Continue on storage classes:

PhyFieldVals.h

class PhyFieldVals

PhyElementFields cVal; // Cartesian values
PhyElementFields rVal; // rotated values

Storage for only one of the coordinate systems is:

class PhyElementFields

PhyFieldElement el;
vector<PhyFieldElement> eF;
bool bInterior;
The last level of hierarchy are values stored for ONE of the elements in ONE coordinate system.

```cpp
class PhyFieldElement

    // values
    mapPfc2Td eVals;

    // derivatives (shapes)
    mapPfc2DTd eDValDeI; // w.r.t. element interior (eIndex = -1)
    vector< mapPfc2DTd > eDValDeF; // w.r.t. elements having facets at the PIC
```

Diagram showing the relationship between values and derivatives in a coordinate system.
The second storage we need at a quadrature point

\[
\int \left( a^p + E\delta + \hat{\upsilon} \right) dV + \ldots
\]

for this integration cell

\[
+ \int \delta \hat{\nu} dS + \delta \hat{u} \nabla \hat{\nu} dS + \delta [u] \hat{\nu} dS = 0
\]
For element 0 we only need to pre-calculate two basis:

\[ H_0, H_1 \]

What if we needed to calculate \( u\text{Dot} \) (time derivative of displacement)?

\[
\dot{U}_e(l_0) = \frac{D}{Dt} U_e(l_0) = \begin{bmatrix}
\frac{D H_0}{Dt} \\
\frac{D H_0}{Dt}
\end{bmatrix}
\]

How do we calculate \( \frac{D H_0}{Dt} \)?

\[
H(x) = H(x_1, x_2, x_3)
\]
The basis of the pField can calculate $dH/dxi$:

```cpp
class phyField
PhyBasisElement pBasis;

class PhyBasisElement
void getH_DH_D2H(eCoord& crd, VECTOR &H, V2TENSOR& DH, V3TENSOR& D2H, int derOrderMax) const;
```
class eCoord
....
GeomPropBase e_gpb;
inside this we have
    GCellGeomProp* geomPropPtr;
Purely geometry class

GMeshing\GCellGeomProp.h

class GCellGeomProp
{
    vector<double> v0; // coordinate of the first vertex
    MATRIX grad_dX_dAlpha;
    MATRIX grad_dAlpha_dX;
    double omega2Alpha;

    // and other members related to normal vectors
}

// look over this class for general geometry calculations

------
We need these shapes to
1. Calculate the values
2. Calculate the FEM shapes (der. w.r.t. element unknowns) of interpolated fields.
These are the building blocks of whatever we need to calculate FIELDS THAT ARE INTERPOLATED.

physics\PhyStoreBasis.h
This file has storages for H, DH/DX, D2HDX2, ... of distinct basis

---- look at this class
class IntHStorage (storage for all elements at a quad point)
class ElShapeStore (storage for ONE of the elements at the quad point)

Final note:
Where are these storages used? Pretty much everywhere.

Look at this function in PhyElementBase class

```c++
void PhyElementBase::ComputeFieldIntegrand(ptCoords& crds, PhyFieldVals& fldVals, IntHStorage& basisShapes, int e_Index, compT cT, phyFld fld, vsT cVH, rotT rT)
```