Computational Graphics: Lecture 15

Alberto Paoluzzi

Thu, Apr 14, 2016

Outline: geometric programming tricks

1. Some tricks in `LarLib`

2. Some tricks in `pyPLaSM`

Some tricks in LarLib

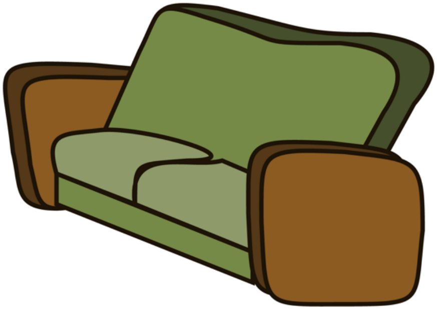
SVG input: http://cvdlab.github.io/svg2lines/ service

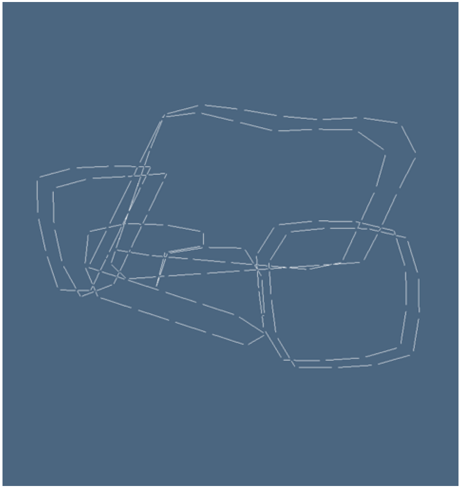
```
from larlib import *
""" Input of SVG drawing via `http://cvdlab.github.io/svg2lines/` service """
# drag-and-drop the ".svg" file in the service window,
# and save a ".lines" file

lines = lines2lines("couch.lines")
V,FV,EV,polygons = larFromLines(lines)
VIEW(EXPLODE(1.2,1.2,1.2)(MKPOLS((V,EV))))
VIEW(EXPLODE(1.2,1.2,1.2)(MKPOLS((V,FV,EV))))
```

SVG input: http://cvdlab.github.io/svg2lines/ service

```
from larlib import *
""" Input of SVG drawing via `http://cvdlab.github.io/svg2lines/` service """
# drag-and-drop the ".svg" file in the service window,
# and save a ".lines" file

lines = lines2lines("scala.lines")
V,FV,EV,polygons = larFromLines(lines)
VIEW(EXPLODE(1.2,1.2,1.2)(MKPOLS((V,EV))))
VIEW(EXPLODE(1.2,1.2,1.2)(MKFACES((V,FV,EV))))

VV = AA(LIST)(range(len(V)))
submodel = STRUCT(MKPOLS((V,EV)))
VIEW(larModelNumbering(1,1,1)(V,[VV,EV,FV],submodel,0.1))

FV = sorted(FV,key=lambda cell: CCOMB([V[k] for k in cell])[0])
VIEW(larModelNumbering(1,1,1)(V,[VV,EV,FV],submodel,0.1))
```

Importing of SVG files via web service





Use Struct objects to apply local transformations

```
struct = Struct([ ([V[k] for k in cell],[range(len(cell))]) for cell in FV ])
scala = embedStruct(1)(struct)

print scala
print scala.body

stairs = CAT(DISTR([scala.body,t(0,0,0.02)]))

print stairs

stairs = struct2lar(Struct(stairs))
VIEW(STRUCT(MKPOLS(stairs)))
```

Viewing the cells indices of new 2-complex in 3D

```
W,FW = stairs
WW = AA(LIST)(range(len(W)))
FW = sorted(FW,key=lambda cell: CCOMB([W[k] for k in cell])[0])
submodel = SKEL_1(STRUCT(MKPOLS((W,FW))))
VIEW(larModelNumbering(1,1,1)(W,[WW,FW],submodel,0.1))
```

Boundary edges and interior edges

boundary (and coboundary) operator

$$\partial_2 : C_2 \to C_1$$

```
csrmat = larBoundary(FV,EV)
print csrmat

boundaryEdges = boundaryCells(FV,EV)
B_EV = [EV[e] for e in boundaryEdges]
I_EV = [EV[e] for e in set(range(len(EV))).difference(boundaryEdges)]
VIEW(EXPLODE(1.2,1.2,1.2)(MKPOLS((V,B_EV))))
VIEW(EXPLODE(1.2,1.2,1.2)(MKPOLS((V,I_EV))))
VIEW(STRUCT(AA(COLOR(CYAN))(MKPOLS((V,B_EV))) +
            AA(COLOR(YELLOW))(MKPOLS((V,I_EV))) ))
```

Orientation of 2-cells in 3D

```
Z,FZ = stairs
submodel = SKEL_1(STRUCT(MKPOLS((Z,FZ))))
VIEW(larModelNumbering(1,1,1)(Z,[VV,FZ],submodel,0.1))

signs = AA(SIGN)(signedSurfIntegration((Z,FZ,EV),signed=True))
FV = [cell if sign>0 else REVERSE(cell) for sign,cell in zip(s
VIEW(STRUCT(MKPOLS((Z,FZ))))
```

Some tricks in pyPLaSM

circularArc curve

```
def circularArc(angle,n):
    return MAP(lambda p: [COS(p[0]), SIN(p[0])])(INTERVALS(angle)(n))

VIEW(circularArc(PI/3,3))
```

OFFSET of 1-skeleton

```
obj = SKEL_1(CUBE(1))

VIEW(OFFSET([.1,.1,.1])(obj))
VIEW(OFFSET([.05,.05,.2])(obj))
```

OFFSET of 1-skeleton

```
obj = SKEL_1(CYLINDER([1,1])(6))

VIEW(OFFSET([.05,.05,.2])(obj))
```

OFFSET of 2-complex

```
obj = SKEL_1(CUBOID([1,1]))

obj2 = OFFSET([.1,.1,1])(obj)
```