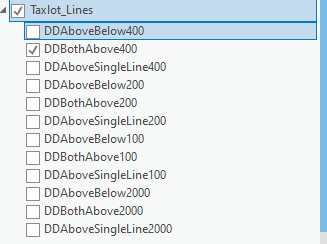
**Labeling and Annotation Test**

**Dean Anderson (Polk Co)**

As a test I created several different label classes for Taxlot\_Lines bearing/distance. For this to work I had to set my Map Reference Scale to be 1:1200.



All annotation classes used the same python expression for labelling for the following rules:

1. If Radius only dislay it (r=)
2. Always display 2 digits after the decimal for distance
3. Always display 2 digits for degrees (02°) instead of 2°

Below is an example of expression for bearing/distance above.

------------------------------------------------------------------------

def FindLabel ( [Direction],[Radius],[Distance] ):

if [Radius] != None:

z = float([radius])

txt = "{:.2f}"

rad = "r = " + txt.format(z)

return rad

else:

z = float([Distance])

txt = "{:.2f}"

dis = txt.format(z)

if ([Direction][2:3]) == "°":

dir = ([Direction][:1]) + " 0" + ([Direction][1:])

beardir = dir + "\n" + dis

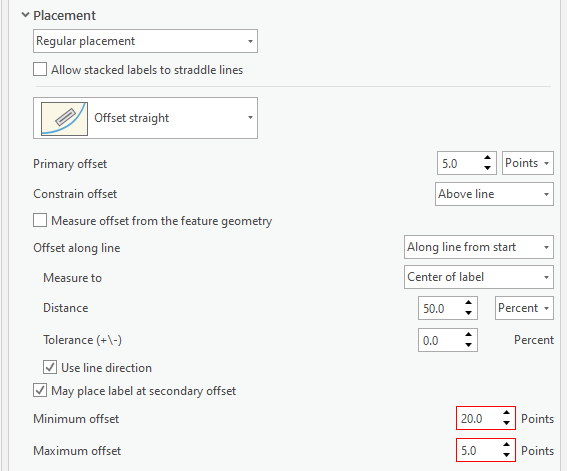
return beardir

else:

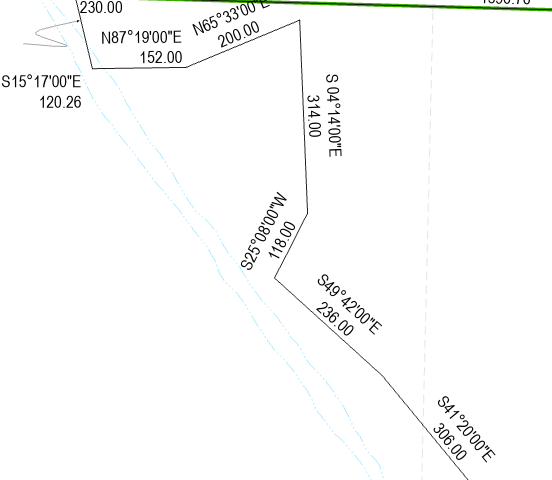
beardir = [Direction] + " \n" + dis

return beardir

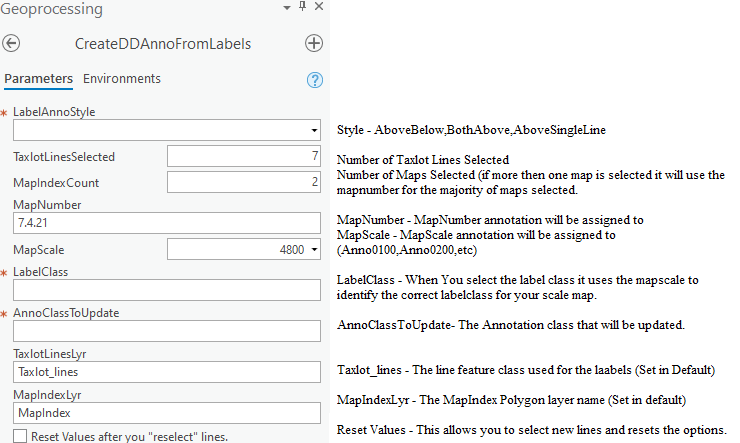
The difference in scales is primarily in the size of the symbol. For example a 1:400 label uses a font size of 20 and 1:200 a font size of 10. I am using Aerial Narrow as the font name and style. The position for the annotation varies depending on what needs to be done. Below is the example for BothAbove style. Primary Offset from the line is scale dependent as is secondary offsets (cool cause it is place anno that does not fit the line length).



The labels appear ok as follows:



To turn these labels into annotation I used the ConvertLabelsToAnnotation tool and drove it from a script “CreateDDAnnoFromLabels”. This allows the user to change label styles and selections to see how it will look before converting. The script tool appears as follows:



Two of the validation functions are used to do this work. Have no figured how to use the parameters to set the layer features.

def \_\_init\_\_(self):

# set self.params for use in other function

TaxlotLineLyr = "Taxlot\_lines"

MapIndexLyr = "MapIndex"

self.params = arcpy.GetParameterInfo()

aprx = arcpy.mp.ArcGISProject("CURRENT")

editMap = aprx.activeMap

TaxlotLines = editMap.listLayers(TaxlotLineLyr)[0]

MapIndex = editMap.listLayers(MapIndexLyr)[0]

self.params[7].value = TaxlotLineLyr

self.params[8].value = MapIndexLyr

TaxlotLinesCnt = 0

if TaxlotLines.getSelectionSet() != None:

TaxlotLinesCnt = len(TaxlotLines.getSelectionSet())

#TaxlotLinesCnt = int(arcpy.GetCount\_management(TaxlotLines).getOutput(0))

self.params[1].value = TaxlotLinesCnt

arcpy.SelectLayerByLocation\_management(MapIndex, "intersect", TaxlotLines)

MapIndexCnt = 0

if MapIndex.getSelectionSet() != None:

UniqueMapList = []

MapList = []

with arcpy.da.SearchCursor(MapIndex,['MapNumber','MapScale']) as cursor:

for row in cursor:

MapNumMapScale = row[0] + "-" + str(row[1])

MapList.append(MapNumMapScale)

UniqueMapList = set(MapList)

MapCount = 0

for Map in UniqueMapList:

if MapList.count(Map) > MapCount:

FinalMap = Map

MapCount = MapList.count(Map)

MapNumScale = FinalMap.split("-")

self.params[3].value = MapNumScale[0]

self.params[4].value = int(MapNumScale[1])

self.params[2].value = len(UniqueMapList)

return

def updateParameters(self):

# Modify parameter values and properties.

# This gets called each time a parameter is modified, before

# standard validation.

TaxlotLineLyr = "Taxlot\_Lines"

self.params = arcpy.GetParameterInfo()

aprx = arcpy.mp.ArcGISProject("CURRENT")

editMap = aprx.activeMap

TaxlotLines = editMap.listLayers(TaxlotLineLyr)[0]

if self.params[4].value != None and self.params[0].value != None:

MapScaleSuf = str(int(self.params[4].value / 12))

if self.params[4].value == "2000":

self.params[6].value = "Anno" + MapScaleSuf + "Scale"

else:

self.params[6].value = "Anno0" + MapScaleSuf + "Scale"

self.params[5].value = "DD" + self.params[0].value + MapScaleSuf

Annolyr = editMap.listLayers(self.params[6].value)[0]

Annolyr.visible = False

for lblClass in TaxlotLines.listLabelClasses():

lblClass.visible = False

lblClass = TaxlotLines.listLabelClasses(self.params[5].value)[0]

lblClass.visible = True

TaxlotLines.showLabels = True

TaxlotLines.visible = True

if self.params[9].altered and self.params[9].value:

self.params[0].value = None

self.params[3].value = None

self.params[4].value = None

self.params[5].value = None

self.params[6].value = None

self.params[9].value = False

for lblClass in TaxlotLines.listLabelClasses():

lblClass.visible = False

ToolValidator.\_\_init\_\_(self)

return

Once the features are selected the user can then run the conversion python script as follows:

# AddDimAnno.py

#

# Creates dimension annotation from new TaxlotLine Labels

#

# Uses TiledLabelsToAnnotation command which is strange

#

#

# Dean - Fall 2018

import arcpy, os

AnnoStyle = arcpy.GetParameterAsText(0)

TaxlotLinesSelected = arcpy.GetParameterAsText(1)

MapIndexCount = arcpy.GetParameterAsText(2)

MapNumber = arcpy.GetParameterAsText(3)

MapScale = arcpy.GetParameterAsText(4)

LabelClass = arcpy.GetParameterAsText(5)

AnnoLayer = arcpy.GetParameterAsText(6)

TaxlotLinesLayer = arcpy.GetParameterAsText(7)

MapIndexLayer = arcpy.GetParameterAsText(8)

# ---Setup Get Paths ------------------------

thisProject = arcpy.mp.ArcGISProject("CURRENT")

Map = thisProject.activeMap

MapIndexLyr = Map.listLayers(MapIndexLayer)[0]

TaxlotLineLyr = Map.listLayers(TaxlotLinesLayer)[0]

AnnoLyr = Map.listLayers(AnnoLayer)[0]

ScriptPath = os.getcwd()

OutGDB = ScriptPath[:-10] + "\\OrMapPFTemplate.gdb"

WorkGDB = ScriptPath[:-10] + "\\Default.gdb"

arcpy.AddMessage (ScriptPath)

arcpy.AddMessage (WorkGDB)

# -------- Set Temporary anno layers

AnnoF = "Anno" + str(MapScale)

GroupAnno = "GroupAnno"

AnnoFPth = WorkGDB + "\\Taxlot\_lines" + AnnoF

AnnoFLyr = "Taxlot\_lines" + AnnoF

annoID = 9

arcpy.AddMessage ("AnnoF: " + AnnoF)

# delete Temp layers

arcpy.Delete\_management (GroupAnno)

arcpy.Delete\_management (AnnoFLyr)

arcpy.Delete\_management(AnnoFPth)

# Make Anno and append to label class and calc symbolID to 0 (it sets symbolID for each label class you have).

arcpy.cartography.ConvertLabelsToAnnotation('Map', 1200, WorkGDB, AnnoF, 'default', 'ONLY\_PLACED', 'NO\_REQUIRE\_ID', 'STANDARD', '', '', GroupAnno, 'SINGLE\_LAYER', TaxlotLineLyr, '', '')

arcpy.CalculateField\_management(AnnoFPth, "SymbolID",0, "PYTHON3")

arcpy.Append\_management(AnnoFLyr, AnnoLyr, "NO\_TEST")

arcpy.AddMessage ("TempMade: " + AnnoFLyr)

# select and update newly added anno

arcpy.SelectLayerByAttribute\_management(AnnoLyr, "NEW\_SELECTION", "SymbolID = 0")

arcpy.CalculateField\_management(AnnoLyr, "SymbolID",annoID, "PYTHON3")

arcpy.CalculateField\_management(AnnoLyr, "AnnotationClassID",annoID, "PYTHON3")

arcpy.CalculateField\_management(AnnoLyr, "MapNumber","'" + MapNumber + "'", "PYTHON3")

# End stuff - delete temps

arcpy.Delete\_management (GroupAnno)

arcpy.Delete\_management (AnnoFLyr)

arcpy.Delete\_management(AnnoFPth)

This all seems to work fine.