Learn the OpenAccess API

Using Python

©Silicon Integration Initiative

Initial Contribution By

James Masters
Intel - 2013
Updates & Additions
by Silicon Integration Initiative - 2020

Section 12: OA Extensions, Polygon Operators

- Additional functionality can be made available to OA within the Si2 standards body as an OA “extension”
- The following extensions exist:
  - oaDebug: Framework to dump information about an oaDesign
  - oaxPop: Polygon operations (Booleans)
- We will cover the polygon operations here (oaxPop)
- We will also look at the new oaxPop 2.0 Connectivity Extraction

Including the oaxPop Module

- A scripting interface is available for oaxPop in all of the oaScript languages:
  - Perl

```
use Pop;
```

  - Python

```
import oa
import pop
```

  - Ruby

```
require 'pop'
```

  - Tcl

```
package require pop
```

Lab 12.1: Testing oaxPop

- Goal – Ensure that the oaxPop library is loaded properly

- Create a script to just load the oaxPop library
  - See previous slide as an example
  - Test the script to ensure that the oaxPop library loads

oaxPop Classes

- oaxPop has the following classes:
  - `FigSet90: represents 90--degree shapes (best performance)`
  - `FigSet45: represents 45--degree  shapes (next best performance)`
  - `FigSetAny: represents any degree shapes (good performance)`
- Example below shows how an oaPointArray can be used to populate a FigSet90 followed by an “and” operation

```
a = pop.FigSet90()
b = pop.FigSet90()

a.append( [[0,0], [0,100], [100,100], [100,0]] )
b.append( [[90,90], [90,190], [190,190], [190,90]] )

c = a & b
```

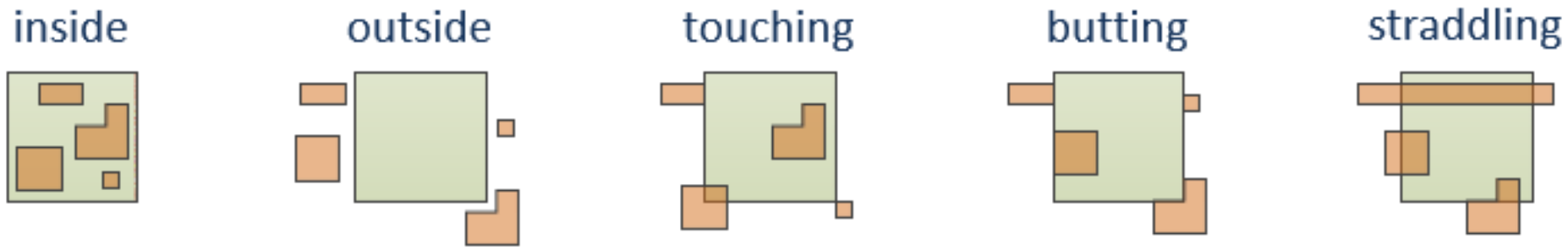
Available Operations

- Standard operators: OR (**+**, **|**); AND (*, **&**); NOT (**--**); XOR (**^**)
  - `Examples: a & b  a | b  a - b  a ^ b`
  - Note: operators & and | are not available in Perl at this time
- Resize operators – uniform or for top, bot, leo, right (90 only)

```
a.resize(100)
a.resize(50, 50, 10, 20)   # FigSet90 only
```

- Unified Layer Model (ULM) descriptions

```
a.touching(b)
a.inside(b)
a.outside(b)
...
```



Lab 12.2: Basic Boolean Operation

- Goal - Become familiar with simple layer operations

- Create a script to load the oaxPop library and:
  - Create two rectangles in FigSet90 named “a” and “b” (you will need to make the boxes using an oaPointArray)

```
a = pop.FigSet90()
a.append( ... )  # pass an oaPointArray
```

  - Perform an “and” opera,on (& or *) – note & is not available in Perl
  - Check to see if anything resulted and if so then print “overlapping” to the screen
    - You can use the isEmpty() func,on to determine if the results have contents or not

```
c.isEmpty()
```

Compare your script to labs/12.2/boolean.py

Importing/Exporting OA Data

- OA data can be directly imported into an FigSet* object
  - All shapes on an oaLPPHeader (not available in this training)

```
a.append(lpph.getShapes())
```

  - Individual shape--by--shape

```
for shape in lpph.getShapes():
    a.append(shape)
```

  - Some other OA--based source which is converted into an oaPointArray and inserted

```
bnd    =  oa.oaPRBoundary.find(block)
bnd_fs   = pop.FigSet90()  bnd_fs.append(bnd.getPoints())
```

- FigSet* data can be committed to OA

```
# m2 is layer num,   drw   is purpose num  result.commit(block,
  m2, drw)
```

Lab 12.3: Booleans on OA Data

- Goal - Become familiar with importing and exporting OA data for polygon operations

- Write a script to:
  - Open existing “res” design **in “a” mode (append)**
  - Put all m1 shapes into a FigSet90 object

    ```
    a.append(shape)
    ```
  - Put all cont objects into another FigSet90 object
  - Get the result of m1 NOT cont (`a - b`)
  - Put the result into the m2/drawing layer

    ```
    result.commit(block, m2, drw)
    ```
  - Save result as “pop” view

Compare your script to labs/12.3/pop_Boolean.py

Connectivity Extraction

- In version 2.0 of Polygon Operators (“oaxPop”) new capability was added
  - 14 Conn* classes were added to enable connectivity extraction
- ConnLayer objects are defined using Layers and Purposes
- ConnExt90.connect() connects the layer objects into layer sets
- A ConnExt90 figure set is created with shapes from the design

  `set = ConnExt90()`

  `set.append` adds the shapes (text too)
- The layer sets are used to extract the contents of the figure set

  `ConnExt90.extract(set)`
- This modifies the figure set grouping the shapes into ConnNodes
  - Each ConnNode contains all the connected shapes for that node

Lab 12.4: Connectivity Extraction

- Goal – Learn how the connectivity extraction works
- Run this script (buildDesign.py) to build a design with some shapes and text (test_connect)
- Write a script to:

  extract all of the connected nodes from the design generated by the script above

  Extract the Connectivity from all the shapes in the design

  Save the extracted ConnNode objects to a new design for viewing (test_extract), only if they have a net name

Compare your script to labs/12.4/connect.py

Section 12 Summary

- Learned what Polygon Operators (oaxPop) Extension is
- Using the oaxPop libraries
- What is a Figure Set and how some of the operations work
- How to use oaxPop with OpenAccess
- And the new connectivity extraction in oaxPop 2.0!

Section 12: Other Areas of Interest

- Not all of the concepts could be included in this training due to time constraints
- Other topics of interest include:
  - **oaProp:** Add a property on an OA object
  - **oaVia/oaViaDef:** Define and place vias in a design
  - **oaAppDef:** Create and access application definitions (application extensions on OA) packed into objects
  - **oaFigGroup:** Create group of figures that move as one entity
  - **oaLine:** Shape used to draw wires on schematics
  - Trapping OA exceptions in oaScript
  - And more!

Silicon Integration Initiative

www.si2.org

For details contact Marshall Tiner

Director of Production Standards

mtiner@si2.org