*Collin McCarthy*

*cmccarthy@ucdavis.edu*

*9/29/2013*

*Boxfish Changes for 5D Torus Module Integration*

1. *Boxfish.mods.\_\_init\_\_.py [mods/\_\_init\_\_.py]*
   1. Add the 5d torus module name and view to the import list as follows:

*import Torus5dModule*

*import Torus5dView*

* 1. Remove import Patch3dModule from the module list since if the user accidentally drops that module onto the drawing area the program needs to be restarted before any other module will accept a drop and be able to run

1. *DataModel.TableItem.createIdAttributeMaps(): [DataModel.py]*
   1. Change the for loop, starting at line 539, as follows to allow attribute dictionary to contain lists, which is necessary for 5d data when two link id’s map to the same group:

for group, id in zip(attribute\_groups, ids[0]):

id\_dict[id] = group

if group in attr\_dict:

attr\_dict[group] = list(np.array([attr\_dict[group],id]).flatten())

else:

attr\_dict[group] = id

* 1. Example of why this is necessary:
     1. link\_id[8] has source coordinates (a, b, c, d, e) == (0, 0, 0, 0, 0) and dest coordinates (0, 0, 0, 0, 1), and the group is the combined tuple (0, 0, 0, 0, 0, 0, 0, 0, 0, 1). Since the e-dimension has a size of two, and every dimension has positive and negative links, link\_id[9] also has the same group. Thus, before this change, we would have:

id\_dict[8] == (0, 0, 0, 0, 0, 0, 0, 0, 0, 1) # good

id\_dict[9] == (0, 0, 0, 0, 0, 0, 0, 0, 0, 1) # good

attr\_dict[(0, 0, 0, 0, 0, 0, 0, 0, 0, 1)] == 9 # bad, should be [8, 9]

After this change we have:

id\_dict[8] == (0, 0, 0, 0, 0, 0, 0, 0, 0, 1) # good

id\_dict[9] == (0, 0, 0, 0, 0, 0, 0, 0, 0, 1) # good

attr\_dict[(0, 0, 0, 0, 0, 0, 0, 0, 0, 1)] == [8, 9] # good

Note that not all attr\_dict values are lists, i.e. a value of seven is just the constant, 7, *not* the list containing the constant seven, [7].

* 1. Diff output:

$ diff boxfish/DataModel.py boxfish/DataModel\_old.py

541,545c541

< if group in attr\_dict:

< attr\_dict[group] = list(np.array([attr\_dict[group], id])

< .flatten())

< else:

< attr\_dict[group] = id

---

> attr\_dict[group] = id

1. *Table.group\_attributes\_by\_attributes(): [Table.py]*
   1. *Change the for loop, starting at line 249, to be as follows to create the correct attribute groups and ids necessary for DataModel.TableItem.createIdAttributeMaps() to successfully build the link\_to\_coord dictionary:*

for group\_tuple in group\_dict:

if len(group\_dict[group\_tuple][0]) > 0:

for i, attr\_list in enumerate(desired\_list):

aggr\_values = np.array(self.operator[aggregator](

group\_dict[group\_tuple][i])).flatten()

for i in aggr\_values:

attr\_list.append(i)

group\_list.append(group\_tuple)

* 1. Example of why this is necessary:
     1. For aggregator == ‘N.A.’ and group\_tuple == (0, 0, 0, 0, 0, 0, 0, 0, 0, 1), we’d have group\_dict[group\_tuple] == [[8, 9]]. Without making this change, we would append the list [8, 9] to the desired\_list, instead of the individual linkid’s 8 and 9. We could modify the code in DataModel.TableItem.createIdAttributeMpas() to account for this, but it’s simpler to take care of it in this method by appending the individual linkid’s. Also, whenever more than one item is added to attr\_list, we also want the corresponding group\_tuple to be appended to the group\_list, so the zip() function will work properly in DataModel.TableItem.createIdAttributeMaps().

Note: If aggregator is anything other than ‘N.A.’, an aggregator function will be run on the list group\_dict[group\_tuple], reducing it to one value as expected. Thus this modification only affects the behavior when aggregator == ‘N.A.’

* 1. Diff output:

$ diff boxfish/Table.py boxfish/Table\_old.py

250a251

> group\_list.append(group\_tuple)

252,256c253,254

< aggr\_values = np.array(self.operator[aggregator](

< group\_dict[group\_tuple][i])).flatten()

< for i in aggr\_values:

< attr\_list.append(i)

< group\_list.append(group\_tuple)

---

> attr\_list.append(self.operator[aggregator](

> group\_dict[group\_tuple][i]))

The following is a suggestion, but because it’s not necessary, it has not been implemented yet.

1. *ModuleFrame.realize(): [ModuleFrame.py]*
   1. Replace ‘self.view’ with ‘self.glview’ – lines 101, 109
   2. Example of why this is an improvement:
      1. Before this change, any module frame must set self.glview in createView() as follows:

def createView(self):

self.glview = GLTorus2dView(self, self.dataModel)

return self.glview

But this should just return the view object, as in previous versions of Boxfish, as follows:

def createView(self):

return GLTorus2dView(self, self.dataModel)

Note that modules written to explicitly set self.glview in createView(), as above, will continue to work as written after making this change.