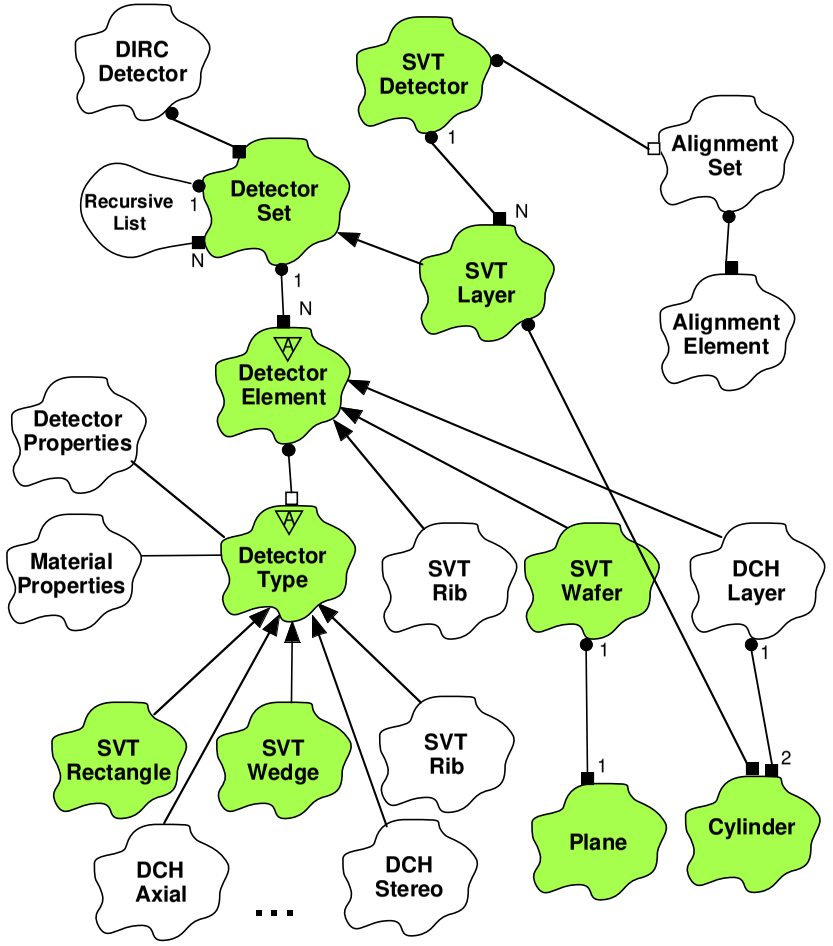
Tracking Material + Geometry

- We present the design and partial implementation of material and geometry classes for tracking
  - Common design for SVT, DCH, and DRC
  - Precision alignment is built in to the design
  - Design is compatable with Kalman Filtering
  - Flexible, OO design
  - Use of Geant4 is not excluded by the design
- Detector Classes
- SVT implementation
- Geometric and Tracking Classes
- Alignment Classes
- Open Questions

Detector Classes

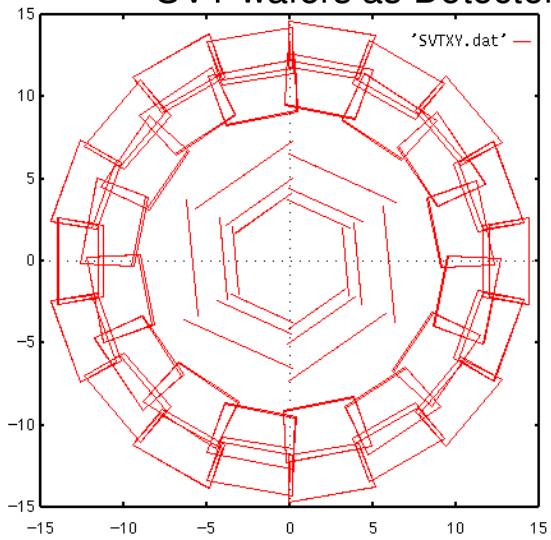


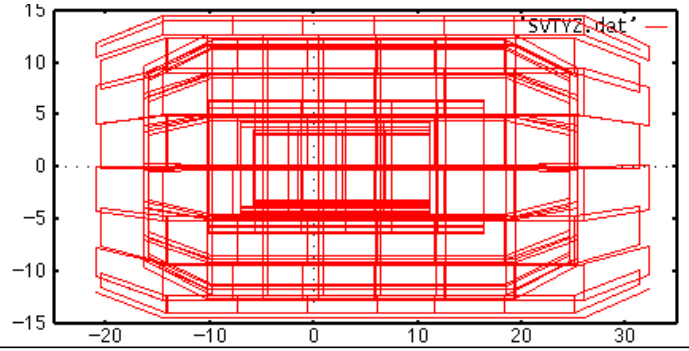
Detector Classes

- XXXDetector
  - Owns the geometric description of a full detector
  - Database interface
- DetectorSet
  - Recursive list of DetectorSets
  - List of DetectorElements
  - High level tracking/material interface class
    - Create 'KalmanSites' for material interactions
  - Base Class for specific implementions
    - SVT Layer allows 'smart' navigation
- DetectorElement
  - One object per physical piece of the detector
  - Base class for specific elements (SiWafer, DCH layer, DIRC bar, etc)
  - Points to a DetectorType
  - Models the geometric properties
    - Position in space (Aligned coordinate Transform)
    - Surface(s)
    - Converts a measurement into a Hit Trajectory
- DetectorType
  - One object per individual (unique) detector piece
  - Base class for specific types (SiRectangle, DCH Layer,...)
  - Describes shape, size, and material composition
  - Interface with material properties model

SVT Implementation

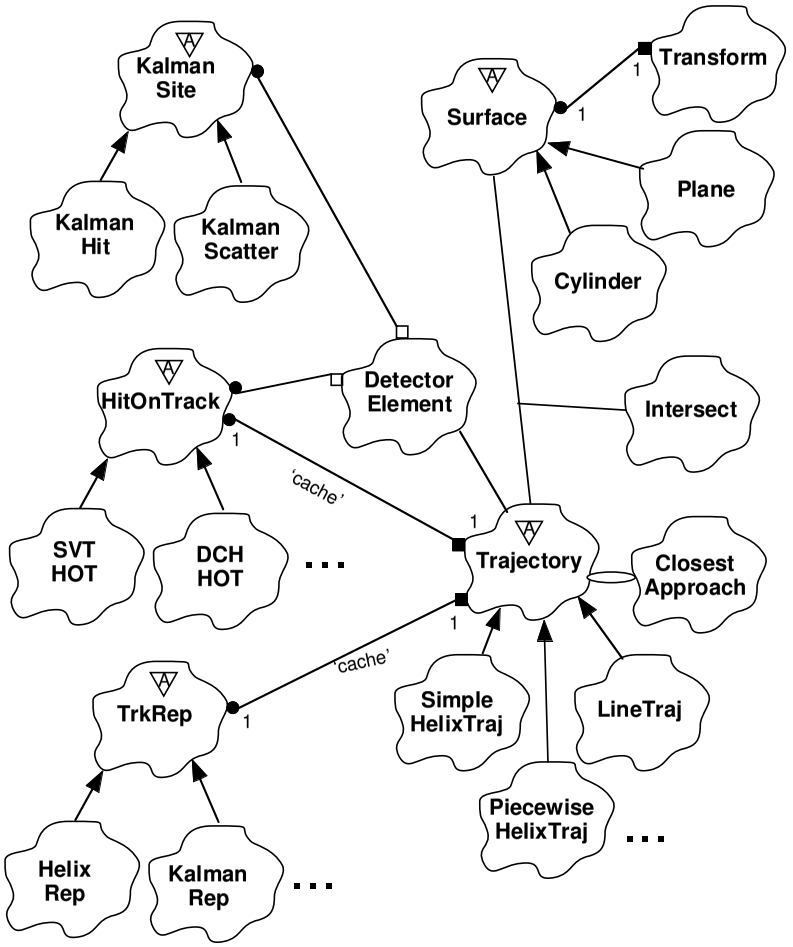
Fully functional 3D implementation of SVT wafers as DetectorElements





Drawing method output of DetectorElement + DetectorSet (gnuplot)

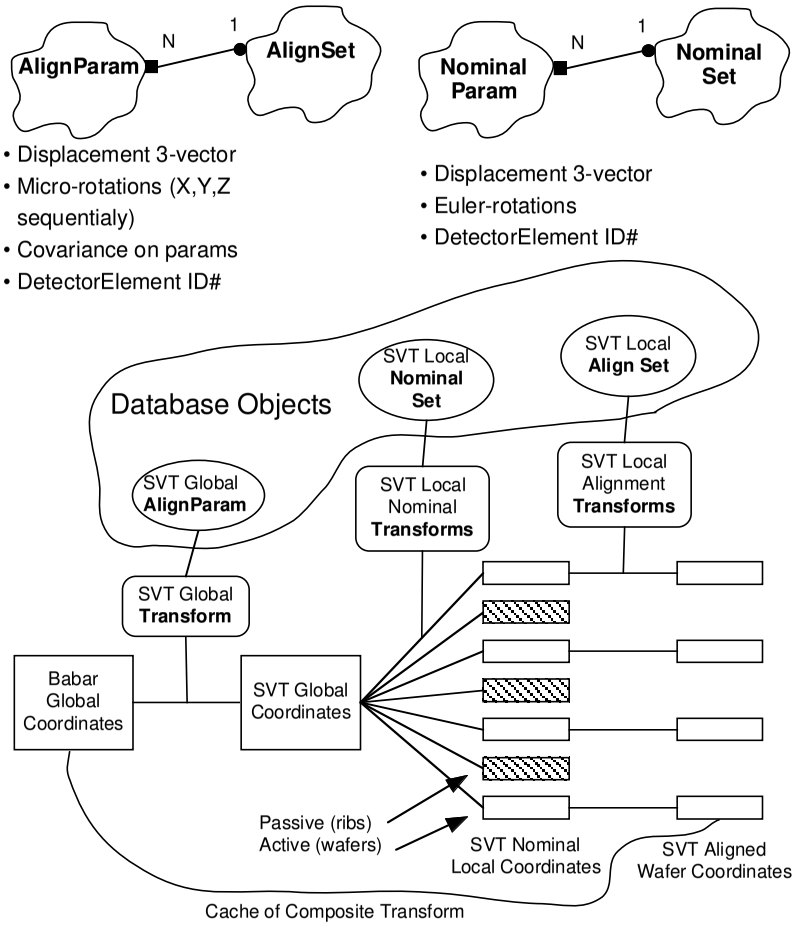
Tracking + Geometry Classes



Tracking + Geometry Classes

- Surface
  - Defines 2D object in space by a transform + equation
  - Defines 'surface coordinates'
- Trajectory
  - Defines 1D objects in space
  - Surface intersect function helper class (Intersect)
  - Closest Approach helper class (POCA)
  - Geometric information as a function of flight path
- Hit On Track
  - Points to its DetectorElement
  - Caches spatial representation as a trajectory (through DetectorElement + Type)
- Track Representation
  - Caches spatial representation as a trajectory
  - Tracking-specific functions (fit, add/subtract hit, ...)
- KalmanSite
  - Where 'something happens' to the track
  - KalHit
    - Measurement information
  - KalScatter
    - Scattering, dE/dX, and fringe B-field effects
  - Created from Track + Hit + DetectorElement

Alignment Classes



Open Questions

Should Transforms keep a history of their concatonation (i.e. do we need a NestedTransform class)?

Yes	No
• Updates when the database changes might be more efficient • Intermediate access is self-describing • Concatonated transform is 'automatic'	• Simpler • Concatonation order must be controled from outside, so the internal history of the NestedTransform is redundant

Can this same design be used for other subsystems (at least in their interaction with tracking)?