Computational Design + Fabrication: Digifab Intro

Austin Buchan, Duncan Haldane, Jonathan Bachrach

EECS UC Berkeley

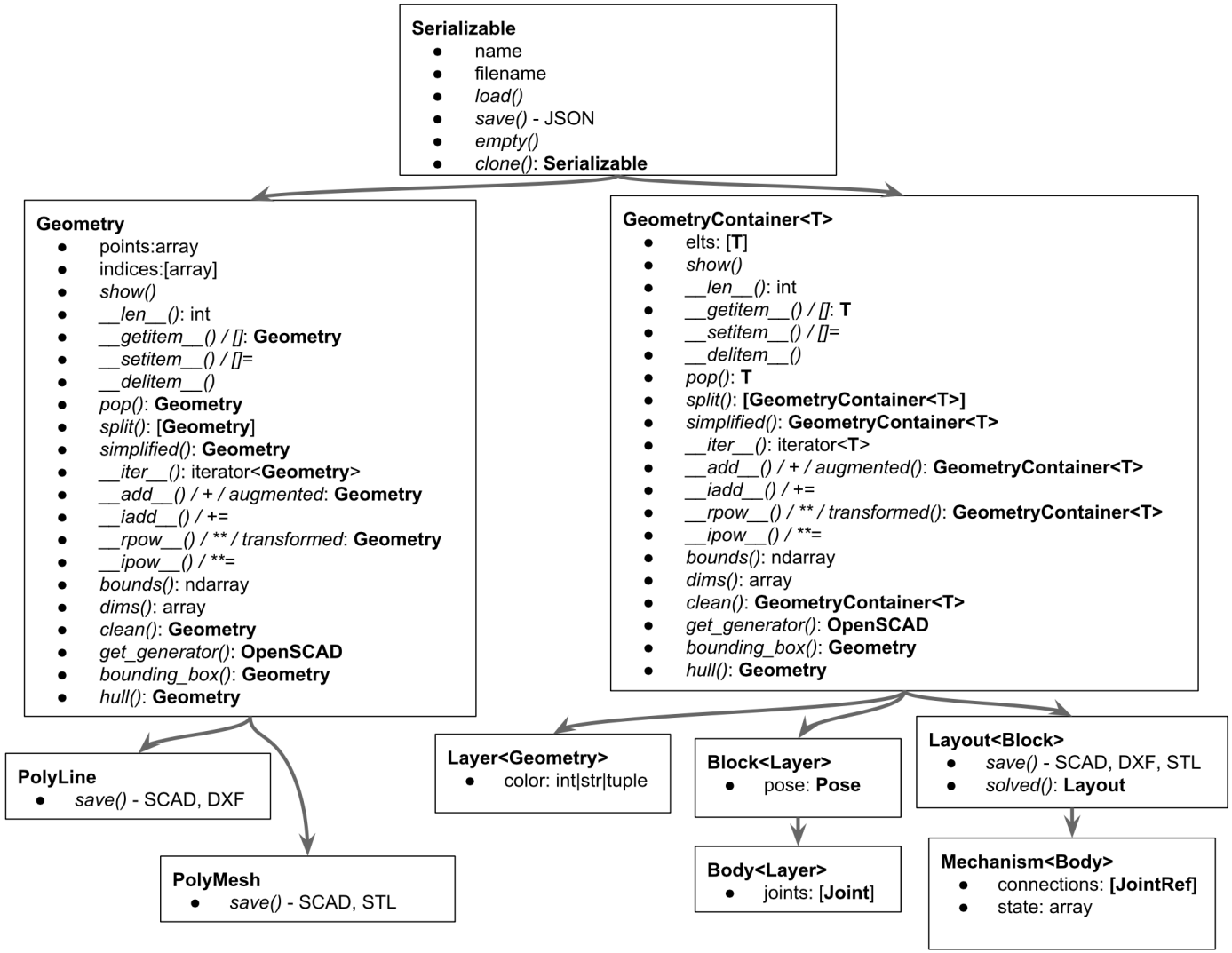
September 4, 2015

- API about solidpython
- manufacturing API
- better python interface
- mechanism library

Overall Hierarchy

- Serializable – Dump Support
- Geometry – points + indices
- PolyLine – collection of line segments
- PolyMesh – collection of meshes
- GeometryContainer – collection of geometry
- Layout – placement of geometry
- Layer – geometry layer
- Block – placed geometry
- Mechanism – ...
- Body – ...

Digifab Hierarchy



PolyLine

- points – all points
- indices – point indices organized into paths
- len – number of paths
- split – breaking paths apart into separate polylines
- add del – as + -
- union intersection difference – as boolean ops
- bounds()
- dims()
- bounding_box()
- hull()
- show() – plotting
- save(filename) – scad, dxf

```
PolyLine(points=[[0,100],[100,100],[100,0],[0,100]])
PolyLine(points=[[0,100],[100,100],[100,0]], indices[[0,1,2,0]])
```

PolyLine Points and Indices


```
pl = PolyLine(points=[[0,100],[100,100],[100,0],[0,100]])
ps = pl.points  -> [[0,100],[100,100],[100,0],[0,100]]
is = pl.indices -> [[0,1,2,3]]
```

```
pl  = PolyLine(points=[[0,100],[100,100],[100,0],[0,100]])
pl += PolyLine(points=[[0,50],[50,50],[0,50]])
ps  = pl.points  -> [[0,100],[100,100],[100,0],[0,100],[0,50],[50,50],[0,50]]
is  = pl.indices -> [[0,1,2,3],[4,5,6]]
```

- abbreviations

```
(x,y,r) * polyline(...)
```

```
0.5 * polyline(...)
```

- reading to/from dxf,scad

```
polyline(...).save('file.scad')
```

```
polyline(file='file.dxf')
```

- show()
- creates python plot window
- close window to continue

```
p.show()
```

Building Up PolyLines

- keep adding to a polyline to add more paths

```
p  = PolyLine(...) + PolyLine(...)
```

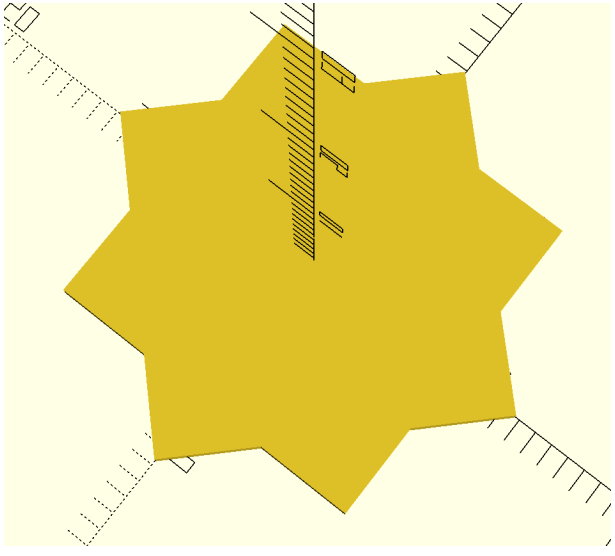
```
p  = PolyLine(...)
p += PolyLine(...)
```

- must produce polyline
- solid is a generator passed in constructor
- one way

```
PolyLine(generator=solid.square(100))
```

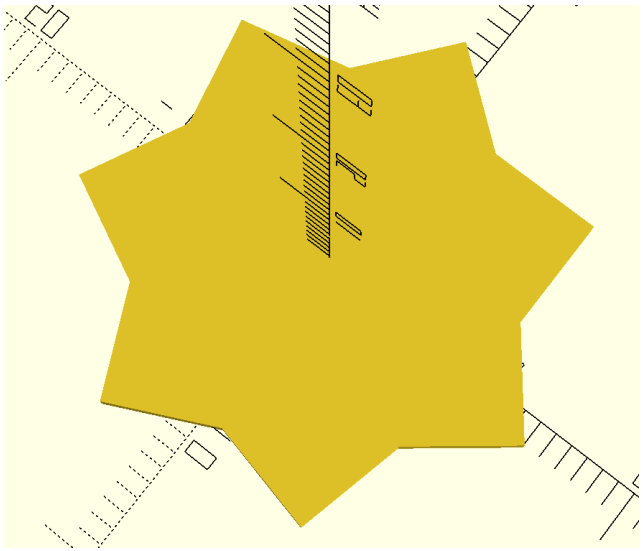
```
sq = solid.square(100)
ci = solid.circle(200)
sc = ci - sq
PolyLine(generator=sc)
```

```
sqs = union()()
for i in range(8) :
 angle = 360 * i / 8.0
 sqs += rotate([0, 0, angle])( square(9) )
PolyLine(generator=sqs).save('sqs.scad')
```



```
def spin_squares (n, w) :
  sqs = union()()
  for i in range(n) :
    angle = 360 * i / float(n)
    sqs += rotate([0, 0, angle])( square(w) )
  return sqs

PolyLine(generator=spin_squares(7, 10)).save('sqs.scad')
```



Properties

- consistent api
- bounds, dims, bounding_box, hull

```
bounds(polyline(...)) -> [low, high]
bounding_box(polyline(...)) -> [width, height]
dims(polyline(...)) -> polyline
hull(polyline(...)) -> polyline
```

- give us your github username if you haven't already
- accept invitation
- follow instructions in lab_0 README
- usb keys
- due by next thursday 9/10