      inline void Construct(cb_VTX_Config cfg,G4Material* worldMaterial, G4VPhysicalVolume *motherVolume) {
39          printf("Begin cb_CTD volume \n");
40
41          ConstructionConfig = cfg;
42
43          printf("Begin cb_VERTEX volume \n");
44
45          Solid = new G4Tubs("cb_VTX_GVol_Solid", cfg.RIn, cfg.ROut, cfg.SizeZ / 2., 0., 360 * deg);
46          Logic = new G4LogicalVolume(Solid, worldMaterial, "cb_VTX_GVol_Logic");
47          Phys = new G4PVPlacement(0, G4ThreeVector(0, 0, cfg.ShiftZ), "cb_VTX_GVol_Phys", Logic,
48                                   motherVolume, false, 0);
49
50          // cb_VTX_GVol_Logic->SetVisAttributes(G4VisAttributes::Invisible);
51          G4VisAttributes *attr_cb_VTX = new G4VisAttributes(G4Color(0.1, 0, 1., 0.1));
52          attr_cb_VTX->SetLineWidth(1);
53          attr_cb_VTX->SetForceSolid(false);
54          Logic->SetVisAttributes(attr_cb_VTX);
55
```

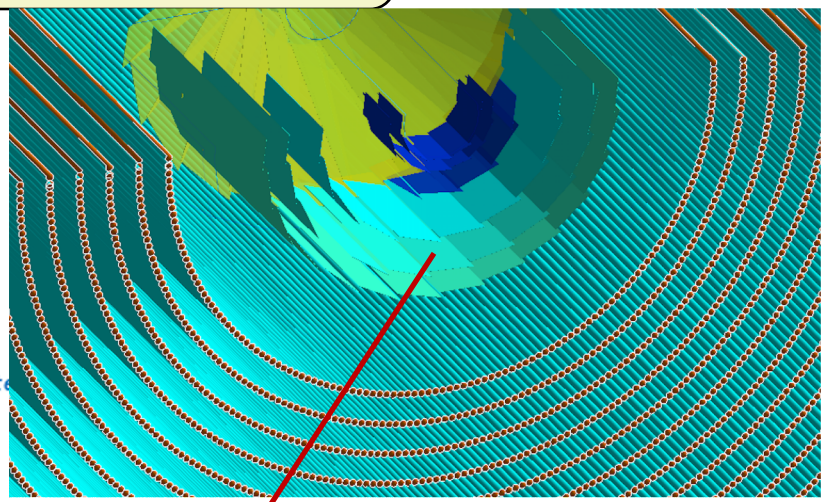
Each sub-detector has their own config structure, with parameters which will be moved to data-base

Construct “global” volume for Vertex detector. Volume, where all ladders, structure, etc will be placed.

Sub-detector configuration/description

```
inline void ConstructLaddersCentral() {
    static char abname[256];
    auto cfg = ConstructionConfig;

    //-----------------------------------------
    //----------vtx barrel ladder geometry-------
    //-----------------------------------------
    //=========================================

    G4RotationMatrix rm[10][40];
    // deltaphi1 = 0;

    double x, y, z;
    z = 0 * cm;
    //phi=26.*deg; x=0; y=0; z=fAbsorberZ;
    //phi=0.*deg; x=0; y=0; z=fAbsorberZ;

    int FDIV = 0;
    double dR;
    double myL;
    double phi=0;

    // Materials are defined in JLeicMaterials
    // TODO calling global stuff like that kills kittens, so think about it late
    cb_VTX_ladder_Material = G4Material::GetMaterial("Si");

    std::vector <cb_VTX_ladder_LayParam> Lays;
    cb_VTX_ladder_LayParam Lay;
    // Lay 0
    Lay.Dx=0.050 * mm; Lay.Dy=2*cm; Lay.Dz=10*cm; Lay.Rin=3.5 * cm; Lays.push_back(Lay);
    // Lay 1
    Lay.Dx=0.050 * mm; Lay.Dy=2*cm; Lay.Dz=11*cm; Lay.Rin=4.5 * cm; Lays.push_back(Lay);
    // Lay 2
    Lay.Dx=0.150 * mm; Lay.Dy=4*cm; Lay.Dz=18*cm; Lay.Rin=6.5 * cm; Lays.push_back(Lay);
    // Lay 3
    Lay.Dx=0.150 * mm; Lay.Dy=4*cm; Lay.Dz=24*cm; Lay.Rin=10.5 * cm; Lays.push_back(Lay);
    // Lay 4
    Lay.Dx=0.150 * mm; Lay.Dy=4*cm; Lay.Dz=36*cm; Lay.Rin=13.5 * cm; Lays.push_back(Lay);
    // Lay 5
    Lay.Dx=0.150 * mm; Lay.Dy=4*cm; Lay.Dz=48*cm; Lay.Rin=15.5 * cm; Lays.push_back(Lay);

    if( Lays.size()>10) {printf("Nlayers in VERTEX >10 !!! \n"); exit(1); }
```

Define structure of all ladders



Sensitive volumes

sensitive volumes should be added, depending on detector type.
Input from detector collaborations is needed !

g4e | Debug

detectorConstruction.cc | HepMCG4AsciiReaderMessenger.cc | HepMCG4AsciiReader.cc | JLeicCalorimeterSD.cc | JLeicStepp

```
//....ooo00000ooo.......ooo00000ooo.......ooo00000ooo.......ooo00000ooo....

G4bool JLeicCalorimeterSD::ProcessHits(G4Step *aStep, G4TouchableHistory *) {
    if (jDebug > 2) printf("--> JLeicCalorimeterSD::ProcessHits() Enter\n");



    //  const G4TouchableHandle touchablepre[128];
    G4double edep = aStep->GetTotalEnergyDeposit();

    G4double stepl = 0.;
    if (aStep->GetTrack()->GetDefinition()->GetPDGCharge() != 0.) //-- gamma ??
        stepl = aStep->GetStepLength();


    const G4TouchableHandle touchablepre = aStep->GetPreStepPoint()->GetTouchableHandle();
    const G4TouchableHandle touchablepost = aStep->GetPostStepPoint()->GetTouchableHandle();
    // depth 1 --> x
    // depth 0 --> y
    G4int copyIDy_pre = touchablepre->GetCopyNumber();
    G4int copyIDx_pre = touchablepre->GetCopyNumber(1);
    G4int copyIDz_pre = 0;
    if (use_depfet > 0) copyIDz_pre = touchablepre->GetCopyNumber(2);
    G4double xstep = (aStep->GetTrack()->GetStep()->GetPostStepPoint()->GetPosition()).x();
    G4double ystep = (aStep->GetTrack()->GetStep()->GetPostStepPoint()->GetPosition()).y();
    G4double zstep = (aStep->GetTrack()->GetStep()->GetPostStepPoint()->GetPosition()).z();
    //--- convert to the local volume --
    G4ThreeVector worldPosition = aStep->GetPreStepPoint()->GetPosition();
    G4ThreeVector localPosition = touchablepre->GetHistory()->GetTopTransform().TransformPoint(worldPosition);
    G4double xinp = localPosition.x();
    G4double yinp = localPosition.y();
    G4double zinp = localPosition.z();

    G4ThreeVector worldPosition2 = aStep->GetPostStepPoint()->GetPosition();
    G4ThreeVector localPosition2 = touchablepre->GetHistory()->GetTopTransform().TransformPoint(worldPosition2);
    G4double xend = localPosition2.x();
    G4double yend = localPosition2.y();
    G4double zend = localPosition2.z();

    G4double xloc = (xinp + xend) / 2;
    G4double yloc = (yinp + yend) / 2;
    G4double zloc = (zinp + zend) / 2;

    if (jDebug > 2) printf("xloc=%f yloc=%f zloc=%f  \n", xloc, yloc, zloc);
    if (use_fdc) {  //----- FDC / TRD  ---
```

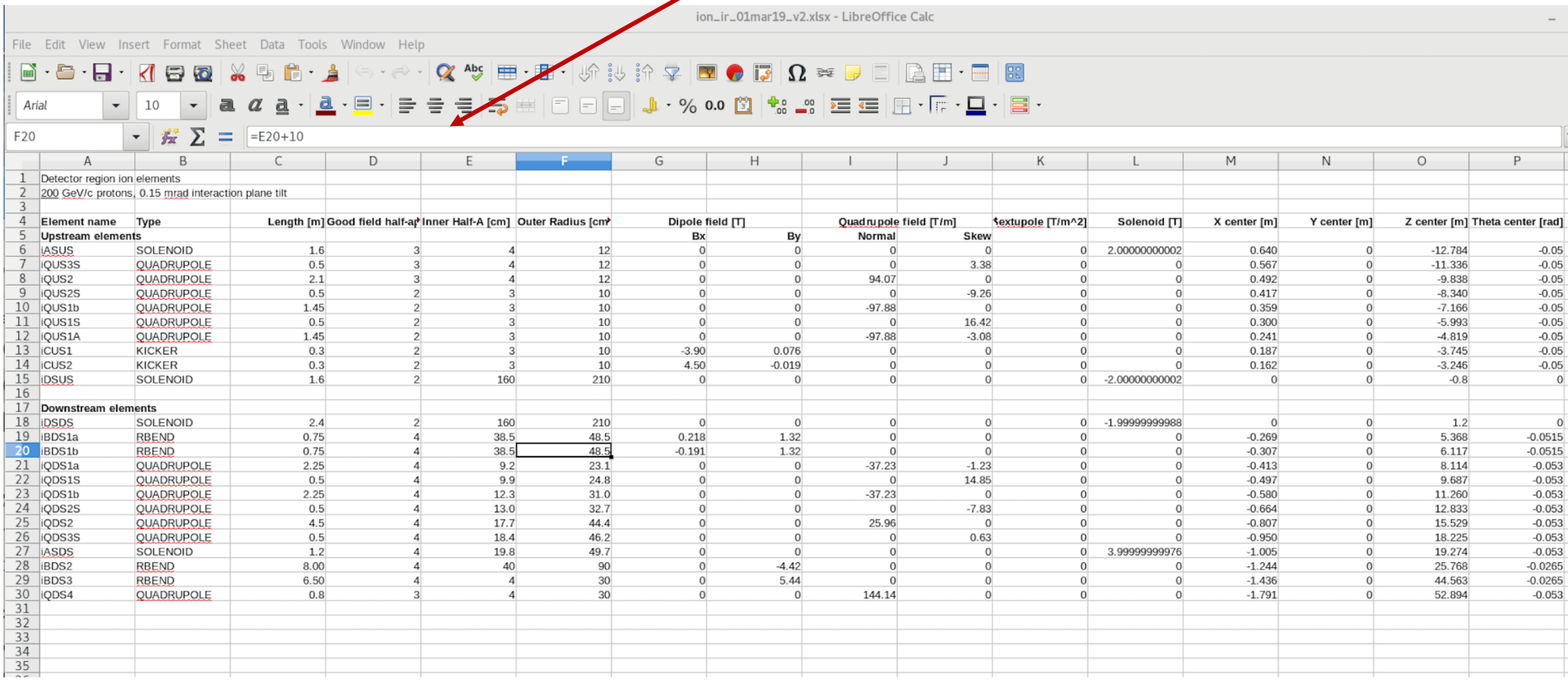
Digitization

Need to discuss how to to digitization! At the moment inside Sensitive Det definition.

```
//---------digitizing------------------------------

G4double gain = 1400. * 1.055;
G4double sigma_gain = 8. / 100.;
gain = G4RandGauss::shoot(gain, sigma_gain / gain);
short int ADC = edep / keV * gain / 59.5;

if (char_sh == 0 && copyIDy_pre < NumRow && copyIDx_pre < NumCol) FRAME[copyIDy_pre * NumCol + copyIDx_pre] +

G4TouchableHistory *theTouchable
        = (G4TouchableHistory *) (aStep->GetPreStepPoint()->GetTouchable());

if (jDebug > 3)
    printf("--> JLeicCalorimeterSD::ProcessHits() Vol: 0=%s \n", theTouchable->GetVolume()->GetName().c_str()

if (jDebug > 3)
    printf("--> JLeicCalorimeterSD::ProcessHits() Vol: 0=%s 1=%s  2=%s 3=%s Abs=%s\n",
           theTouchable->GetVolume()->GetName().c_str(), theTouchable->GetVolume(1)->GetName().c_str(),
           theTouchable->GetVolume(2)->GetName().c_str(), theTouchable->GetVolume(3)->GetName().c_str(),
           Detector->GetAbsorber()->GetName().c_str());

if (use_depfet > 0) {
    G4String VTXmod = theTouchable->GetVolume()->GetName();
    if (jDebug > 2) printf("VTX_ladder=%s \n", VTXmod.c_str());
    //   int Mod=int (VTXmod<<10);
    // printf("VTX_ladder=%d \n",Mod);
    if (jDebug > 2)
        printf("--> JLeicCalorimeterSD::ProcessHits() de=%f ADC=%d  len=%f Mod=%s  IDxy=(%d,%d,%d)\n", edep /
               ADC, stepl / um, VTXmod.c_str(), copyIDx_pre, copyIDy_pre, copyIDz_pre);
    for (int in = 0; in < 12; in++) {
        sprintf(buffer, "VTX_ladder1_%d", in);
        if (strcmp(VTXmod.c_str(), buffer) == 0) {
            runaction->FillHistmatrixOccup(in, copyIDx_pre, copyIDy_pre, edep / keV);
            runaction->FillHistmatrixOccupCM(in, xstep, ystep, edep / keV);
            if (jDebug > 2) printf("xstep=%f ystep=%f \n", xstep, ystep);
        }
    }
    if (save_frames_root) {

        for (int in = 0; in < 12; in++) {
            sprintf(buffer, "VTX_ladder1_%d", in);
```

Geometry exchange with accelerator

New updates from accelerator people every day
Test file with all geometrical parameters and field gradients



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Detector region ion elements															
2	200 GeV/c protons, 0.15 mrad interaction plane tilt															
3																
4	Element name	Type	Length [m]	Good field half-ap	Inner Half-A [cm]	Outer Radius [cm]	Dipole field [T]		Quadrupole field [T/m]		Sextupole [T/m^2]	Solenoid [T]	X center [m]	Y center [m]	Z center [m]	Theta center [rad]
5	Upstream elements						Bx	By	Normal	Skew						
6	IASUS	SOLENOID	1.6	3	4	12	0	0	0	0	0	2.00000000002	0.640	0	-12.784	-0.05
7	IQUS3S	QUADRUPOLE	0.5	3	4	12	0	0	0	3.38	0	0	0.567	0	-11.336	-0.05
8	IQUS2	QUADRUPOLE	2.1	3	4	12	0	0	94.07	0	0	0	0.492	0	-9.838	-0.05
9	IQUS2S	QUADRUPOLE	0.5	2	3	10	0	0	0	-9.26	0	0	0.417	0	-8.340	-0.05
10	IQUS1b	QUADRUPOLE	1.45	2	3	10	0	0	-97.88	0	0	0	0.359	0	-7.166	-0.05
11	IQUS1S	QUADRUPOLE	0.5	2	3	10	0	0	0	16.42	0	0	0.300	0	-5.993	-0.05
12	IQUS1A	QUADRUPOLE	1.45	2	3	10	0	0	-97.88	-3.08	0	0	0.241	0	-4.819	-0.05
13	ICUS1	KICKER	0.3	2	3	10	-3.90	0.076	0	0	0	0	0.187	0	-3.745	-0.05
14	ICUS2	KICKER	0.3	2	3	10	4.50	-0.019	0	0	0	0	0.162	0	-3.246	-0.05
15	IDSUS	SOLENOID	1.6	2	160	210	0	0	0	0	0	-2.00000000002	0	0	-0.8	0
16																
17	Downstream elements															
18	IDSDS	SOLENOID	2.4	2	160	210	0	0	0	0	0	-1.99999999988	0	0	1.2	0
19	IBDS1a	RBEND	0.75	4	38.5	48.5	0.218	1.32	0	0	0	0	-0.269	0	5.368	-0.0515
20	IBDS1b	RBEND	0.75	4	38.5	48.5	-0.191	1.32	0	0	0	0	-0.307	0	6.117	-0.0515
21	IQDS1a	QUADRUPOLE	2.25	4	9.2	23.1	0	0	-37.23	-1.23	0	0	-0.413	0	8.114	-0.053
22	IQDS1S	QUADRUPOLE	0.5	4	9.9	24.8	0	0	0	14.85	0	0	-0.497	0	9.687	-0.053
23	IQDS1b	QUADRUPOLE	2.25	4	12.3	31.0	0	0	-37.23	0	0	0	-0.580	0	11.260	-0.053
24	IQDS2S	QUADRUPOLE	0.5	4	13.0	32.7	0	0	0	-7.83	0	0	-0.664	0	12.833	-0.053
25	IQDS2	QUADRUPOLE	4.5	4	17.7	44.4	0	0	25.96	0	0	0	-0.807	0	15.529	-0.053
26	IQDS3S	QUADRUPOLE	0.5	4	18.4	46.2	0	0	0	0.63	0	0	-0.950	0	18.225	-0.053
27	IASDS	SOLENOID	1.2	4	19.8	49.7	0	0	0	0	0	3.99999999976	-1.005	0	19.274	-0.053
28	IBDS2	RBEND	8.00	4	40	90	0	-4.42	0	0	0	0	-1.244	0	25.768	-0.0265
29	IBDS3	RBEND	6.50	4	4	30	0	5.44	0	0	0	0	-1.436	0	44.563	-0.0265
30	IQDS4	QUADRUPOLE	0.8	3	4	30	0	0	144.14	0	0	0	-1.791	0	52.894	-0.053
31																
32																
33																
34																
35																

Geometry exchange with accelerator

DetectorConstruction in Geant4 reads this file and create volumes with field

```
        printf("SKIP LINE %s\n", buffer);
        continue;
    }

    sscanf(buffer, "%s %s %f %f %f %f %f %f  %f %f %f %f %f  %f %f %f %f",  ffqnameDi, ffqtype, &ffqsSizeZDi,
           &ffqsRinDiG, &ffqsRinDi, &ffqsRoutDi, &qFIELDx, &qFIELDy,
           &qFIELQn, &qFIELQs, &qFIELSek, &qFIELSol, &ffqsX, &ffqsY, &ffqsZ, &ffqsTheta, &ffqsPhi);

    printf(" NUM: |%s| %s Dz=%f %f Rin=%f Rout=%f  Dipole =%f, %f Quads =%f, %f sec =%f sol= %f xyz= %f %f %f
           ffqnameDi, ffqtype, ffqsSizeZDi, ffqsRinDiG, ffqsRinDi, ffqsRoutDi,
           qFIELDx, qFIELDy, qFIELQn, qFIELQs, qFIELSek, qFIELSol, ffqsX, ffqsY, ffqsZ, ffqsTheta, ffqsPhi);

    // ---------- create volumes for QUADRUPOLE----------
if (strcmp(ffqtype, "QUADRUPOLE") == 0) {
        printf(" found QUAD %s iq=%d \n", ffqtype, iq);
        CreateQuad(iq, ffqnameDi, ffqsSizeZDi, ffqsRinDiG, ffqsRinDi, ffqsRoutDi, qFIELDx, qFIELDy, qFIELQn,
                   qFIELQs, qFIELSek, qFIELSol, ffqsX, ffqsY, ffqsZ, ffqsTheta, ffqsPhi);
        iq++; iqmax_i=iq;
    }

    // ---------- create volumes for kickers and rbend----------
    if ((strcmp(ffqtype, "KICKER") == 0) || (strcmp(ffqtype, "RBEND") == 0)) {
        printf(" found KICKER %s \n", ffqtype);
        CreateDipole(ik, ffqnameDi, ffqsSizeZDi, ffqsRinDiG, ffqsRinDi, ffqsRoutDi, qFIELDx, qFIELDy, qFIELQn
                     qFIELQs, qFIELSek, qFIELSol, ffqsX, ffqsY, ffqsZ, ffqsTheta, ffqsPhi);
        ik++;
    }
    // ---------- create volumes for solenoid  ----------
    if ((strcmp(ffqtype, "SOLENOID") == 0) &&
        ((strcmp(ffqnameDi, "iASDS") == 0) || (strcmp(ffqnameDi, "iASUS") == 0))) {
        printf(" found SOLENOID %s \n", ffqtype);

        CreateASolenoid(is, ffqnameDi, ffqsSizeZDi, ffqsRinDiG, ffqsRinDi, ffqsRoutDi, qFIELDx, qFIELDy, qFIEL
                        qFIELQs, qFIELSek, qFIELSol, ffqsX, ffqsY, ffqsZ, ffqsTheta, ffqsPhi);
        is++;
    }
}


fclose(rc);
return;
```

Geometry and Field for magnets

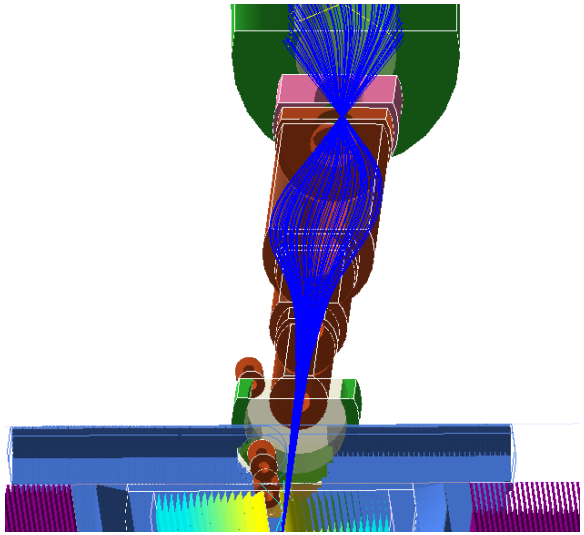
```
//--------------------Volumes ---------
sprintf(abname, "Solid_QUADS_hd_v_%s", ffqsNAME);
fSolid_QUADS_hd_v[j] = new G4Tubs(abname, 0., (ffqsRoutDi + 0.01) * cm, (ffqsSizeZDi / 2.) * m, 0., 360 * deg);
sprintf(abname, "Logic_QUADS_hd_v_%s", ffqsNAME);
fLogic_QUADS_hd_v[j] = new G4LogicalVolume(fSolid_QUADS_hd_v[j], World_Material, abname);
sprintf(abname, "Physics_QUADS_hd_v_%s", ffqsNAME);
fPhysics_QUADS_hd_v[j] = new G4PVPlacement(G4Transform3D(brm_hd[j], G4ThreeVector(ffqsX * m, ffqsY * m, ffqsZ * m)),
                                           abname,
                                           fLogic_QUADS_hd_v[j], World_Phys, false, 0);
//printf("create %s ");

//--------------------Iron---------
sprintf(abname, "Solid_QUADS_hd_ir_%s", ffqsNAME);
fSolid_QUADS_hd_ir[j] = new G4Tubs(abname, ffqsRinDi * cm, (ffqsRoutDi + 0.005) * cm, (ffqsSizeZDi / 2.) * m, 0.,
                                   360 * deg);
sprintf(abname, "Logic_QUADS_hd_ir_%s", ffqsNAME);
fLogic_QUADS_hd_ir[j] = new G4LogicalVolume(fSolid_QUADS_hd_ir[j], ffqsMaterial, abname);
sprintf(abname, "Physics_QUADS_hd_ir_%s", ffqsNAME);
fPhysics_QUADS_hd_ir[j] = new G4PVPlacement(0, G4ThreeVector(), abname, fLogic_QUADS_hd_ir[j],
                                            fPhysics_QUADS_hd_v[j], false, 0);
fLogic_QUADS_hd_ir[j]->SetVisAttributes(vb1);

//---------------- set magnetic field ---------------
sprintf(abname, "Solid_QUADS_hd_m_%s", ffqsNAME);
fSolid_QUADS_hd_m[j] = new G4Tubs(abname, 0. * cm, ffqsRoutDi * cm, (ffqsSizeZDi / 2.) * m, 0., 360 * deg);
sprintf(abname, "Logic_QUADS_hd_m_%s", ffqsNAME);
fLogic_QUADS_hd_m[j] = new G4LogicalVolume(fSolid_QUADS_hd_m[j], ffqsMaterial_G, abname);
sprintf(abname, "Physics_QUADS_hd_m_%s", ffqsNAME);
fPhysics_QUADS_hd_m[j] = new G4PVPlacement(0, G4ThreeVector(), abname, fLogic_QUADS_hd_m[j], fPhysics_QUADS_hd_v[j],
                                           false, 0);


//    G4FieldManager* fieldMgr = SetQMagField(qFIELDx[j],qFIELDy[j]);   // gradient tesla/m;
// fLogic_QUADSm[j]->SetFieldManager(fieldMgr,true);

printf("CreateQuad:: FIELD =%f %f --  %f %f -- %f %f \n", qFIELDx, qFIELDy, qFIELQn, qFIELQs, qFIELSek, qFIELSol);


fieldMgr_QUADS_hd[j] = SetQMagField(qFIELQn, qFIELQs, ffqsTheta,
                                    G4ThreeVector(ffqsX * m, ffqsY * m, ffqsZ * m));   // gradient tesla/m;

fLogic_QUADS_hd_m[j]->SetFieldManager(fieldMgr_QUADS_hd[j], true);
```

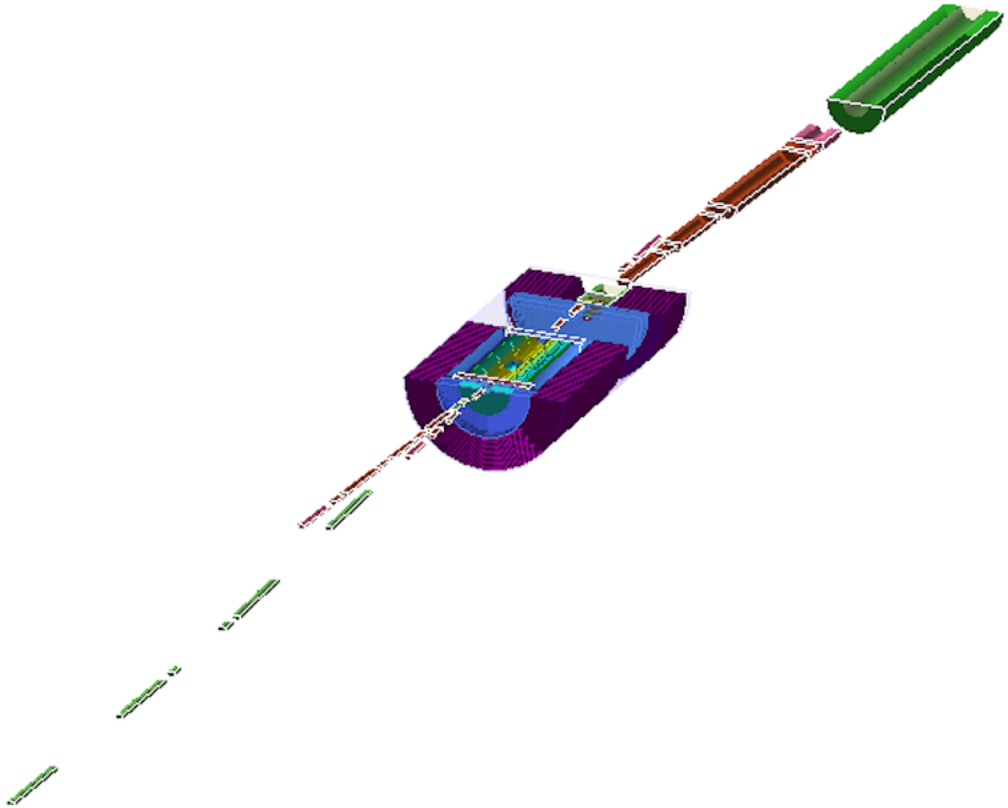
Global volume (could be converted from CAD)

"Iron"

Magnetic field

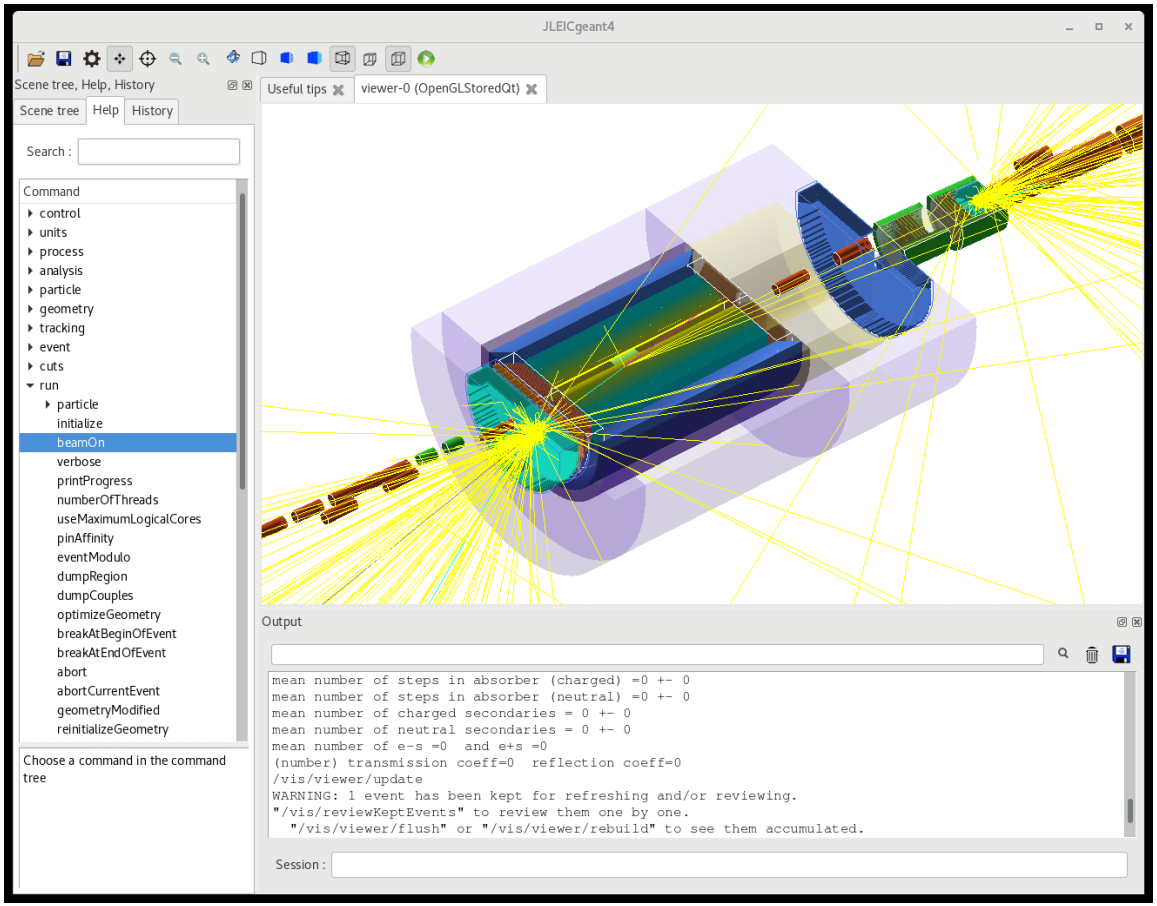
Geometry exchange with accelerator





Input files

- Particle GUN
- LUND (Pythia6)
- HEPMC (Pythia8, Herwig)
- BEAGLE (in the process of being implemented)



Conclusion

https://gitlab.com/jlab-eic/g4e/