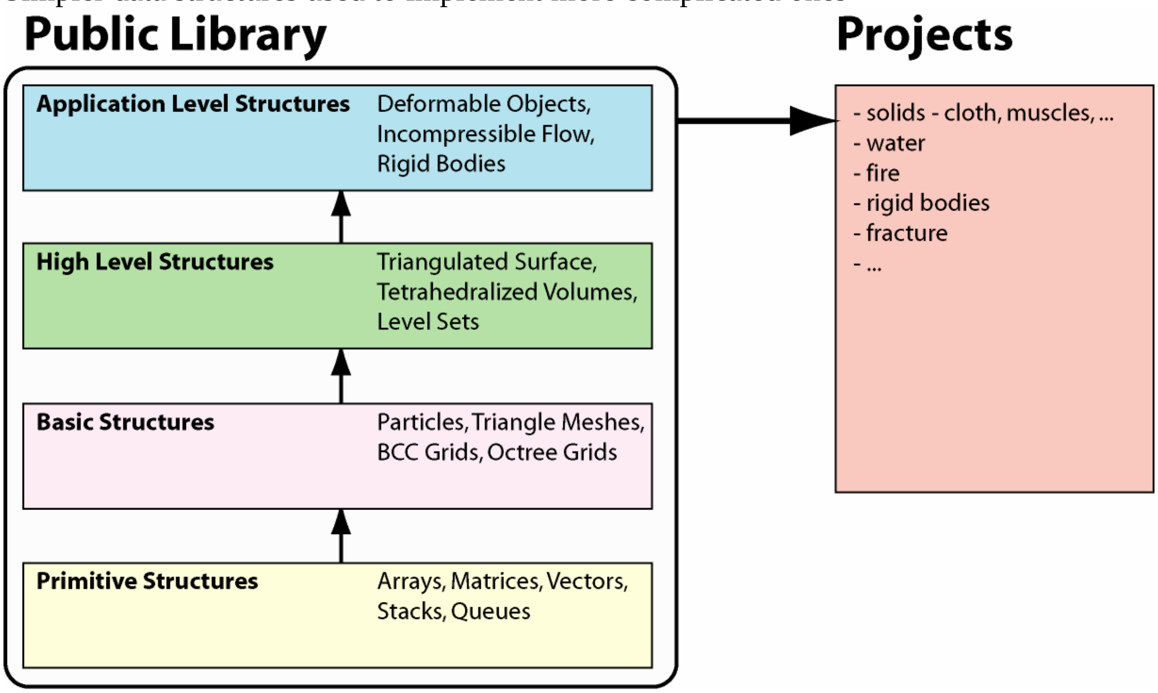
PhysBAM Architecture

1. **General ideas**
   a. Use object oriented style to associate data, models and related methods
   b. Everything that has real numbers is templatized at least on scalar values (often T)
   c. Containers templatized for maximum reuse
   d. Use as efficient algorithm as possible
   e. Simpler data structures used to implement more complicated ones
   f. 

2. **Primitive Structures**
   a. Many data structures based on Arrays (Public_Library/Arrays)
      i. Bounds Checked in debug mode with assert. Finds lots of bugs in algorithms
      ii. Handles allocating and freeing memory automatically from heap
      iii. Have lots of useful algorithms
         1. Sort, Heap, exchange, min, max, etc.
      iv. Lots of variants for various purposes
            a. ARRAYS_1D, ARRAYS_2D, ARRAYS_3D
            b. allow arbitrary start and end indices
            c. used for fluids, images, anything needing boundary conditions
         2. LIST_ARRAY/LIST_ARRAYS
            a. Like vector in Java
            b. Good for when you don't know the size
            c. Usually want to pre-allocate larger block so frequent resizes not necessary
         3. ARRAYS

a. Has another dimension (length)
b. Used for specifying point indices for a list of triangles...length=3

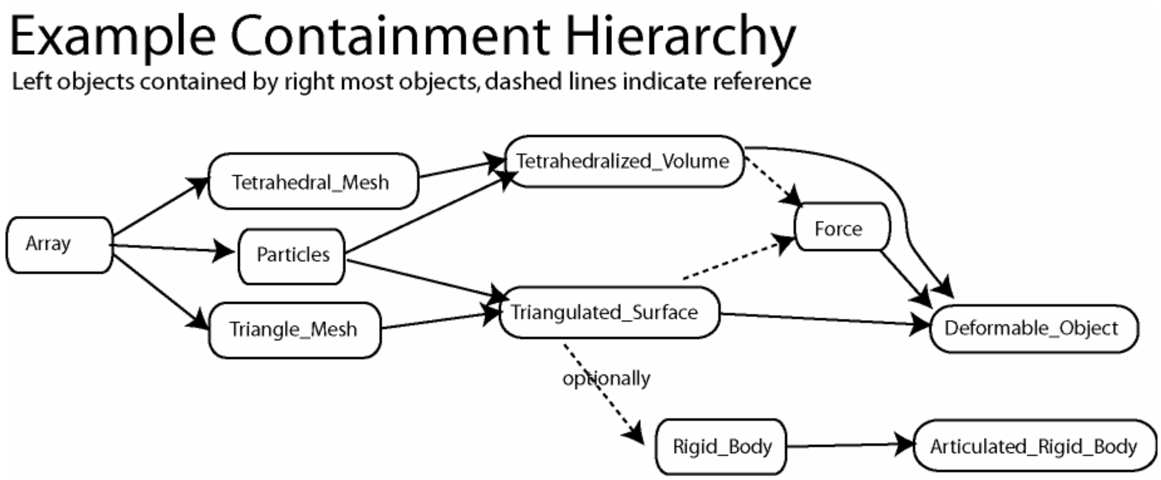
b. Other useful data structures (most built using arrays) (Public_Library/Data_Structures)
   i. Stacks and Queues
   ii. Splay Trees
   iii. Undirected/Directed Graphs
c. Vectors and Matrices (Public_Library/Matrices_And_Vectors)
   i. General versions
      1. Dense matrices MATRIX_MXN, MATRIX_NXN and vectors VECTOR_ND
      2. Sparse matrices SPARSE_MATRIX_NXN and SPARSE_VECTOR_ND
   ii. Specialized low dimensional versions
      1. matrices MATRIX_2X2, MATRIX_3X3, MATRIX_4X4, SYMMETRIC_MATRIX_2X2, SYMMETRIC_MATRIX_3X3
      2. vectors VECTOR_1D, VECTOR_2D, VECTOR_3D

3. **Basic structures**
   a. Particles (Public_Library/Particles)
      i. Set of points with other data
      ii. Implemented as a set of PARTICLE_ATTRIBUTES
         1. Contain arrays
         2. add attribute specific functionality (e.g. Euler step for position/velocity)
      iii. PARTICLE base class templatized on both scalar type and vector type.
      iv. Derived classes include additional PARTICLE_ATTRIBUTES. i.e. SOLIDS_ATTRIBUTES contain position, velocity, mass. PARTICLE_LEVELSET_PARTICLES include position and radius.
   b. Grids
      i. Provide topology
      ii. Map data to locations in space
      iii. Separation of topology and spatial mapping from data allows for more flexibility and efficiency when swapping or manipulating data.
      iv. Examples: TRIANGLE_MESH, SEGMENT_MESH, BCC_GRID, OCTREE_GRID, GRID_3D.
4. **High Level Structures**
   a. These bring together different basic and primitive structures
   b. Provide both an abstraction for the data as well as a interface to all the algorithms associated.
   c. Examples
      i. Triangulated_Surface (Public_Library/Geometry)

            1. Uses particles to store spatial locations
            2. Uses a triangle mesh to store connectivity
            3. Provides routines for computing normals, intersecting with rays, etc.
        ii. Tetrahedralized_Volume (Public_Library/Geometry)
            1. Also uses particles and mesh
            2. Additionally contains a Triangulated_Surface for boundary that uses some subset of volume's particles
        iii. Level Sets
            1. Uses grid to describe spatial domain and resolution
            2. Uses ARRAYS_3D to store set of samples for implicit surface
            3. Provides routines e.g. for evolving level set, computing properties like normals, computing inside
    d. Reused on many applications

5. **Application Level Structures**
    a. Data Classes
        i. Bring together all preceding structures to provide some application specific functionality.
        ii. May be used by multiple projects but are rather specialized.
        iii. Examples
            1. DEFORMABLE_OBJECT
                a. Has a TRIANGULATED_SURFACE and/or a TETRAHEDRALIZED_VOLUME
                b. Has additional data on particles for velocities
                c. Has an ARRAY of force objects that will be applied when advancing the body
            2. DEDFORMABLE OBJECT LIST
                a. Contains a set of DEFORMABLE_OBJECTS
                b. Handles collisions using a joined TRIANGULATED_SURFACE
            3. INCOMPRESSIBLE_3D
                a. Contains a velocity field as a GRID_3D and ARRAYS_3D
                b. Contains a POISSON class to make a velocity field divergence free

Example containment relationships:



- b. Evolution Classes
    - i. Contain multiple application data classes.
    - ii. Handle advancing data in time. Uses functions defined on data classes
    - iii. Can easily be swapped with different classes to vary the evolution techniques.
    - iv. Examples
        1. PARTICLE_LEVELSET_EVOLUTION → Evolves PARTICLE_LEVELSET
        2. RIGID_BODY_EVOLUTION → Evolves Rigid Bodies
        3. SOLIDS_EVOLUTION → Evolves both rigid bodies and deformable bodies
- c. Drivers and Examples
    - i. Drivers
        1. Handle steps required to get each frame of simulation
        2. Call Functions on evolution classes
    - ii. Example
        1. Holds instances of
            - a. Evolution Classes
            - b. Data classes
            - c. Simulation Parameters
        2. Derived examples
            - a. Inherit from base Example class
            - b. Implement various call backs.
                - i. e.g. constraining velocities and positions for a cloth simulation

6. **Projects**
    - a. Implement and instantiate a driver (and sometimes an example)
    - b. Many derived examples
    - c. Bring in external libraries
    - d. Present front end … GUI, etc.
    - e. For example
        - i. Solids (Projects/solids)

1. Uses a generic driver
2. Many derived examples
    a. Curtain_And_Ball (Triangulated Surface)
    b. Torus (Tetrahedralized)

ii. water_free_surface_3d (Projects/water_free_surface_3d)
1. Uses a specialized driver but a generic SOLIDS_FLUIDS_EXAMPLE_3D
2. Many derived examples
    a. Falling_Drop
    b. Filling_Cup