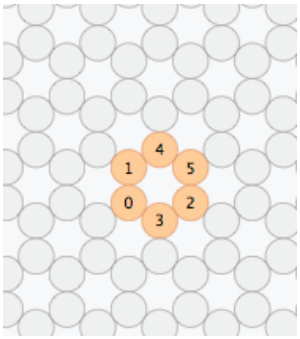
caDNAno Model

Part

A meaningful unit of structure (the user decides what that means) is called a "part". The idea of DNA origami is to build a part out of DNA. Since DNA naturally forms base-paired helices, a part is composed of many base-paired helices stuck together with "staples," strands of DNA that cross from one helix to another.



The helices from which a part is constructed can be thought of as subsets of larger virtual helices which span the entire height of the part (they run along the Z axis). It is these virtual helices that are represented in the model as VirtualHelix objects and displayed in the path view. You can acquire a VirtualHelix object from a DNAPart object with a vhref, which is a fancy name for any one of the following three things using part.getVirtualHelix(vhref). Anywhere you see "vhref" (by convention) you can use any of the following:

The number displayed on the helix in the slice view (left)
The (row, col) location in the slice view – just count # circles down, # from the left
An actual VirtualHelix instance

Helix

A VirtualHelix is composed of a list of bases (in two conceptual strands, the scaffold and staple strands, represented by rows of bases) that can be linked together (depicted as a horizontal line). Bases that aren't linked together have no physical counterpart in the final structure – they exist in a conceptual sense only to help space the DNA strands present in the final structure (bases that *have* been linked together). Bases are typically referred to using coordinates collectively referred to as a "baseref". Confusingly, there are two kinds of baseref in common use.

virtualHelix is a VirtualHelix, of course
strandType is one of StrandType.Scaffold, StrandType.Staple (from enum impor t StrandType)
baseIndex is the index (starts at 0) from the left (in the path view) of the base
First kind of baseref:
(virtualHelix, strandType, baseIndex)
Second kind of baseref (omit VH when it's implied, for instance when calling a method on a VH):
(strandType, baseIndex)
Sometimes these are actual tuples and sometimes they're just a set of suggestively named variables or parameters. They occur in the indicated order if they are parameters.

Base

Here are some definitions which allow one to make sense of the VirtualHelix methods that provide information about its bases.

vh.has{Neighbor,Crossover}{5p,3p,L,R}(strandType, baseIndex):
(Actual) *Neighbor:* a base connected to another base through a 3′ or 5′ linkage
Natural Neighbor: the base immediately to the left or right (same strand) of a given base
L, R: correspond to whichever of {5p, 3p} lies in the left and right directions, respectively
Crossover: a 3′-5′ linkage between two bases that are not natural neighbors of eachother
Strand(1): a row of bases in a VirtualHelix that share a conceptual role as scaffold or staple bases
Strand(2): (yep, it's ambiguous) a set of bases all connected through 3′-5′ linkages
Segment: a maximal set of connected bases.

vh.connectsToNat{5p,3p,L,R}(strandType, baseIndex):
True iff a base connects to its natural neighbor in the given direction

vh.neighbor{5p,3p,L,R}(strandType, baseIndex):
This gives you a baseref (type 1) for the actual neighbor of the receiver in the indicated direction