



Lines & Polygons A Love Story

2023 OKSCAUG Edmond Meeting

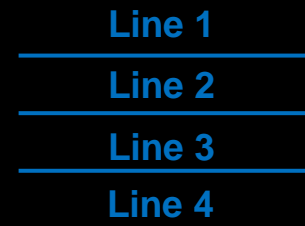
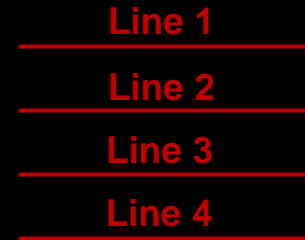
Joel A. Foster

GIS Coordinator

Canadian County Assessor's Office

Our story begins...

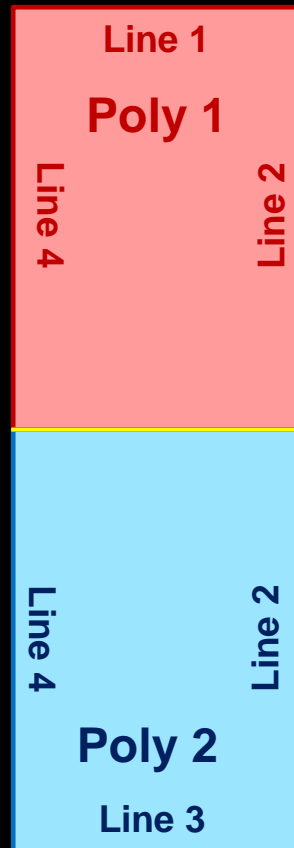
ArcMap Data Model



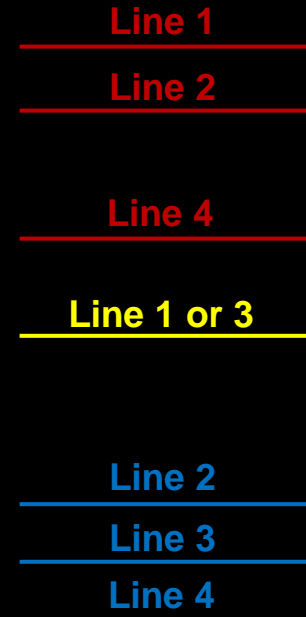
1 Polygon ↔ Many Lines

Relationship Lost...

ArcGIS Pro Data Model



Line 1 or 3



Many Polygons ↔ Many Lines

Relationship Found?

Many to Many Relationship Class



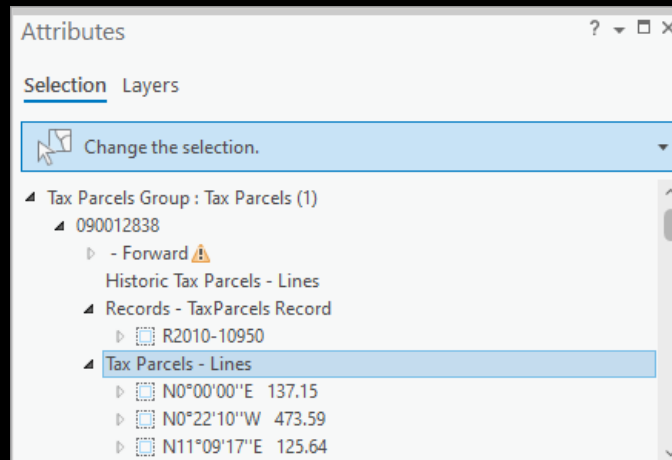
Tax_Parcels_To_Tax_Parcels_Lines



Relationship Found?

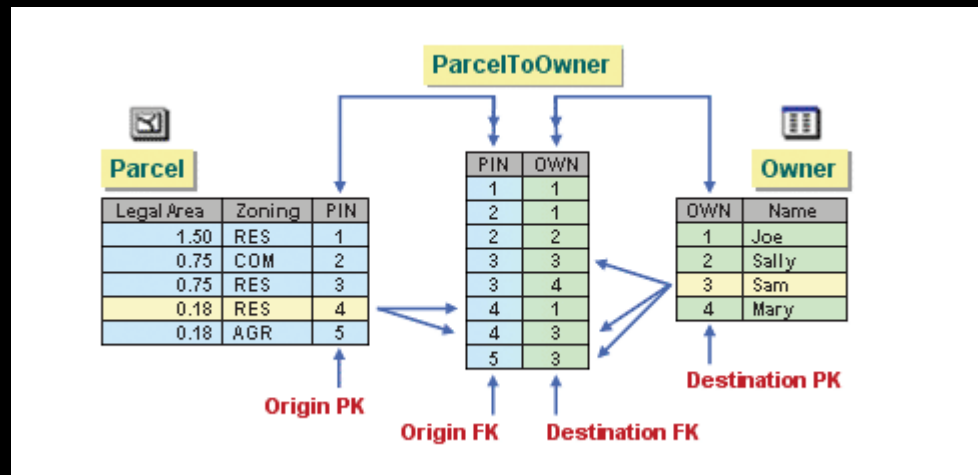
Relationship Classes Allow...

- Show related features in Attribute Pane
- Select related features on map
- Additional attributes if needed



Many to Many Relationship Class

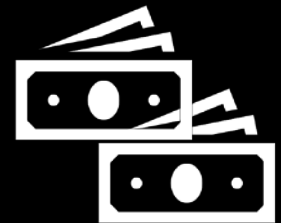
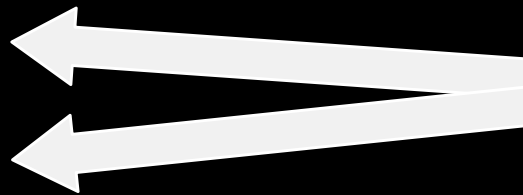
Many to Many requires an intermediate table



Many to Many Relationships

Example-

Invoices and Payments



Populating the Relationship Class Table

- Table to Relationship Class Geoprocessing tool

The screenshot shows the 'Table To Relationship Class' tool window in ArcGIS. The 'Parameters' tab is active, displaying various configuration options for creating a relationship class from a table. The tool is titled 'Table To Relationship Class' and includes a 'Parameters' section with the following fields:

- Origin Table:** A dropdown menu for selecting the source table.
- Destination Table:** A dropdown menu for selecting the destination table.
- Output Relationship Class:** A text field for specifying the output relationship class name.
- Relationship Type:** A dropdown menu set to 'Simple'.
- Forward Path Label:** A text field for specifying the forward path label.
- Backward Path Label:** A text field for specifying the backward path label.
- Message Direction:** A dropdown menu set to 'None (no messages propagated)'.
- Cardinality:** A dropdown menu set to 'One to one (1:1)'.
- Relationship Table:** A dropdown menu for selecting the relationship table.
- Attribute Fields:** A text field for specifying attribute fields.
- Origin Primary Key:** A text field for specifying the origin primary key.
- Origin Foreign Key:** A text field for specifying the origin foreign key.
- Destination Primary Key:** A text field for specifying the destination primary key.
- Destination Foreign Key:** A text field for specifying the destination foreign key.

At the bottom right, there is a 'Run' button with a play icon. Below the tool window, a row of buttons is visible: 'Geop...', 'Create...', 'Modif...', 'Attrib...', 'Label...', 'Elem...', 'Locate', and 'Mana...'.

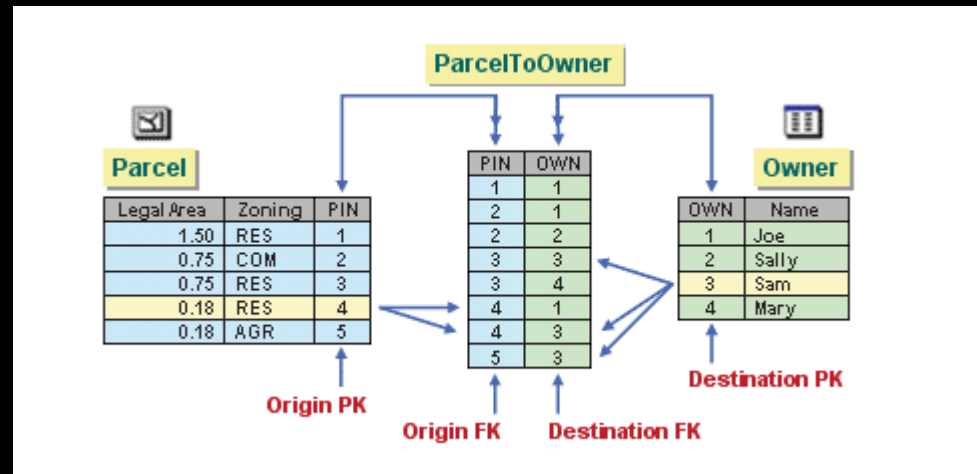
Populating the Relationship Class Table

- Manual Editing using Attributes Pane
 1. Select polygon and lines you want to relate
 2. Choose "Add Selected To Relationship" in Attributes Pane



Relationship not Quite Found...

Populating the intermediate table



When the intermediate table is created, only the fields are generated for you. ArcGIS does not know which origin objects are associated with which destination objects, so you must manually create the rows in the table. Populating this table is the most time-consuming part of setting up the relationship.

Another Way

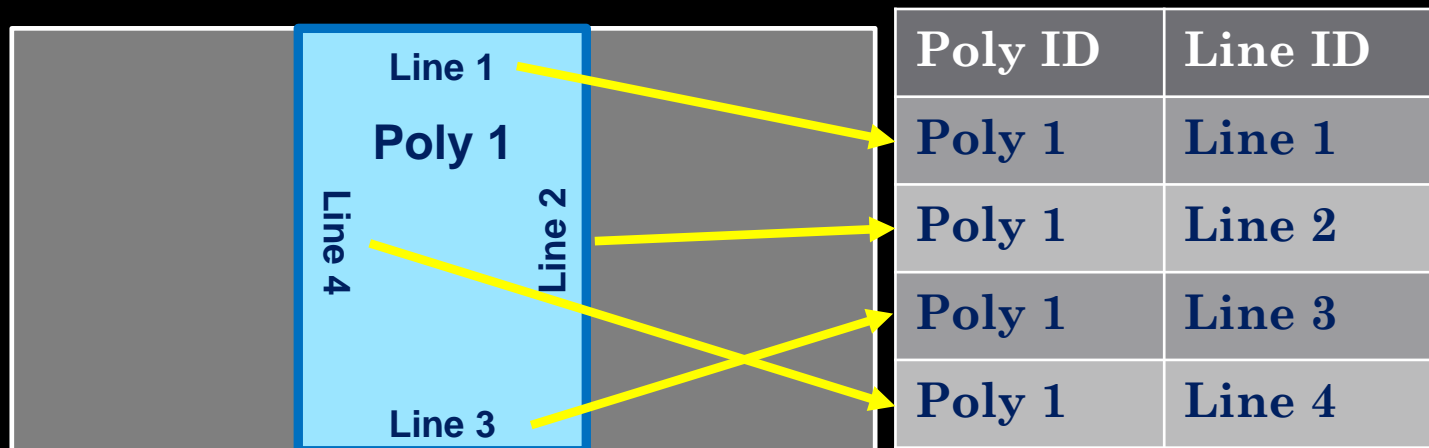
Populating the intermediate
table...programmatically

  Tax_Parcels_To_Tax_Parcels_Lines



Python Table Population

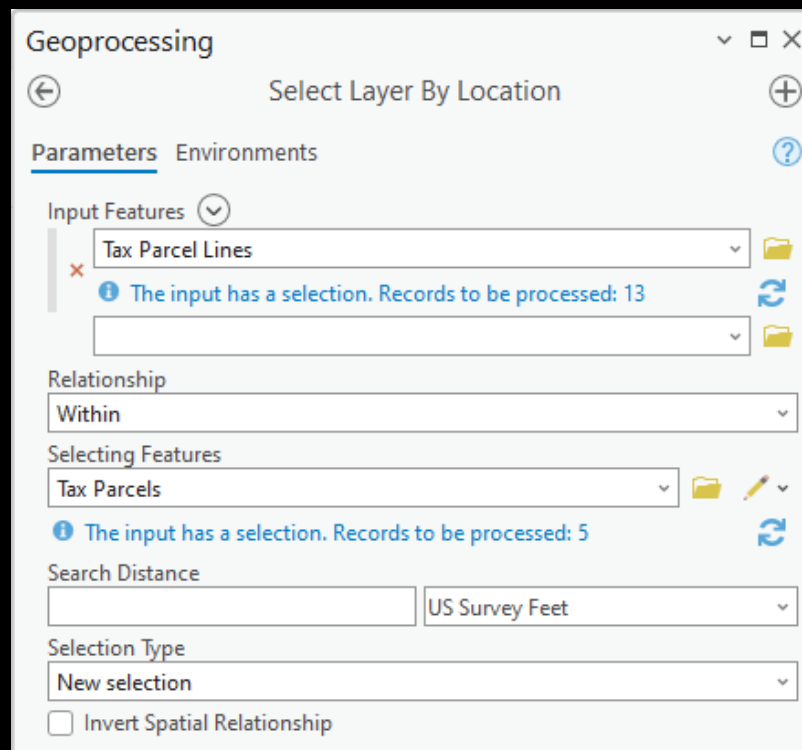
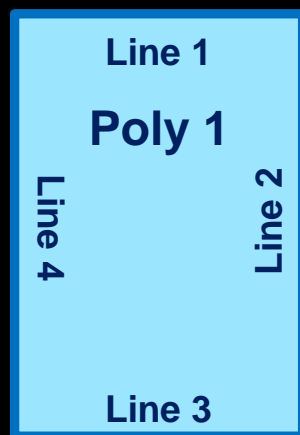
1. Find the lines that define the boundaries of the polygon
2. Insert a row in table for each line with the line's unique ID and the poly's unique ID



Find the Lines

Select By Location

- Select lines that are “within” the polygon



Find the Lines



**ArcPy Select By
Location Tools**



**ArcPy Geometry
Objects**

Python Geometry Comparison

1. Bring in the polygon or line's Shape field (ArcPy geometry object) with a cursor
2. Read the attributes and coordinates into custom python objects

```
for line in arcpy.da.SearchCursor(possible_lines, ("GlobalID", "CreatedByRecord", "SHAPE@")):
    #Create empty list for line X,Y coordinates
    line_xy_list = []
    for part in line[2]:
        for pnt in part:
            if pnt:
                line_xy_list.append([pnt.X,pnt.Y])
    new_line = line_feature(line[0],line[1],line[2],line_xy_list)
    lineList.append(new_line)
```




Python Geometry Comparison

Compare parcel geometry object with each possible line geometry using the geometry “within” method

Replace...

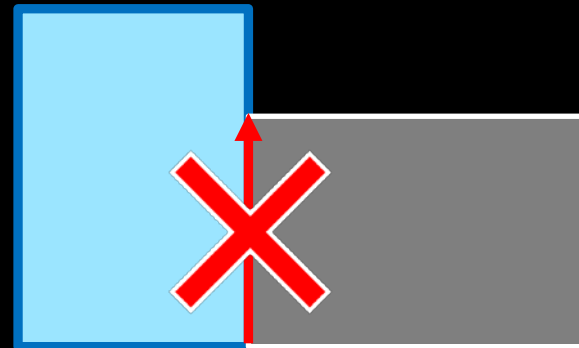
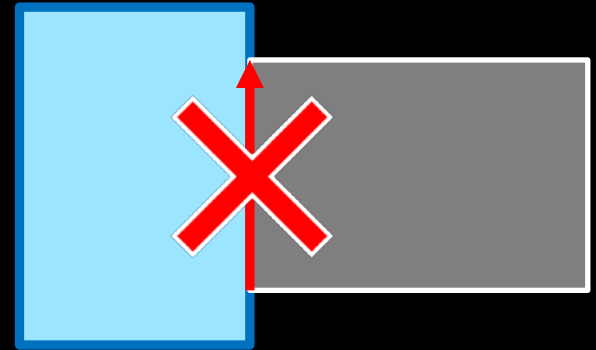
```
for parcel in parcelList:
    ##Select only that parcel to run next analysis
    parcelSelect = arcpy.management.SelectLayerByAttribute(inputParcelLayer,'NEW_SELECTION',"GlobalID = '" + str(parcel[0]) + "'")
    ##Select lines by location that share a boundary with the selected parcel
    parcelLineBoundarySelect = arcpy.management.SelectLayerByLocation(inputParcelLineLayer,'SHARE_A_LINE_SEGMENT_WITH',parcelSelect,0,'NEW_SELECTION')
    ##Select from previous selection to narrow results (not sure if this is necessary)
    parcelLineWithinSelect = arcpy.management.SelectLayerByLocation(parcelLineBoundarySelect,'WITHIN',parcelSelect,0,'SUBSET_SELECTION')
    ##Select from previous selections only those lines that have the same "Created By Record"
```

with...

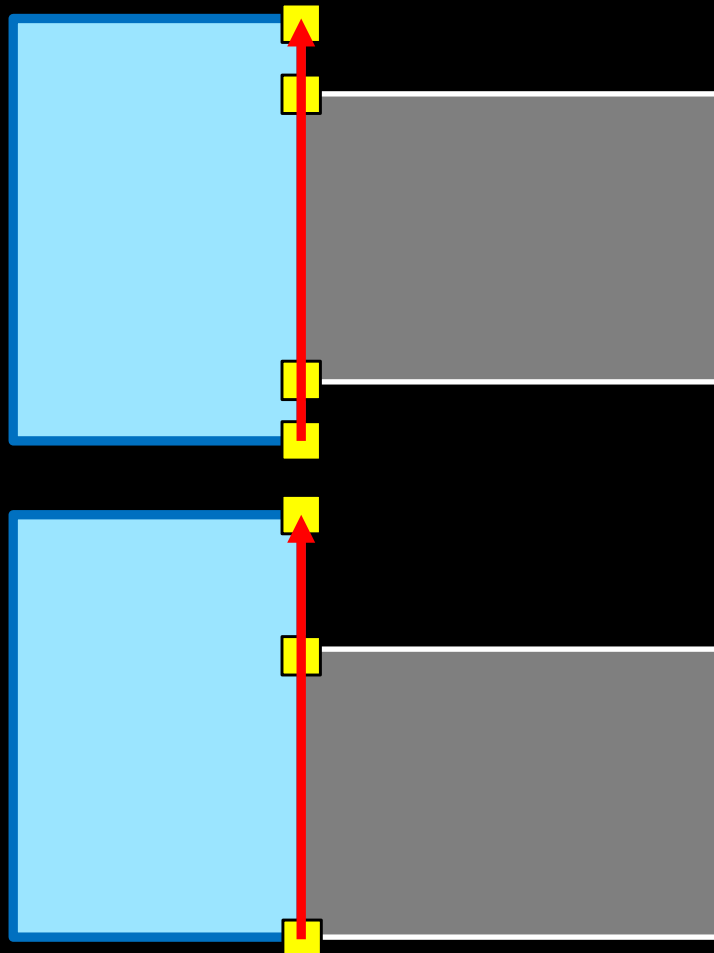
```
for parcel in parcelList:
    arcpy.AddMessage('Parcel {0}'.format(parcel.oid))
    #Empty list to contain lines that make up the parcel
    possibleLineList = []
    #Check each line to see if it is within the parcel
    for line in lineList:
        if line.geom.within(parcel.geom,'BOUNDARY'):
```

\$\$\$ 1000% TIME SAVINGS \$\$\$

At least so far as I have tested...



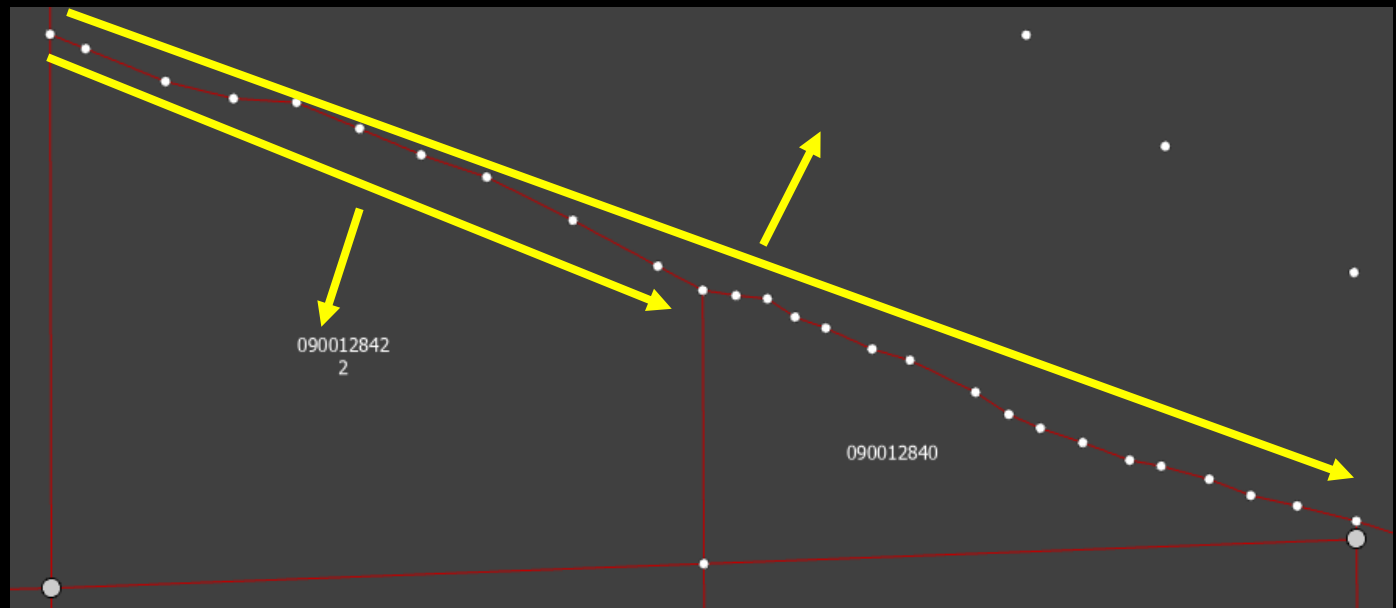
Remove the Imposters



1. Check any line that has more than two vertices for all overlapping lines
2. Keep the line that has the most vertices because it covers more of the boundary

Remove the Imposters

Works for valid >2 point lines too

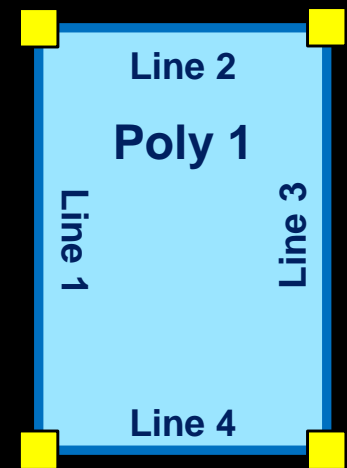


Adding Line Sequencing Attribute

Use python to find the order of the lines and fill in a sequence attribute

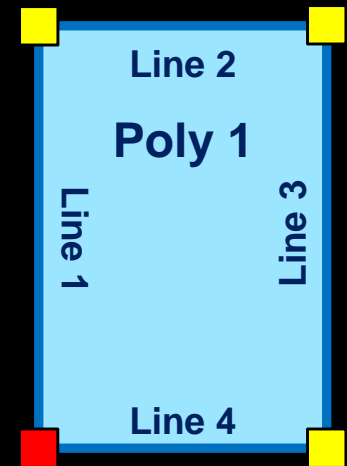
Use ONLY line start points and end points

Leave NULLs if a there is an error in data construction



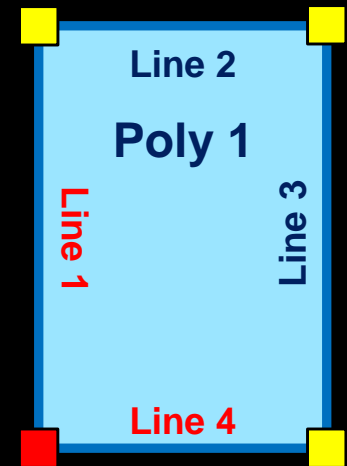
Adding Line Sequencing Attribute

1. Find SW corner as starting point



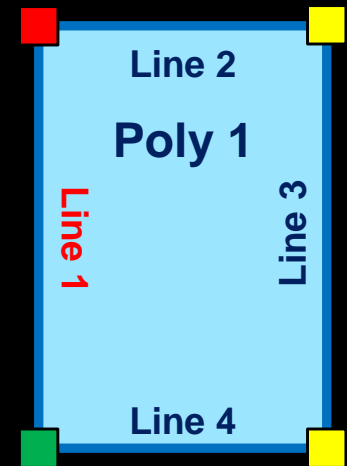
Adding Line Sequencing Attribute

1. Find SW corner as starting point
2. Find the lines that share starting point



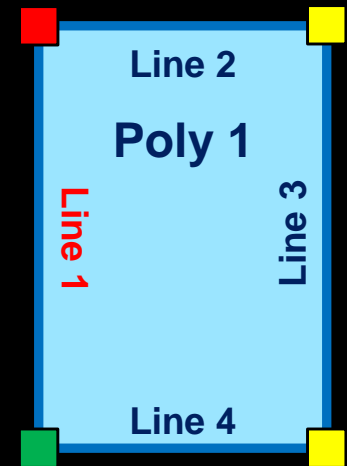
Adding Line Sequencing Attribute

1. Find SW corner as starting point
2. Find the lines that share starting point
3. Pick line with higher "Y" to go clockwise and save the point at the other end of the line



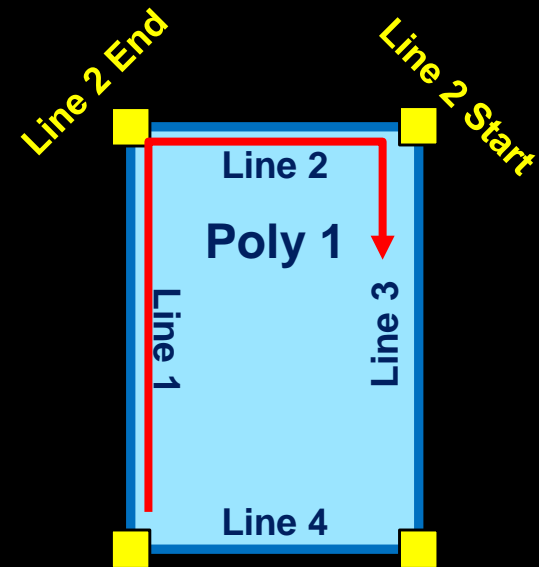
Adding Line Sequencing Attribute

1. Find SW corner as starting point
2. Find the lines that share starting point
3. Pick line with higher "Y" to go clockwise and save the point at the other end of the line
4. Continue until all points are reviewed



Adding "Reversed" Attribute

As lines are sequenced, if the line "end" is the matching point in the next line, set the line reversed attribute to "True" or "1"





Adding “Part” and “Interior Ring” Attributes

Use nested custom Python objects

Parcel Object 1

Part Object 1

Line Objects 1, 2, 3, 4

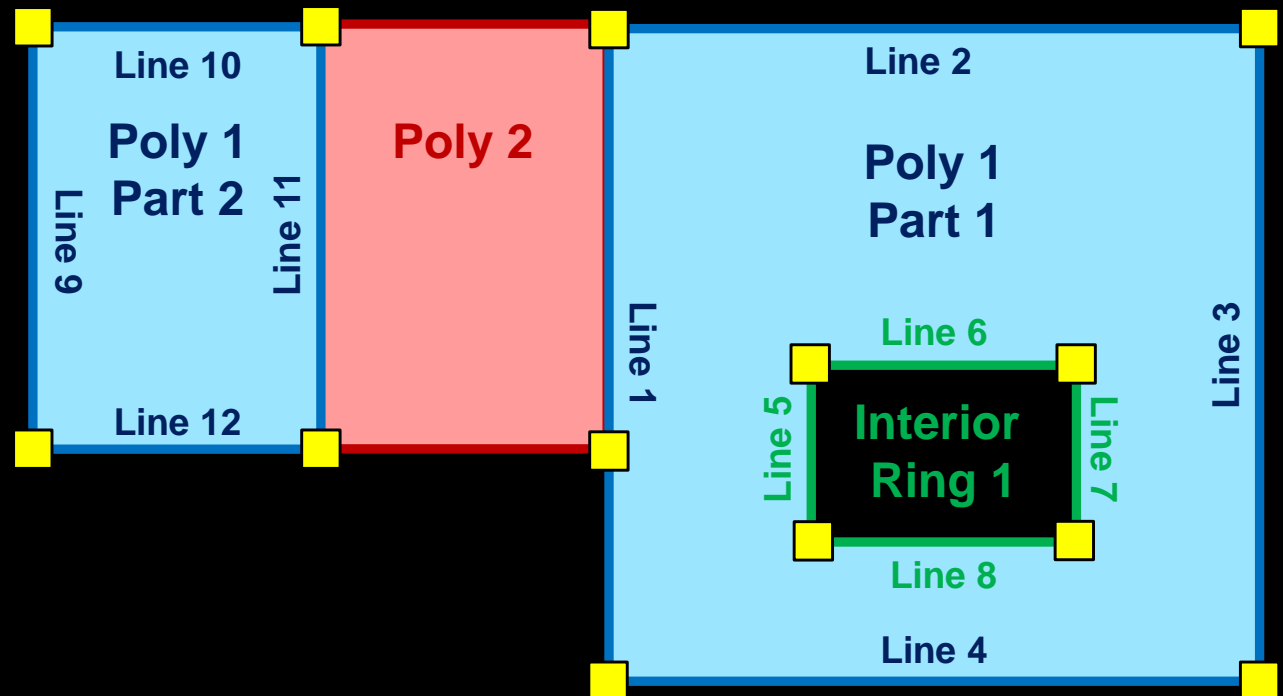
Interior Ring Object 1

Line Objects 5, 6, 7, 8

Part Object 2

Line Objects 9, 10, 11, 12

Adding "Part" and "Interior Ring" Attributes



Companion Tool to Re-order Lines

If the SW corner is not the starting point, take sequenced lines and rearrange them with the correct sequence.

```
with arcpy.da.UpdateCursor(inputRelationshipClassTableView, ('line_sequence', 'reverse_direction')) as updateCursor:
    for row in updateCursor:
        # Changing from clockwise to counter-clockwise
        if reverseDirection:
            # Change the "reverse" attribute to the opposite if changing from clockwise to counter-clockwise
            if row[rev] == 0:
                row[rev] = 1
            else:
                row[rev] = 0
            # Change the sequence based on the input starting line sequence
            if row[seq] <= inputStartLineSeq:
                row[seq] = (inputStartLineSeq - row[seq]) + 1
            else:
                row[seq] = ((rel_table_count + 1) - row[seq]) + inputStartLineSeq
        # Do NOT change the direction from clockwise to counter-clockwise
        else:
            # Change the sequence based on the input starting line sequence
            if row[seq] >= inputStartLineSeq:
                row[seq] = (row[seq] - inputStartLineSeq) + 1
            else:
                row[seq] = ((rel_table_count + 1) - inputStartLineSeq) + row[seq]
        # Send the updated row back to the table
        updateCursor.updateRow(row)
```

Lines & Polygons Reunited

1. Many-to-Many Relationship Class handles keeping the two feature classes related, with additional attributes if needed
2. Python speeds up & automates populating the relationship class making maintenance easier





Using Final Relationship Class Table

- Display the lines that make up a parcel in the correct order in Pro or Enterprise Apps
- Auto-generate legal descriptions from the parcel and line data
- Automatically re-draw a parcel from original legal description

Final Thoughts

- Find a way to add in a “Point-of-Beginning Connection” line to complete a full legal description
- Correctly deal with purposely “stacked” lines
- Get user input for a starting line
 - Tool may then have to run on one parcel at a time and “batch” the tool for multiple parcels
 - How would it deal with parts and interior rings?





Lines & Polygons A Love Story

2023 OKSCAUG Edmond Meeting

Joel Foster

GIS Coordinator

Canadian County Assessor's Office

(405) 295-6331

foster@canadiancounty.org