CPlot 2.5

Christophe Benoit, Stephanie Peron, Pascal Raud, Matthieu Soismier,
Bertrand Michel
- Onera -

1 CPlot: a simple plotter for arrays/pyTrees

1.1 Preamble

CPlot is a simple plotter for arrays (as defined in Converter documentation) or for CGNS/python trees (pyTrees).

This module is part of Cassiopee, a free open-source pre- and post-processor for CFD simulations.

For use with the Converter array interface, you must import the CPlot module:

```
import CPlot
```

Then, in the following, a is an array, and A a list of arrays.
For use with the pyTree interface, you must import the CPlot.PyTree module:

```
import CPlot.PyTree as CPlot
```

Then, in the following, a is a zone node and A is a list of zone nodes or a complete pyTree.

1.2 Actions

CPlot.display: display arrays/pyTrees:

```
CPlot.display(A)
```

Display function has a lot of options that can be specified as arguments:

- **dim**: 1: 1D, 2: 2D, 3: 3D (default: 3)
- **mode**: 0 or 'Mesh': mesh, 1 or 'Solid': solid, 2 or 'Render': render, 3 or 'Scalar': scalar field, 4 or 'Vector': vector field (default: 0)
- **scalarField**: scalar field number or scalarField name
- **displayBB**: 0: bounding box display (default: 1)

- **displayInfo**: 0 means no info display (default: 1)
- **displayIsoLegend**: 0 means no iso legend display (default: 0)
- **meshStyle**: 0: white solid and red wireframe, 1: colored wireframe, 2: colored solid and wireframe, 3: cyan solid and black wireframe (default: 2)
- **solidStyle**: 0: blue, 1: colored by zone, 3: cyan (default: 1)
- **scalarStyle**: 0: banded, 1: banded+mesh, 2: lines, 3: lines+mesh (default: 0)
- **colormap**: 0: Blue2Red, 2: Green2Red, 4: Black2White, 6: White2Black, 8: Diverging (default: 0)
- **niso**: number of isos (default: 25)
- **isoEdges**: width of iso edges for scalar display (default: -1)
- **isoScales**: list of min and max of a variable [novar, niso, min, max] (default: [])
- **win**: (sizeWinX, sizeWinY) window size (default: 700,700)
- **posCam**: (x,y,z) camera position
- **posEye**: (x,y,z) eye position
- **dirCam**: (x,y,z) camera direction (default: 0,0,1)
- **viewAngle**: camera angle in degrees (default: 50)
- **bgColor**: 0-10 (default: 0)
- **shadow**: 0-1 (default: 0)
- **dof**: 0-1 (default: 0)
- **stereo**: 1 or 2 means red/cyan anaglyph (default: 0)
- **stereoDist**: distance between eyes for stereo.
- **export**: file name for export
- **exportResolution**: resolution for export ("1200x900")
- **zoneNames**: optional list of zone names (same size as arrays, struct zones, then unstruct zones)
- **renderTags**: optional list of render tags (same size as arrays, struct zones, then unstruct zones)

- **offscreen**: 1 means offscreen rendering (mesa), 2 means offscreen rendering (openGL) (default: 0)

(See: display.py) (See: displayOffScreen.py) (See: displayPT.py)

CPlot.render: force rendering:

```
CPlot.render()
```

CPlot.delete: delete zones from plotter. This function does not render. Argument is either a list of zone numbers (struct zones then unstructured zones order) or a list of zone names if zoneNames arg has been provided before to display function:

```
CPlot.delete([12,13,...]) .or. CPlot.delete(['zone1', 'zone2',...])
```

(See: delete.py) (See: deletePT.py)

CPlot.add: add/insert one array in plotter. This function does not render. For array interface, no is the position of insertion of a in A:

```
CPlot.add(A, no, a, zoneName=[], renderTag=[])
```

For the pyTree interface, insert or append a to the base nob, at position noz in the zone list. If noz=-1, append to end of list:

```
CPlot.add(t, nob, noz, a)
```

CPlot.replace: performs A[no]=a, keeping plotter coherent. This function does not render. For array interface:

```
CPlot.replace(A, no, a, zoneName=None, renderTag=None)
```

For the pyTree interface, performs t[2][nob][2][noz]=a, keeping plotter coherent:

```
CPlot.replace(t, nob, noz, a)
```

CPlot.display1D: display 1D curves overlayed to CPlot display. On one screen, you can display many plots using different slots. gridPos specified the position on the screen (see the State.gridSize). bgBlend indicates the blending of the background, var1 and var2 are the name of the variables to be plotted:

```
CPlot.display1D(A, slot=0, gridPos=(0,0), gridSize=(-1,-1), bgBlend=1., var1='', var2='')
```

(See: display1D.py) (See: display1DPT.py)

CPlot.pressKey: wait for a key to be pressed:

```
CPlot.pressKey()
```

CPlot.finalizeExport: finalize an export. It acts as a barrier if no arg is specified. Must be called after dumping all images of a movie with arg 1:

```
CPlot.finalizeExport()
```
(See: displayOffScreen2.py)

1.3 Set / get functions

CPlot.getState: return the specified state value in plotter:
```
v = CPlot.getState(stateName)
```
(See: getState.py)

CPlot.getSelectedZone: return the currently selected zone. If none is selected, return -1. If multiple zones are selected, return the last selected zone:
```
n = CPlot.getSelectedZone()
```
(See: getSelectedZone.py)

CPlot.getSelectedZones: return the list of selected zones. If none is selected, return []:
```
N = CPlot.getSelectedZones()
```
(See: getSelectedZones.py)

CPlot.getSelectedStatus: return the selected status (1: selected, 0: not selected) of zone nz:
```
status = CPlot.getSelectedStatus(nz)
```
(See: getSelectedStatus.py)

CPlot.getActiveZones: return the list of active (displayed) zones:
```
N = CPlot.getActiveZones()
```
(See: getActiveZones.py)

CPlot.getActiveStatus: return the active status (1: active, 0: inactive) of zone nz:
```
status = CPlot.getActiveStatus(nz)
```
(See: getActiveStatus.py)

CPlot.getActivePoint: return the last clicked point position (coordinates in 3D world) as a list of three coordinates:
```
point = CPlot.getActivePoint()
```
(See: getActivePoint.py)

CPlot.getActivePointIndex: return the active point index. For structured grids, return [ind, indc, i,j,k], where ind is the global index of the nearest node to active point, indc is the global index of the nearest center to active point and i,j,k are the indices of nearest node. For unstructured grids, return [ind, indc, no, 0, 0], where ind is the global index of nearest node, indc is the nearest center to the clicked point and no is the number of ind in the center2node connectivity of indc:
```
ind = CPlot.getActivePointIndex()
```
(See: getActivePointIndex.py)

CPlot.getMouseState: return the current button state of mouse (0: left pressed, 1: middle pressed, 2: righ pressed, 5: not pressed) and the current mouse position (if pressed). Use it when dragging:

```
(state, ind) = CPlot.getMouseState()
```

(See: getMouseState.py)

CPlot.getKeyboard: return the pressed keys as a string:

```
string = CPlot.getKeyboard()
```

(See: getKeyboard.py)

CPlot.resetKeyboard: reset the pressed key string:

```
CPlot.resetKeyboard()
```

CPlot.changeVariable: change displayed variable:

```
CPlot.changeVariable()
```

(See: changeVariable.py)

CPlot.changeStyle: change CPlot display style:

```
CPlot.changeStyle()
```

(See: changeStyle.py)

CPlot.changeBlanking: change the blanking procedure:

```
CPlot.changeBlanking()
```

(See: changeBlanking.py)

CPlot.setState: set CPlot state. The same keywords as display can be used:

```
CPlot.setState(dim=2, mode=0, ...)
```

Additional keywords are:

- **ghostifyDeactivatedZones**: 1 means deactivated zones will appear blended.
- **edgifyActivatedZones**: 1 means activated zones will appear as edges.
- **edgifyDeactivatedZones**: 1 means deactivated zones will appear as edges.
- **message**: ”A string” or ”Clear”
- **viewAngle**: the camera angle (default: 50 degrees).
- **cursor**: mouse cursor type (0: normal, 1: cross, 2: wait).
- **lightOffset**: offset to default light position (default: (0,0)).
- **dofPower**: power of depth of field effect (default: 6.).

- **selectionStyle**: style for selection (default: 0).

(See: setState.py)

CPlot.setMode: set CPlot display mode (0 or 'Mesh': means mesh, 1 or 'Solid': means solid, 2 or 'Render': means render, 3 or 'Scalar' means scalar field, 4 or 'Vector' means vector fields):

```
CPlot.setMode(0) .or. CPlot.setMode('Mesh')
```

(See: setMode.py)

CPlot.setSelectedZones: set the selected zone status (1: selected, 0: not selected) by zone global number:

```
CPlot.setSelectedZones([(0,1), (1,0), (2,1),...])
```

(See: setSelectedZones.py)

CPlot.unselectAllZones: unselect all zones:

```
CPlot.unselectAllZones()
```

(See: unselectAllZones.py)

CPlot.setActiveZones: set the active (displayed) zones:

```
CPlot.setActiveZones([(0,1), (1,0), (2,1),...])
```

(See: setActiveZones.py)

CPlot.setZoneNames: set the specified zone names:

```
CPlot.setZoneNames([(0,'Zone0'), (1,'Zone1'), (2,'Zone2'),...])
```

(See: setZoneNames.py)

CPlot.lookFor: look for selected zone:

```
CPlot.lookFor()
```

(See: lookFor.py)

CPlot.moveCamera: move camera along check points:

```
CPlot.moveCamera([(0,0,0),(1,0,0),(1,1,0), moveEye=False, N=100, speed=50.)
```

(See: moveCamera.py)

CPlot.travelLeft/Right/Up/Down/In/Out: travel camera left/Right/Up/Down/In/Out. Xr is the range (in 0.,1.). N is the number of check points:

```
CPlot.travelLeft(xr, N=100)
```

(See: travel.py)

1.4 Set rendering informations in pyTree

CPlot.addRender2Zone: add rendering info to a zone. Info are added in a .RenderInfo user defined node:

```
b = CPlot.addRender2Zone(a, material, color, blending, meshOverlay, shaderParameters)
```

(See: addRender2ZonePT.py)

CPlot.addRender2PyTree: add rendering info to a tree. Info are added in a .RenderInfo user defined node. To load the settings to the view, call explicitely CPlot.loadView:

```
b = CPlot.addRender2PyTree(a, slot, posCam, posEye, dirCam, mode, scalarField, niso,
isoScales, isoEdges, isoLight, colormap)
```

(See: addRender2PyTreePT.py)

CPlot.loadView: load a view defined in a slot to the plotter:

```
CPlot.loadView(a, slot=0)
```

(See: loadViewPT.py)

1.5 Commands in CPlot window

- **f**: fit view.
- **Ctrl+f**: full screen.
- **Arrows** or **left mouse**: move around.
- **Shift + Arrows** or **right mouse**: strafe.
- **Ctrl + Arrows** or **Ctrl + left mouse**: move your head.
- **Ctrl + right mouse**: tilt your head.
- **Shift + left mouse**: select zone (mouse plugin).
- **Shift + Ctrl + left mouse**: multiple select (mouse plugin).
- **Ctrl + left mouse**: Accurate select (mouse plugin).
- **Shift + right mouse**: deactivate zone (mouse plugin).
- **Shift + double left mouse**: center view.
- **o** or **left mouse**: move up.
- **p** or **left mouse**: move down.
- **1-Shift+1** : display fields (primary variable).
- **2-Shift+2** : change secondary variable.
- **Space bar** : mesh display.

- **Shift+Space bar** : solid display.
- **m-M**: toggle between 1D, 2D, 3D mode.
- **z** or **Z**: change selected zone.
- **a** or **A**: activate/deactivate a zone.
- **l**: look for active zone.
- **c**: change render appearance (colormaps,...)
- **i** or **I** or **Ctrl+i**: change displayed i plane.
- **j** or **J** or **Ctrl+j**: change displayed j plane.
- **k** or **K** or **Ctrl+k**: change displayed k plane.
- **r**: reload files.
- **q**: quit.
- **Esc**: display CPlot menu.

1.6 CPlot shell command

CPlot can be also used as a shell command:

```
cplot file
```

CPlot can read all formats readable with converter.

1.7 Example files

Example file: display.py

```
# - display (array) -
import Generator as G
import CPlot
import Transform as T
import Converter as C

a = G.cart((0,0,0),(1,1,1),(18,28,3))
CPlot.display(a, displayBB=0, mode='mesh')

for i in xrange(360):
    a = T.rotate(a, (9, 14, 3.5), (0,0,1), 1.)
    CPlot.display(a)
```

Example file: displayOffScreen.py

```
# - display (array) -
# Offscreen using mesa
import Generator as G
import CPlot
import Transform as T
import Converter as C
import Geom as D
import time

a = D.sphere( (0,0,0), 1)

# One image
CPlot.display([a], offscreen=1, bgColor=1, mode=1, meshStyle=2,
              solidStyle=1, posCam=(0,6,0), export="one.png")

# Movie
for i in xrange(50):
    a = T.rotate(a, (0,0,0), (0,0,1), 1.)
    CPlot.display([a], offscreen=1, bgColor=1, mode=1, meshStyle=2,
                  solidStyle=1, posCam=(0,6,0),
                  exportResolution='680x600', export="export.mpeg")
time.sleep(0.1)
time.sleep(1); CPlot.finalizeExport()
import os; os._exit(0)
```

Example file: displayPT.py

```
# - display (pyTree) -
import Generator.PyTree as G
import Converter.PyTree as C
import CPlot.PyTree
import Transform.PyTree as T

a = G.cart((0,0,0),(1,1,1),(18,28,3))
t = C.newPyTree(['Base',a])

for i in xrange(360):
    t = T.rotate(t, (9, 14, 3.5), (0,0,1), 1.)
    CPlot.PyTree.display(t)
```

Example file: delete.py

```
# - delete (array) -
import Generator as G
import CPlot
import time

a = G.cart( (0,0,0), (1,1,1), (10,10,10) )
b = G.cart( (11,0,0), (1,1,1), (10,10,10) )

CPlot.display([a,b]); time.sleep(1)
CPlot.delete([0]); CPlot.render(); time.sleep(1)
CPlot.delete(['S-Zone 1']); CPlot.render(); time.sleep(1)
```

Example file: deletePT.py

```
# - delete (pyTree) -
import Generator.PyTree as G
import CPlot.PyTree as CPlot
import Converter.PyTree as C
import time
```

```
a = G.cart( (0,0,0), (1,1,1), (10,10,10) )
b = G.cartTetra( (11,0,0), (1,1,1), (10,10,10) )
c = G.cart( (0,11,0), (1,1,1), (10,10,10) )
t = C.newPyTree(['Base', a, b, c])

CPlot.display(t); time.sleep(1)
CPlot.delete(['Base/cartTetra']); CPlot.render(); time.sleep(1)
```

Example file: display1D.py

```
# - display1D (array) -
import Generator as G
import CPlot
import Transform as T
import Converter as C
import numpy

# 1D data defined in arrays
b = G.cart((0,0,0), (0.1,1,1), (50,1,1))
c = G.cart((5,0,0), (0.1,1,1), (50,1,1))
B = [b, c]
CPlot.setState(gridSize=(1,2))
for i in xrange(100):
     B = C.initVars(B, 'f=sin({x}+0.02*%d)'%i)
     B = C.initVars(B, 'g={x}')
     CPlot.display1D(B, slot=0, bgBlend=1., gridPos=(0,0),
                     var1='x', var2='f')

# 1D data defined in numpys
x = numpy.linspace(0, 2*numpy.pi)
y = numpy.sin(x)
CPlot.display1D([x,y], slot=1, var1='x', var2='y', gridPos=(0,1), bgBlend=0.)
```

Example file: display1DPT.py

```
# - display1D (pyTree) -
import Generator.PyTree as G
import CPlot.PyTree as CPlot
import Converter.PyTree as C
import numpy

# 1D data defined in zones
b = G.cart((0,0,0), (0.1,1,1), (50,1,1))
c = G.cart((5,0,0), (0.1,1,1), (50,1,1))
B = [b, c]
CPlot.setState(gridSize=(1,2))
for i in xrange(100):
    C._initVars(B, 'f=sin({CoordinateX}+0.01*%d)'%i)
    C._initVars(B, 'g={CoordinateX}')
    CPlot.display1D(B, slot=0, bgBlend=1., gridPos=(0,0), var1='CoordinateX', var2='f')

# 1D data defined in numpys
import numpy
x = numpy.linspace(0, 2*numpy.pi)
y = numpy.sin(x)
CPlot.display1D([x,y], slot=1, var1='x', var2='y', gridPos=(0,1), bgBlend=0.8)
```

Example file: displayOffScreen2.py

```
# - display (array) -
# Offscreen using FBO
import Generator as G
import CPlot
import Transform as T
import Converter as C
import Geom as D
import time

a = D.sphere((0,0,0), 1, N=200)

# Multi images
CPlot.display([a], offscreen=2, bgColor=1, mode=0, meshStyle=2,
              solidStyle=1, posCam=(0,6,0), export='one.png')
for i in xrange(5):
    a = T.rotate(a, (0,0,0), (0,0,1), 1.)
    CPlot.display([a], offscreen=2, bgColor=1, mode=0, meshStyle=2,
              solidStyle=1, posCam=(0,6,0), export='one%d.png'%i)
    CPlot.finalizeExport()
import os; os._exit(0)

# Mpeg Movie
for i in xrange(50):
    a = T.rotate(a, (0,0,0), (0,0,1), 1.)
    CPlot.display([a], bgColor=1, mode=0,
                  solidStyle=1, posCam=(0,6,0), export='export.mpeg',
                  exportResolution='680x600', offscreen=2)
    CPlot.finalizeExport()
CPlot.finalizeExport(1)
import os; os._exit(0)
```

Example file: getState.py

```
# - getState (array) -
import Generator as G
import CPlot
import time

a = G.cart( (0,0,0), (1,1,1), (5,5,5) )
CPlot.display([a])

print 'dim=',CPlot.getState('dim')
print 'mode=',CPlot.getState('mode')
print 'displayBB=',CPlot.getState('displayBB')
print 'displayInfo=',CPlot.getState('displayInfo')
print 'meshStyle=',CPlot.getState('meshStyle')
print 'solidStyle=',CPlot.getState('solidStyle')
print 'isoStyle=',CPlot.getState('isoStyle')
print 'win=',CPlot.getState('win')
print 'posCam=',CPlot.getState('posCam')
print 'posEye=',CPlot.getState('posEye')
print 'dirCam=',CPlot.getState('dirCam')
```

Example file: getSelectedZone.py

```
# - getSelectedZone (array) -
import Generator as G
import CPlot
import time

a = G.cart((0,0,0), (1,1,1), (5,5,5))
```

```
CPlot.display(a)

nz = -1
while (nz == -1):
    nz = CPlot.getSelectedZone(); time.sleep(0.1)
print 'One zone has been selected: ', nz
```

Example file: getSelectedZones.py

```
# - getSelectedZones (array) -
import Generator as G
import CPlot
import time

a1 = G.cart( (0,0,0), (1,1,1), (5,5,5) )
a2 = G.cart( (7,0,0), (1,1,1), (3,3,3) )
CPlot.display([a1, a2])

ret = []
while (ret == []):
    ret = CPlot.getSelectedZones(); time.sleep(2.)
print 'Zones have been selected: ', ret
```

Example file: getSelectedStatus.py

```
# - getSelectedStatus (array) -
import Generator as G
import CPlot
import time

a1 = G.cart( (0,0,0), (1,1,1), (5,5,5) )
a2 = G.cart( (7,0,0), (1,1,1), (3,3,3) )
CPlot.display([a1, a2])

time.sleep(5.)
ret = CPlot.getSelectedStatus(0)
if (ret == 1): print 'Zone 0 is selected.'
else: print 'Zone 0 is not selected.'
```

Example file: getActiveZones.py

```
# - getActiveZones (array) -
import Generator as G
import CPlot
import time

a1 = G.cart( (0,0,0), (1,1,1), (5,5,5) )
a2 = G.cart( (7,0,0), (1,1,1), (3,3,3) )
CPlot.display([a1, a2])

time.sleep(5.)
ret = CPlot.getActiveZones()
print 'Active zones: ', ret
```

Example file: getActiveStatus.py

```
# - getActiveStatus (array) -
import Generator as G
import CPlot
import time

a1 = G.cart( (0,0,0), (1,1,1), (5,5,5) )
```

```
a2 = G.cart( (7,0,0), (1,1,1), (3,3,3) )
CPlot.display([a1, a2])

time.sleep(5.)
ret = CPlot.getActiveStatus(0)
if (ret == 1): print 'Zone 0 is active.'
else: print 'Zone 0 is inactive.'
```

Example file: getActivePoint.py

```
# - getActivePoint (array) -
import Generator as G
import CPlot
import time

a = G.cart( (0,0,0), (1,1,1), (5,5,5) )
CPlot.display([a])

l = []
while (l == []):
    l = CPlot.getActivePoint(); time.sleep(0.1)
print 'ActivePoint: ', l
```

Example file: getActivePointIndex.py

```
# - getActivePointIndex (array) -
import Generator as G
import CPlot
import time

a = G.cartTetra( (0,0,0), (1,1,1), (5,5,1) )
CPlot.display([a], dim=2)

l = []
while (l == []):
    l = CPlot.getActivePointIndex(); time.sleep(0.1)
print 'ActivePointIndex : ', l
```

Example file: getMouseState.py

```
# - getMoouseState (array) -
import Generator as G
import CPlot
import time

a = G.cartTetra( (0,0,0), (1,1,1), (5,5,1) )
CPlot.display([a], dim=2)

c = 1000
while (c > 0):
    l = CPlot.getMouseState(); time.sleep(0.5)
    print l
    c -= 1
```

Example file: getKeyboard.py

```
# - getKeyboard (array) -
import Generator as G
import CPlot
import Geom as D
import time
```

```
a = G.cart((0,0,0), (1,1,1), (8,8,1))
CPlot.display(a, dim=2)
CPlot.setState(activateShortCuts=0)

for i in xrange(50):
    l = ''
    while l == '':
        l = CPlot.getKeyboard(); time.sleep(0.1)
    try:
        a = D.text2D(l)
        CPlot.display(a)
    except:
        v = ord(l[0])
        if v == 1: print 'up'
        elif v == 2: print 'down'
        elif v == 3: print 'left'
        elif v == 4: print 'right'
        time.sleep(0.1)
        l = ''
```

Example file: changeVariable.py

```
# - changeVariable (array) -
import Generator as G
import Converter as C
import CPlot
import time

def F(x,y): return x*x + y*y

a = G.cart( (0,0,0), (1,1,1), (5,5,1) )
a = C.addVar(a, 'Density')
a = C.initVars(a, 'F', F, ['x','y']); t = [a]
CPlot.display(t, dim=2, mode=3)

CPlot.changeVariable(); time.sleep(2)
CPlot.changeVariable(); time.sleep(2)
```

Example file: changeStyle.py

```
# - changeStyle (array) -
import Generator as G
import CPlot
import time

a = G.cart( (0,0,0), (1,1,1), (5,5,1) ); t = [a]
CPlot.display(t, dim=2, mode=1)

CPlot.changeStyle(); time.sleep(2)
CPlot.changeStyle(); time.sleep(2)
```

Example file: changeBlanking.py

```
# - changeBlanking (array) -
import Generator as G
import Converter as C
import CPlot
import time

a = G.cart( (0,0,0), (1,1,1), (5,5,1) )
```

```
a = C.initVars(a, 'cellN', 0)
CPlot.display([a], dim=2, mode=0)

CPlot.changeBlanking(); time.sleep(2)
CPlot.changeBlanking(); time.sleep(2)
```

Example file: setState.py

```
# - setState (array) -
import Generator as G
import CPlot
import time

a = G.cart((0,0,0), (1,1,1), (5,5,5))
CPlot.display([a], mode='solid')
time.sleep(0.2)
CPlot.setState(posCam=(8,8,8), posEye=(5,5,5))
```

Example file: setMode.py

```
# - setMode (array) -
import Generator as G
import CPlot
import time

a = G.cart( (0,0,0), (1,1,1), (5,5,1) )
CPlot.display(a, mode=0, dim=2); time.sleep(2)
CPlot.setMode(1); time.sleep(2) # solid
CPlot.setMode('mesh'); time.sleep(2)
CPlot.setMode('solid'); time.sleep(2)
```

Example file: setSelectedZones.py

```
# - setSelectedZones (array) -
import Generator as G
import CPlot
import time

a1 = G.cart( (0,0,0), (1,1,1), (5,5,5) )
a2 = G.cart( (7,0,0), (1,1,1), (3,3,3) )
CPlot.display([a1, a2])

time.sleep(1.)
CPlot.setSelectedZones([(0,1), (1,1)])
```

Example file: unselectAllZones.py

```
# - unselectAllZones (array) -
import Generator as G
import CPlot
import time

a1 = G.cart( (0,0,0), (1,1,1), (5,5,5) )
a2 = G.cart( (7,0,0), (1,1,1), (3,3,3) )
CPlot.display([a1, a2])

time.sleep(2.)
CPlot.unselectAllZones()
```

Example file: setActiveZones.py

```
# - setActiveZones (array) -
import Generator as G
import CPlot
import time

a1 = G.cart( (0,0,0), (1,1,1), (5,5,5) )
a2 = G.cart( (7,0,0), (1,1,1), (3,3,3) )
CPlot.display([a1, a2])

time.sleep(1.)
CPlot.setActiveZones([(0,0), (1,0)])
```

Example file: setZoneNames.py

```
# - setZoneNames (array) -
import Generator as G
import CPlot
import time

a1 = G.cart( (0,0,0), (1,1,1), (5,5,5) )
a2 = G.cart( (7,0,0), (1,1,1), (3,3,3) )
CPlot.display([a1, a2])

time.sleep(1.)
CPlot.setZoneNames([(0,'FirstZone'), (1,'SecondZone')])
```

Example file: lookFor.py

```
# - lookFor (array) -
import Generator as G
import CPlot
import time

a = G.cart( (0,0,0), (1,1,1), (5,5,5) )
CPlot.display([a])
CPlot.lookFor()
```

Example file: moveCamera.py

```
# - moveCamera (array) -
import Geom as D
import CPlot

# Model
a = D.sphere((0,0,0), 1., N=20)
CPlot.display(a, posCam=(3,-1,0.7), posEye=(0,0,0))

# Move camera
CPlot.moveCamera([(3,-1,0.7),(3,5,0.7),(3,7,0.7)])
```

Example file: travel.py

```
# - travel* (array) -
import Geom as D
import Generator as G
import CPlot
import Converter as C
import Transform as T

# Model
a = D.sphere((0,0,0), 1., N=20)
a = C.convertArray2Hexa(a); a = G.close(a)
```

```
CPlot.display(a, posCam=(3,0,0), posEye=(0,0,0))

time = 0.
for i in xrange(1300):
    # change model
    defo = C.initVars(a, '{df}=0.1*cos(%f)*sin(10*pi*{x})'%(time))
    defo = C.extractVars(defo, ['df'])
    b = T.deformNormals(a, defo)
    CPlot.display(b)
    time += 0.05

    if i < 200: CPlot.travelLeft(0.001, N=3)
    elif i < 400: CPlot.travelRight(0.001, N=3)
    elif i < 600: CPlot.travelUp(0.001, N=3)
    elif i < 800: CPlot.travelIn(0.001, N=3)
```

Example file: addRender2ZonePT.py

```
# - addRender2Zone (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import CPlot.PyTree as CPlot

a = G.cart((0,0,0), (1,1,1), (10,10,1))
C._initVars(a, 'Density={CoordinateX}')
C._initVars(a, 'centers:VelocityX={centers:CoordinateY}')
# With a material
a = CPlot.addRender2Zone(a, material='Glass', color='Blue', blending=1.,
                         meshOverlay=1, shaderParameters=[1.,1.])
# With field
a = CPlot.addRender2Zone(a, material='Solid', color='Iso:Density', blending=1.,
                         meshOverlay=1, shaderParameters=[1.,1.])
# With field+material
a = CPlot.addRender2Zone(a, material='Chrome', color='Iso:centers:VelocityX', blending=1.,
                         meshOverlay=1, shaderParameters=[1.,1.])

C.convertPyTree2File(a, 'out.cgns')
```

Example file: addRender2PyTreePT.py

```
# - addRender2PyTree (pyTree) -
import Converter.PyTree as C
import CPlot.PyTree as CPlot
import Generator.PyTree as G

a = G.cart((0,0,0), (1,1,1), (10,10,1))
t = C.newPyTree(['Base', a])
# isoScales specify field no, niso for this field, min, max for this field
t = CPlot.addRender2PyTree(t, slot=0, posCam=(1,1,1), posEye=(10,1,1),
                           mode='Solid', niso=10, isoScales=[0, 10, 0., 1.],
                           isoEdges=0.1, isoLight=1, colormap='Blue2Red')
C.convertPyTree2File(t, 'out.cgns')
```

Example file: loadViewPT.py

```
# - loadView (pyTree) -
import Converter.PyTree as C
import CPlot.PyTree as CPlot
import Generator.PyTree as G
```

```
a = G.cart( (0,0,0), (1,1,1), (10,10,1) )
t = C.newPyTree(['Base', 2, a])

# isoScales specify field no, niso for this field, min, max for this field
t = CPlot.addRender2PyTree(t, slot=0, posCam=(1,1,1), posEye=(10,1,1),
                           mode='Solid', niso=10, isoScales=[0, 10, 0., 1.],
                           isoEdges=0.1, isoLight=1, colormap='Blue2Red')

CPlot.display(t)
import time; time.sleep(0.1)
CPlot.loadView(t, slot=0)
```