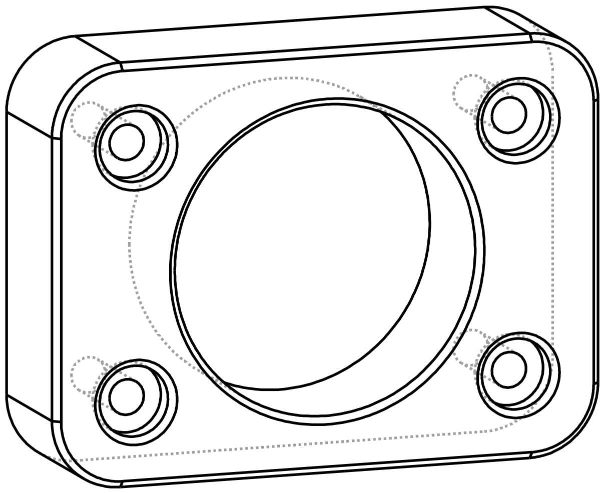
Towards CadQuery 2.0

Adam Urbańczyk

Introduction

- CadQuery is a Python module for building parametric 3D CAD models in boundary representation (B-rep)

```
import cadquery as cq

height = 40.0
width = 30.0
thickness = 10.0
diameter = 22.0
padding = 12.0
rf = 5
cbore_r1,cbore_r2, cbore_d = 2.5, 5, 2
ch = .5

result = (cq.Workplane("XY").box(height, width, thickness)
    .faces(">Z").workplane().hole(diameter)
    .faces(">Z").workplane()
    .rect(height - padding, width - padding, forConstruction=True)
    .vertices().cboreHole(cbore_r1,cbore_r2, cbore_d)
    .edges("|Z").fillet(rf).faces('>Z').chamfer(ch))
```

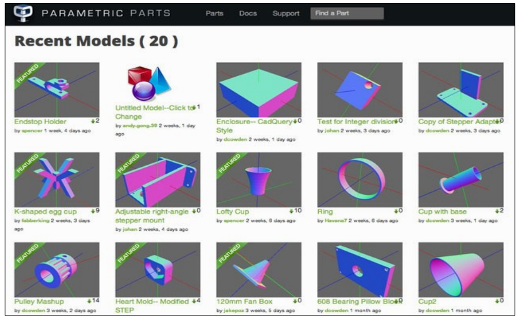


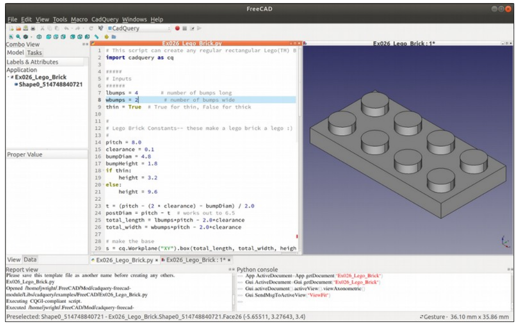
List of contributors

- Core team
  - *Dave Cowden, Jeremy Wright and Adam Urbańczyk*
- Contributors CadQuery
  - *hyOzd, bragostin, mgreminger, justbuchanan, huskier, fragmuffin, Peque, bweissinger, osterwood, moeb, asukiaaa, HLevering, westurner, Renha, armyofevilrobots, gebner, krasin, Grawp, phillipthelen, bsilvereagle, xix-xeaon*
- Contributors CQ-editor
  - *gebner, justbuchanan, jmwright*

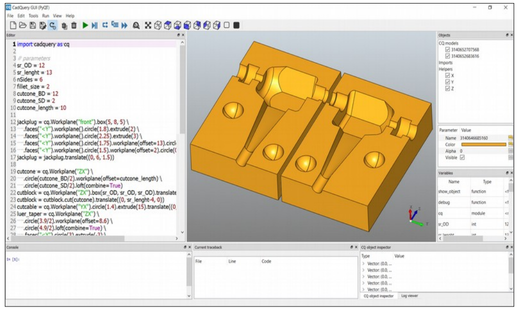
History

- Project started by Dave
  - Used in the backend of a parametric modeling website
- Jeremy wrote a FreeCAD workbench
  - https://github.com/jmwright/cadquery-freecad-module
- I joined the team last
  - Started the PythonOCC transition and CQ-editor





```
# This script can create any regular rectangular Lego(TM) B
import cadquery as cq

#####
# Inputs
######
lbumps = 4       # number of bumps long
wbumps = 2       # number of bumps wide
thin = True  # True for thin, False for thick

#
# Lego Brick Constants-- these make a lego brick a lego :)
#
pitch = 8.0
clearance = 0.1
bumpDiam = 4.8
bumpHeight = 1.8
if thin:
    height = 3.2
else:
    height = 9.6

t = (pitch - (2 * clearance) - bumpDiam) / 2.0
postDiam = pitch - t  # works out to 6.5
total_length = lbumps*pitch - 2.0*clearance
total_width = wbumps*pitch - 2.0*clearance

# make the base
s = cq.Workplane("XY").box(total_length, total_width, heigh
```



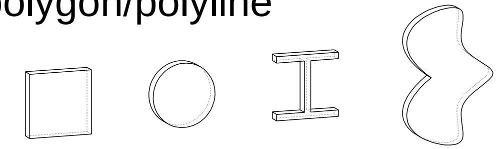
```
import cadquery as cq

# parameters
sr_OD = 12
sr_lenght = 13
nSides = 6
fillet_size = 2
cutcone_BD = 12
cutcone_SD = 2
cutcone_length = 10

jackplug = cq.Workplane("front").box(5, 8, 5) \
    .faces("<Y").workplane().circle(1.8).extrude(2) \
    .faces("<Y").workplane().circle(2.25).extrude(3) \
    .faces("<Y").workplane().circle(1.75).workplane(offset=13).circl
    .faces("<Y").workplane().circle(1.5).workplane(offset=2).circle((
jackplug = jackplug.translate((0, 6, 1.5))

cutcone = cq.Workplane("ZX") \
    .circle(cutcone_BD/2).workplane(offset=cutcone_length) \
    .circle(cutcone_SD/2).loft(combine=True)
cutblock = cq.Workplane("ZX").box(sr_OD, sr_OD, sr_OD).translate
cutblock = cutblock.cut(cutcone).translate((0, sr_lenght-4, 0))
cutcable = cq.Workplane("YX").circle(1.4).extrude(15).translate((0
luer_taper = cq.Workplane("ZX") \
    .circle(3.9/2).workplane(offset=8.6) \
    .circle(4.9/2).loft(combine=True) \
```

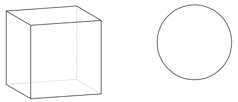
Project goals and motivation

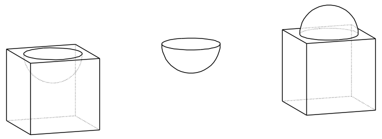
- Use standard and user-friendly programming language (Python)
- B-rep CAD kernel (OpenCascade)
- Fluent API
- Advanced modeling capabilities
- Ability to import and export STEP models
- High performance

Capabilities

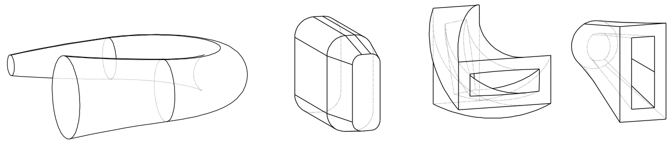
- 2D primitives
  - Rectangle, circle, polygon/polyline
  - Spline
  - Parametric curves
- 3D primitives
  - Box, sphere
- CSG operations
  - Cut
  - Intersect
  - Union
- Selectors DSL
  - Choose vertices, edges, faces, solids
  - Combine selectors logically or chain them

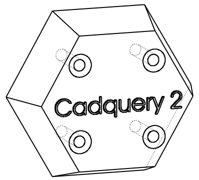






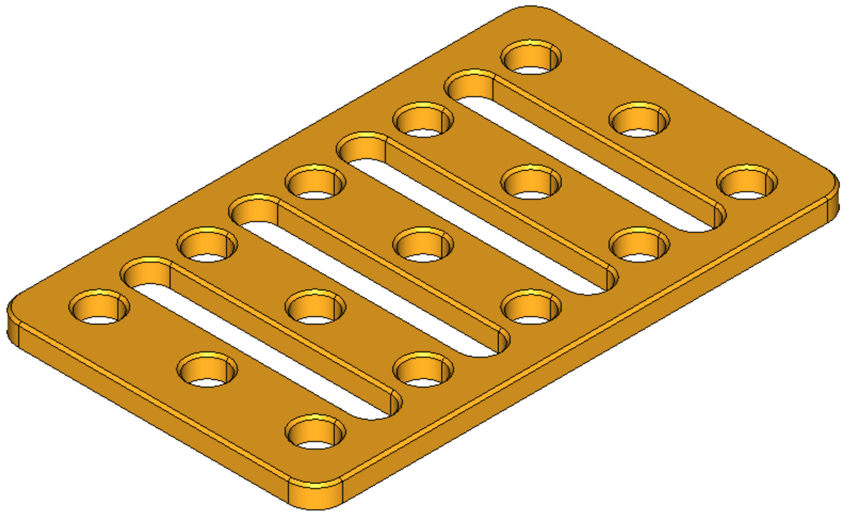
- 3D operations
  - Extrude (tapered, twisted)
  - Revolve
  - Loft
  - Shell
  - Fillet, chamfer
  - Sweep / multi-section sweep
  - 3D text
- Supported formats
  - STEP (R/W)
  - STL (W)
  - AMF (W)
  - SVG (W)





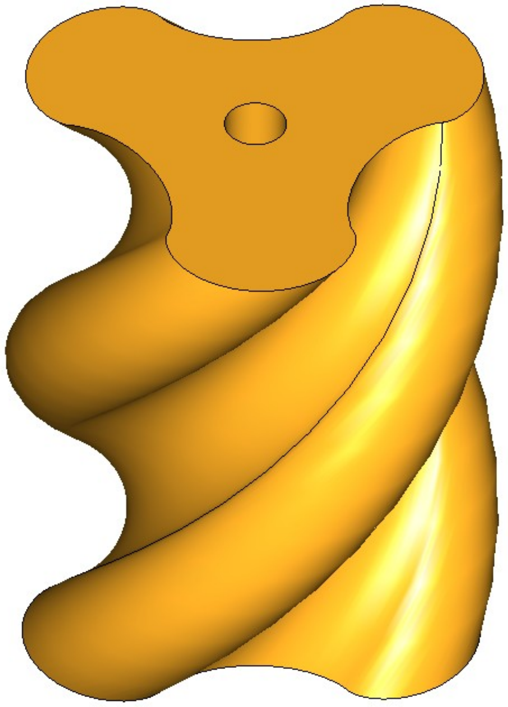
cq Example – parametric mounting plate

```
import cadquery as cq

w,d,h = 100,60,4
pitch = 20
nx,ny = 5,3
r_hole = 7
ch = .5

Plate =( cq.Workplane('XY')
         .box(w,d,h)
         .edges('|Z')
         .fillet(5)
         .faces('>Z').workplane()
         .rarray(pitch, pitch, nx,ny, True)
         .hole(r_hole)
         .rarray(pitch, pitch, nx-1,1, True)
         .slot2D(d*0.8,r_hole,90).cutThruAll()
         .faces('>Z').edges()
         .chamfer(ch) )
```

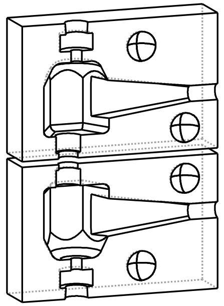
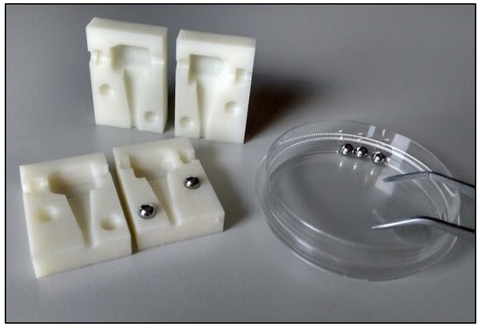


Example – complex shape

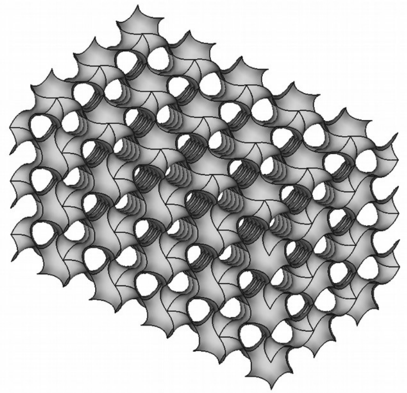
```
import cadquery as cq
from math import sin, cos,pi,floor

def hypocycloid(t,r1,r2):
    return ((r1-r2)*cos(t)+r2*cos(r1/r2*t-t),(r1-r2)*sin(t)
+r2*sin(-(r1/r2*t-t)))

def epicycloid(t,r1,r2):
    return ((r1+r2)*cos(t)-r2*cos(r1/r2*t+t),(r1+r2)*sin(t)-
r2*sin(r1/r2*t+t))

def gear(t,r1=4,r2=1):
    if (-1)**(1+floor(t/2/pi*(r1/r2))) < 0:
        return epicycloid(t,r1,r2)
    else:
        return hypocycloid(t,r1,r2)

h =20; twist = 180 ;d = 2

res = (cq.Workplane('XY')
       .parametricCurve(lambda t: gear(t*2*pi,6,1))
       .twistExtrude(20,180)
       .faces('>Z').workplane()
       .hole(d) )
```



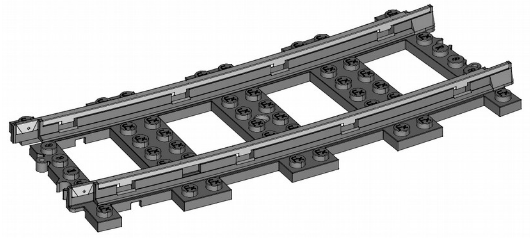
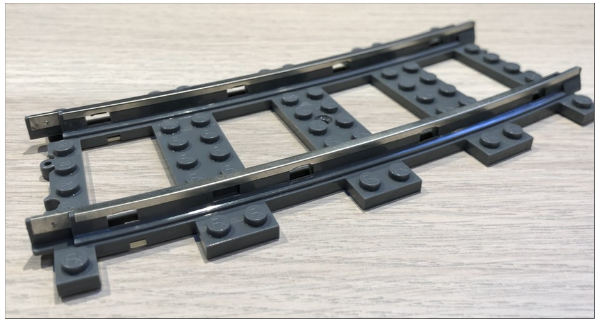
User showcase

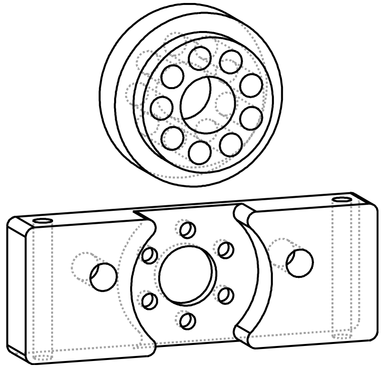
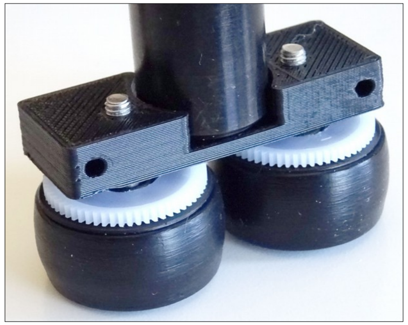
@eddieliberato - eddieliberato.github.io



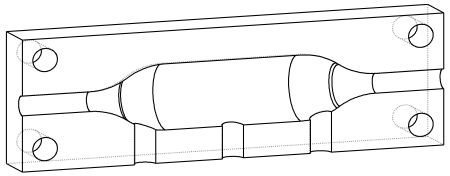
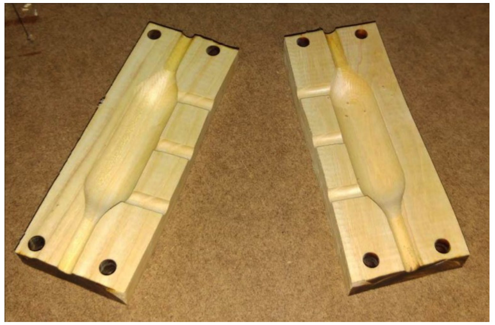
@bragostin

@michaelgale - www.fxbricks.com

@Peque - bulebule.readthedocs.io

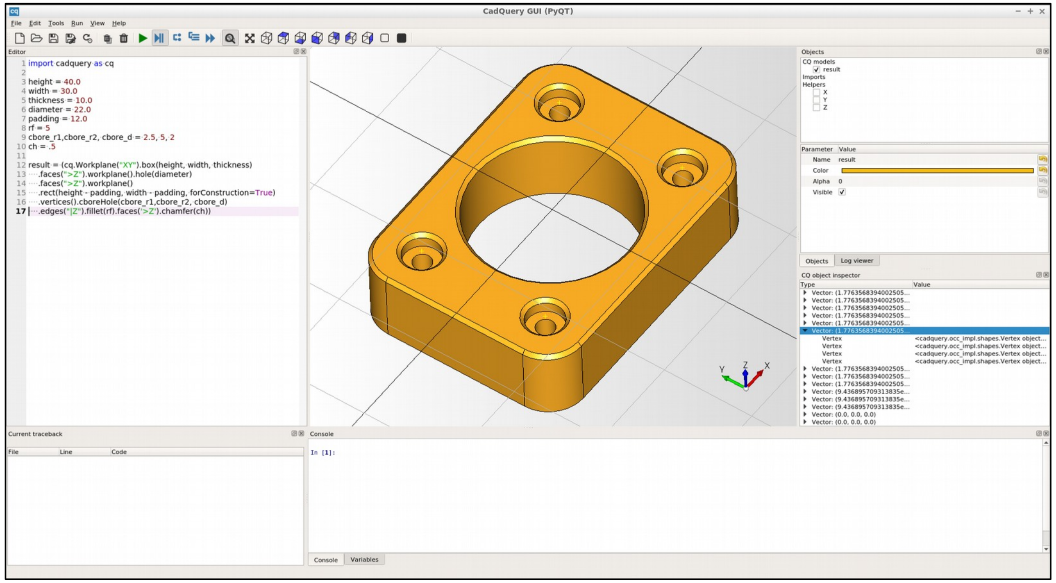



@hyOzd

NB: GitHub @usernames

CQ-editor

- Lightweight GUI for CQ
- Based on PyQT and Spyder
- CQ specific goodies
  - Graphical debugging
  - CQ stack inspection



```
import cadquery as cq

height = 40.0
width = 30.0
thickness = 10.0
diameter = 22.0
padding = 12.0
rf = 5
cbore_r1,cbore_r2, cbore_d = 2.5, 5, 2
ch = .5

result = (cq.Workplane("XY").box(height, width, thickness)
    .faces(">Z").workplane().hole(diameter)
    .faces(">Z").workplane()
    .rect(height - padding, width - padding, forConstruction=True)
    .vertices().cboreHole(cbore_r1,cbore_r2, cbore_d)
    .edges("|Z").fillet(rf).faces('>Z').chamfer(ch))
```

Objects
CQ models
[✓] result
Imports
Helpers
[ ] X
[ ] Y
[ ] Z

Parameter	Value
Name	result
Color	🟧
Alpha	0
Visible	✓

CQ object inspector

Type	Value
Vector: (1.7763568394002505...	
Vector: (1.7763568394002505...	
Vector: (1.7763568394002505...	
Vector: (1.7763568394002505...	
Vector: (1.7763568394002505...	
Vector: (1.7763568394002505...	
Vertex	<cadquery.occ_impl.shapes.Vertex object...
Vertex	<cadquery.occ_impl.shapes.Vertex object...
Vertex	<cadquery.occ_impl.shapes.Vertex object...
Vertex	<cadquery.occ_impl.shapes.Vertex object...
Vector: (1.7763568394002505...	
Vector: (1.7763568394002505...	
Vector: (1.7763568394002505...	
Vector: (9.436895709313835e...	
Vector: (9.436895709313835e...	
Vector: (9.436895709313835e...	
Vector: (0.0, 0.0, 0.0)	
Vector: (0.0, 0.0, 0.0)	

```
In [1]:
```

Future plans

- Release CQ 2.0
- Release CQ-editor 0.1
- Move to OpenCascade 7.4
- Boolean operations speed improvements
- Additional surface modeling capabilities
- DXF import
- glTF export

Standing on the shoulders of giants

CQ and CQ-editor wouldn’t be possible without the following open source projects

- Python
- OpenCascade
- FreeCAD
- PythonOCC
- PyParsing
- Conda
- Qt
- PyQt
- Spyder
- PyQtGraph
- PyInstaller