VPython - symulacje fizyczne z grafiką 3D dla każdego

wykład 10

Dr hab. Adam Bzdak, prof. AGH

scene.objects

vp_scene_1.py - C:/Users/Wysokie Energie/Desktop/VPython/lec_a_10/vp_scene_1.py

File Edit Format Run Options Windows Help

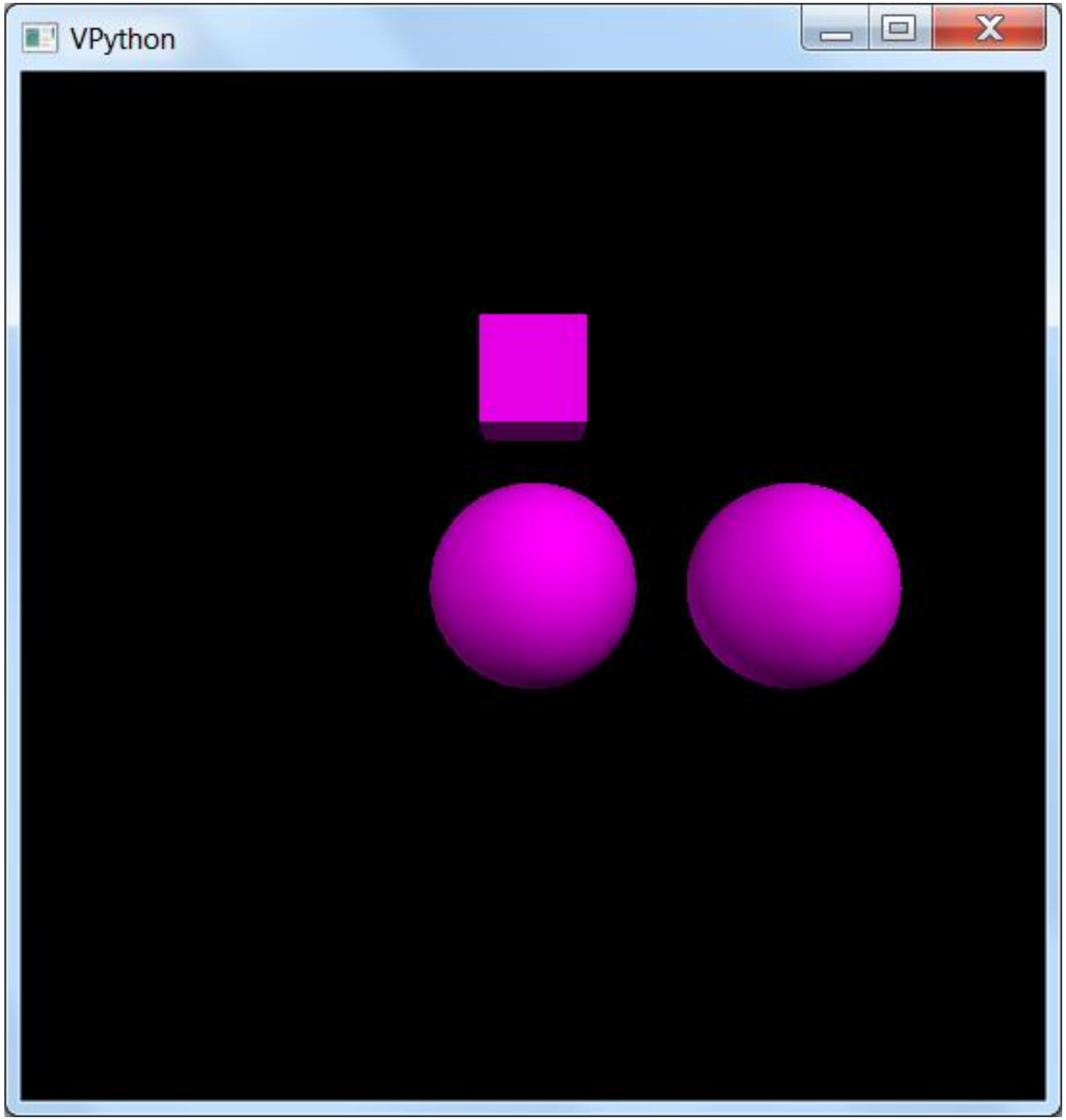
```
from visual import *

scene = display(width=550, height=580, range=5)

ba1 = sphere(pos=(0,0,0), radius=1)
ba2 = sphere(pos=(2.5,0,0), radius=1)

wa1 = box(pos=(0,2,0))
wa2 = box(pos=(0,4,0))

sleep(2)

wa2.visible = False

for k in scene.objects:
    print k.pos
    k.color = color.magenta
```

wszystkie widoczne obiekty

Ln: 20 Col: 0

```
<0, 0, 0>
<2.5, 0, 0>
<0, 2, 0>
```

Ln: 10 Col: 4


VPython

krzywa (curve)

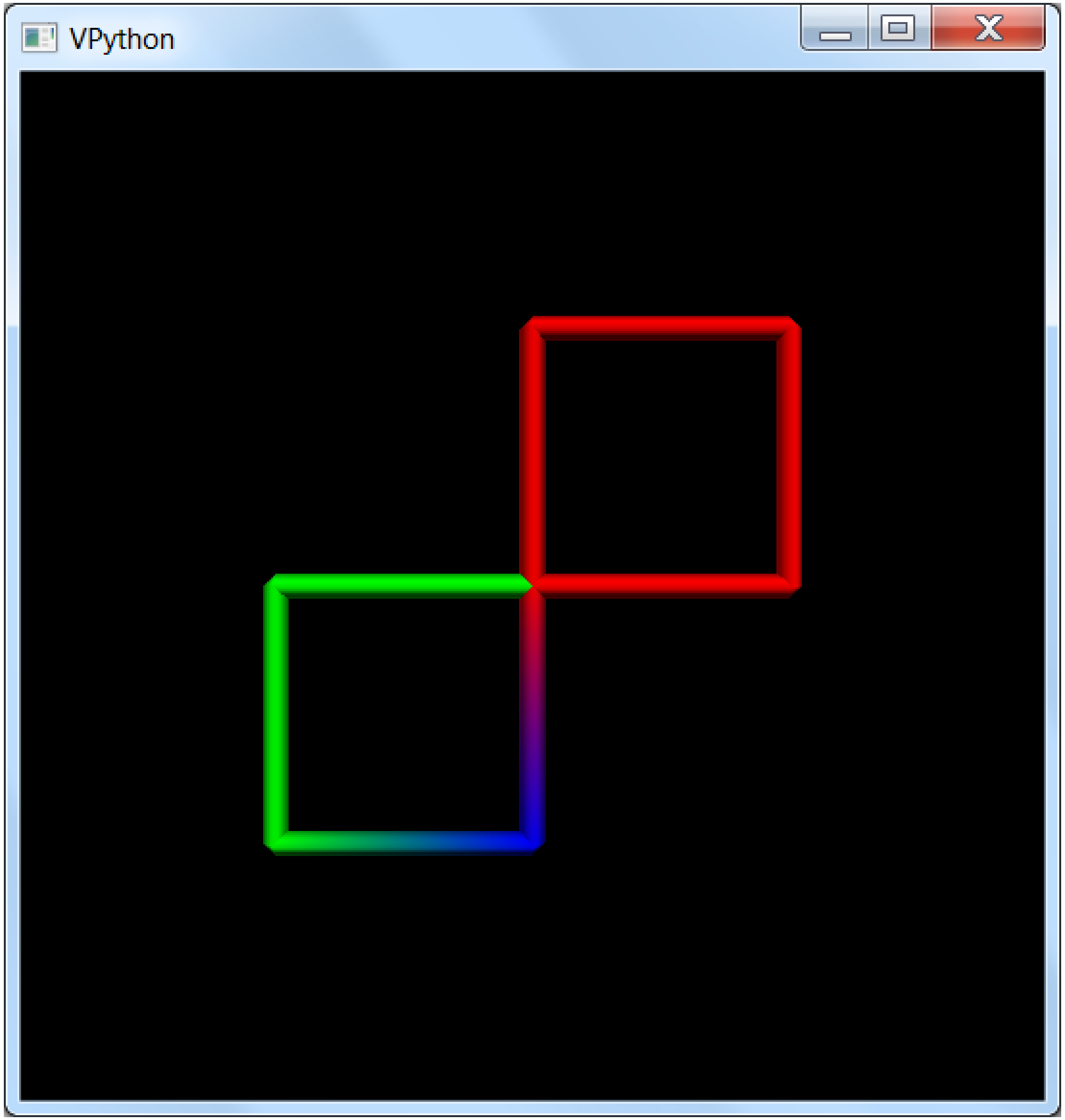
vp_curve_1.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_a_10\vp_curve_1.py

```
from visual import *

scene = display(width=550, height=580, range=2)

line = curve(pos=[(0,0,0),(1,0,0),(1,1,0),(0,1,0),(0,0,0)],
             color=color.red, radius=0.05)

line.append(pos=(0,-1,0), color=color.blue)
line.append(pos=(-1,-1,0), color=color.green)
line.append(pos=(-1,0,0))
line.append(pos=(0,0,0))
```

Ln: 13 Col: 0

Jak nie podamy promienia to rysuje cienką linię

(0,0,0) → vector(0,0,0) w VPython 7


VPython

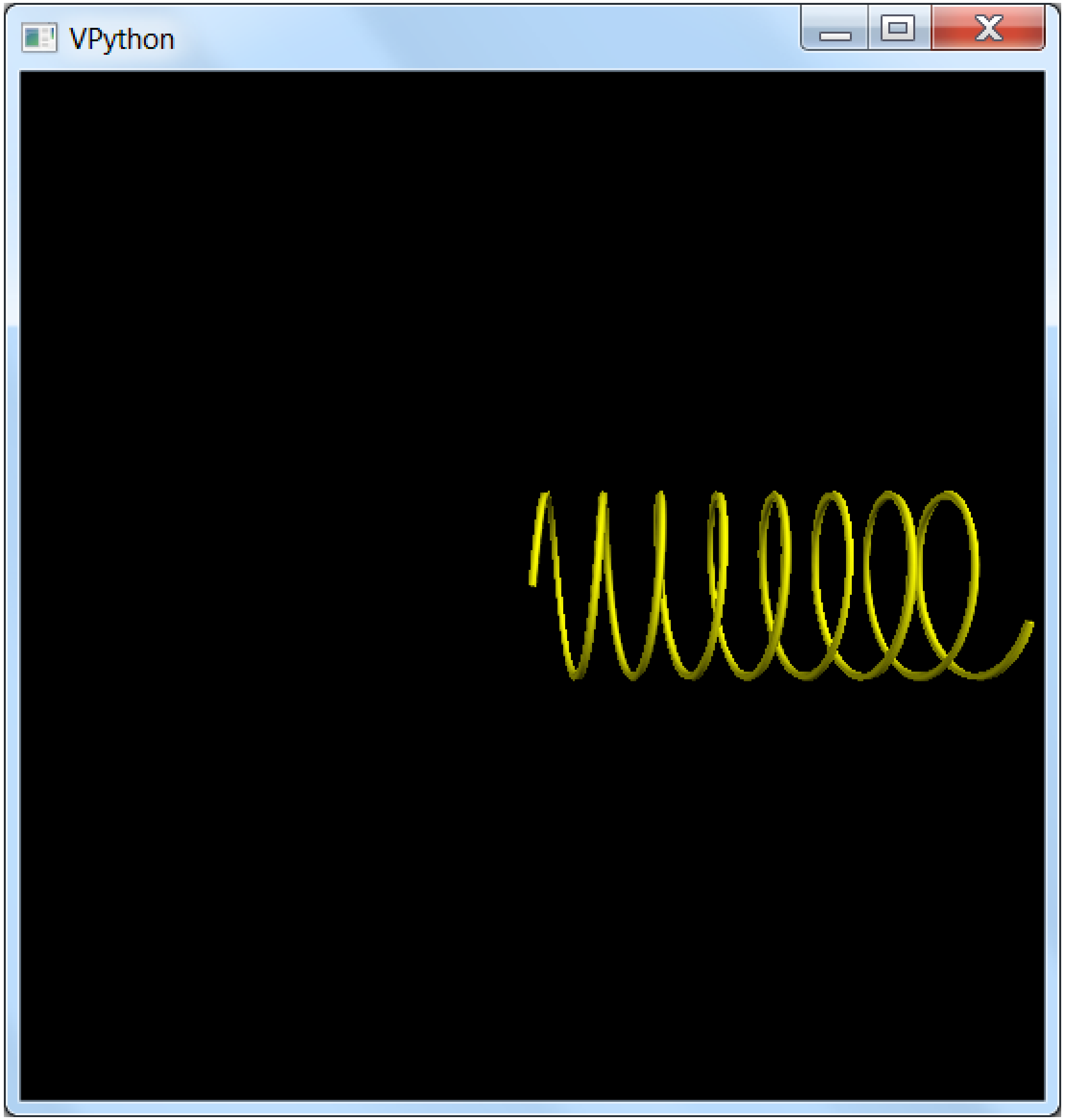
curve - x, y, z

vp_curve_2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_a_10\vp_curve_2.py

File Edit Format Run Options Windows Help

```
from visual import *

scene = display(width=550, height=580)

line = curve(x=arange(0,5,0.01), color=color.yellow,
             radius=0.05)

line.y = sin(10*line.x)
line.z = cos(10*line.x)
```

Ln: 11 Col: 0


VPython

curve

vp_curve_3.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_a_10\vp_curve_3.py

File Edit Format Run Options Windows Help

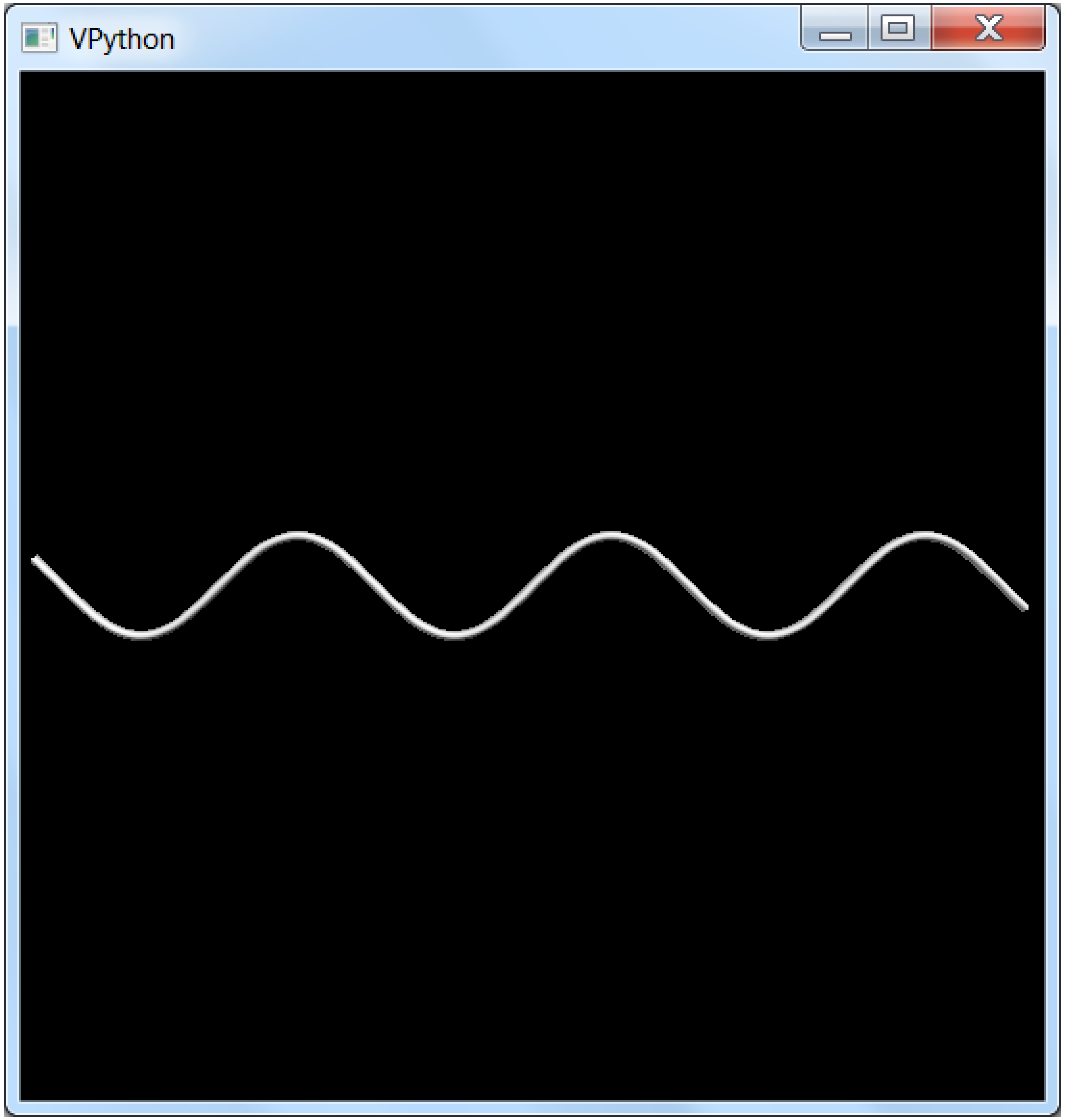
```
from visual import *

scene = display(width=550, height=580)

x = arange(-10,10,0.1)
func = raw_input('function: ')

curve(x=x, y=eval(func), radius=0.1)
```

Ln: 10 Col: 0

```
>>>
function: sin(x)
```

wpisuje sin(x)

Ln: 8 Col: 4


VPython

curve, ślad

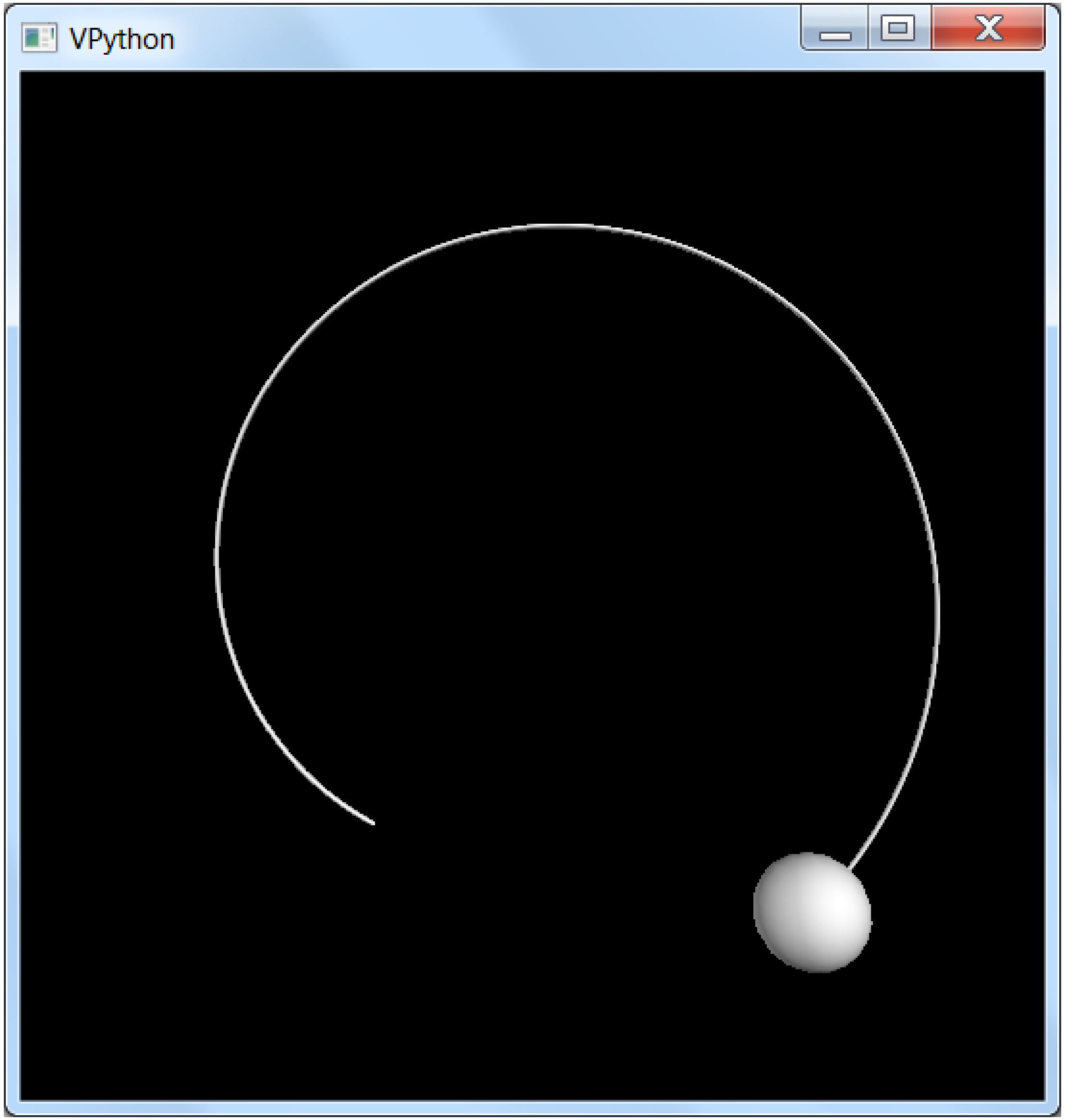
vp_curve_4.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_a_10\vp_curve_4.py

File Edit Format Run Options Windows Help

```
from visual import *

scene = display(width=550, height=580, range=9)

ba = sphere(pos=(0,5,0))
tra = curve(radius=0.05)

t = 0
while t<15:
    rate(200)
    ba.pos.x = 0.5*t*sin(t)
    ba.pos.y = 0.5*t*cos(t)
    tra.append(pos=ba.pos, retain=500)
    t += 0.01
```

Ln: 17 Col: 0

trzymamy ostatnie
500 punktów


VPython

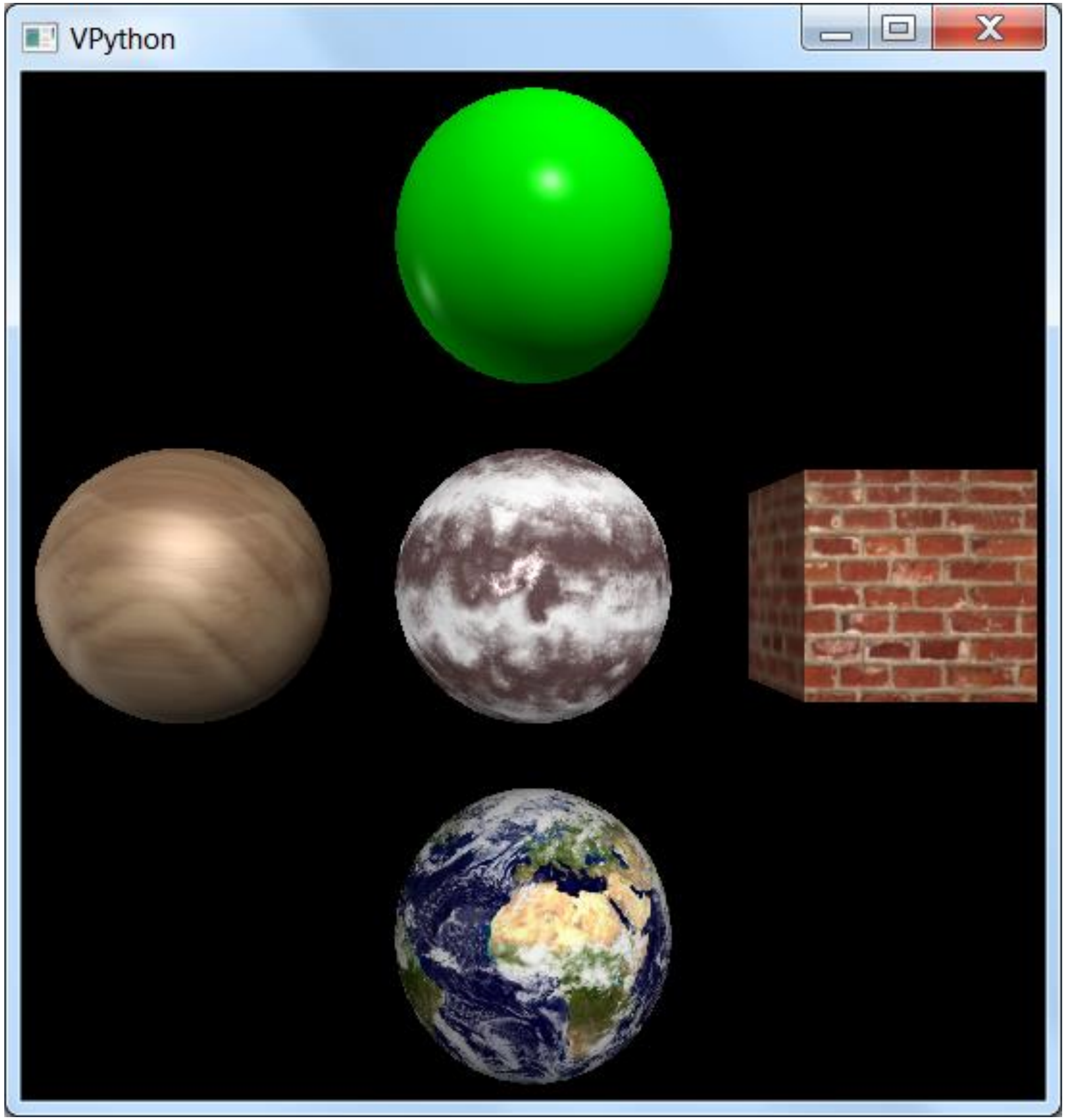
Materiał (material), działa w VPython 6

vp_material_1.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_a_10\vp_material_1.py

File Edit Format Run Options Windows Help

```
from visual import *

scene = display(width=550, height=580)

scene.material = materials.marble

sphere(pos=(0,0,0))

sphere(pos=(-2.5,0,0), material=materials.wood)

sphere(pos=(0,2.5,0), material=materials.plastic,
       color=color.green)

box(pos=(2.5,0,0), size=(1.5,1.5,1.5),
    material=materials.bricks)

sphere(pos=(0,-2.5,0), material=materials.BlueMarble)

# wood, rough, marble, plastic, earth, diffuse
# emissive, unshaded, shiny, chrome, blazed, silver,
# BlueMarble, bricks
```

Ln: 23 Col: 0


VPython

W VPython 7 mamy texture

https://www.glowscript.org/docs/VPythonDocs/textures.html

np.

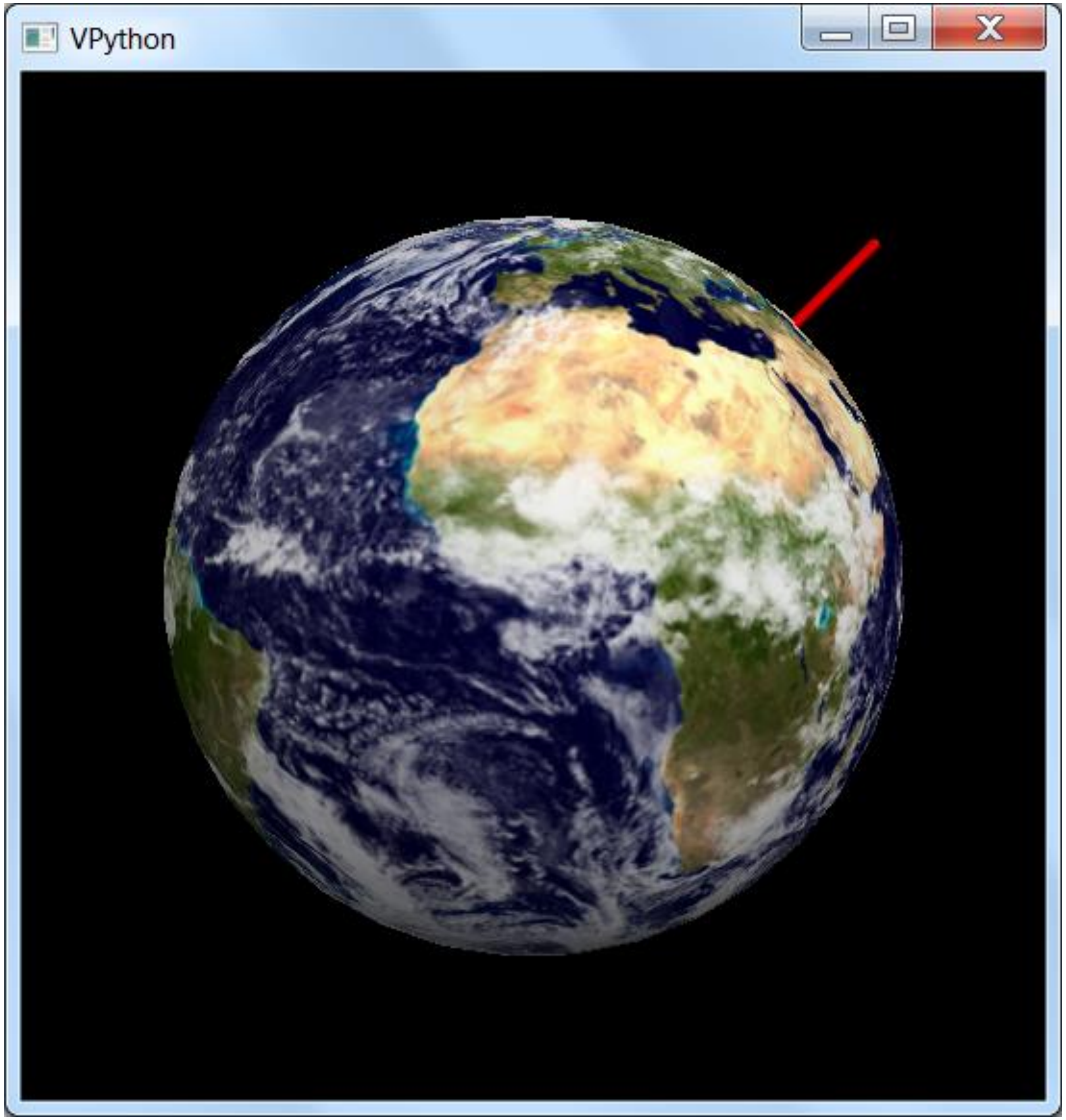
box(texture=textures.stucco)

obrót (rotate)

vp_rotate_1.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_a_10\vp_rotate_1.py

```python
from visual import *

scene = display(width=550, height=580, range=1.5)

ba = sphere(material=materials.BlueMarble)

curve(pos=[(0,0,0),(1,1,0)], radius=0.02, color=(1,0,0))

while True:
    rate(500)
    ba.rotate(angle=pi/5000, axis=(1,1,0), origin=(0,0,0))
```

Oś obrotu jest zdefiniowana przez linię pomiędzy origin i origin+axis


VPython

scene.center

vp_center_1.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_a_10\vp_center_1.py

File Edit Format Run Options Windows Help

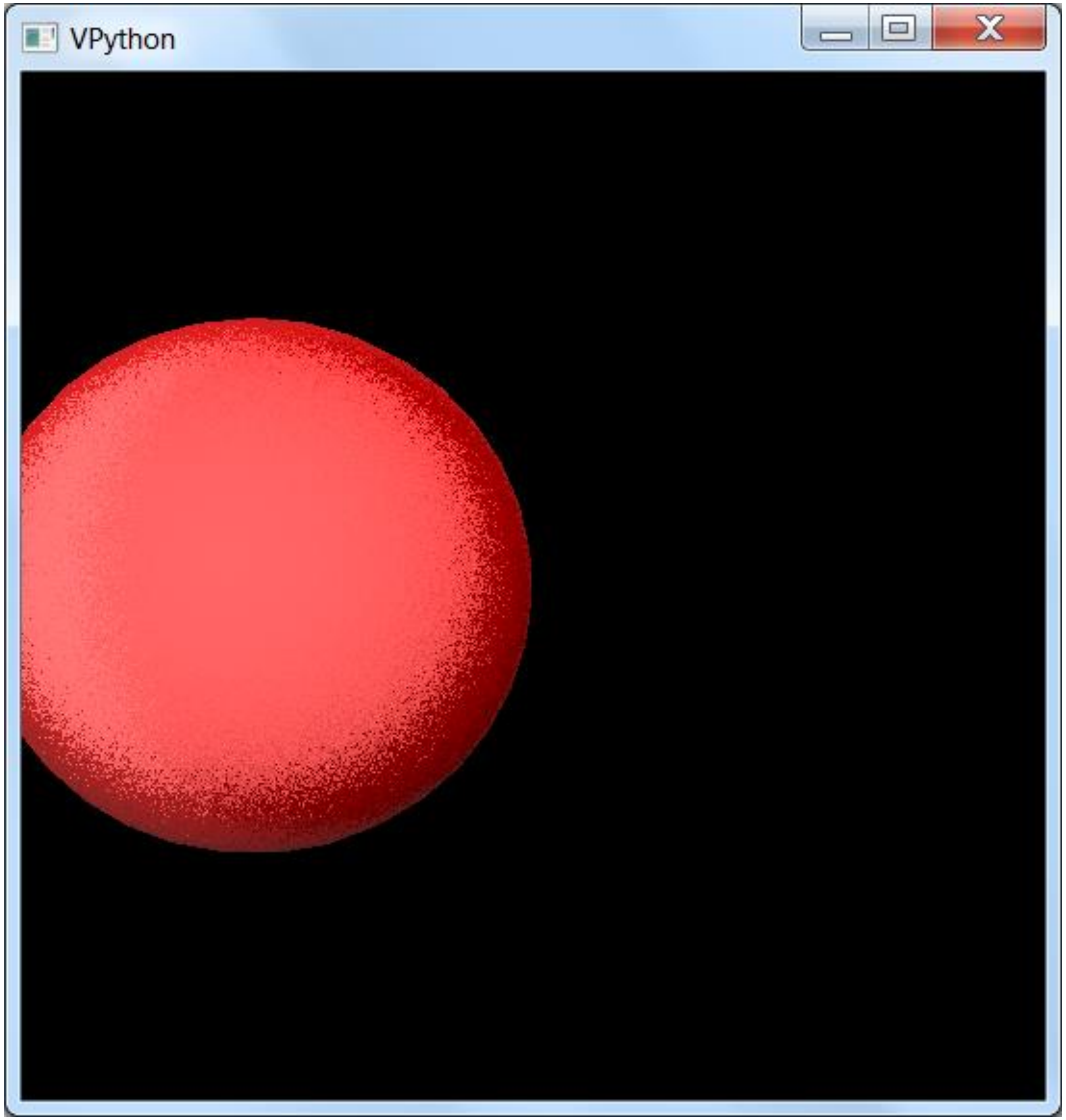
```python
from visual import *

scene = display(width=550, height=580, range=2)

ball = sphere(material=materials.silver, color=color.red)

print ball.pos

sleep(2)

scene.center = (1,0,0)

print ball.pos
```

kamera patrzy na (1,0,0)

Ln: 15 Col: 0

```
>>>
<0, 0, 0>
<0, 0, 0>
```

Ln: 9 Col: 4

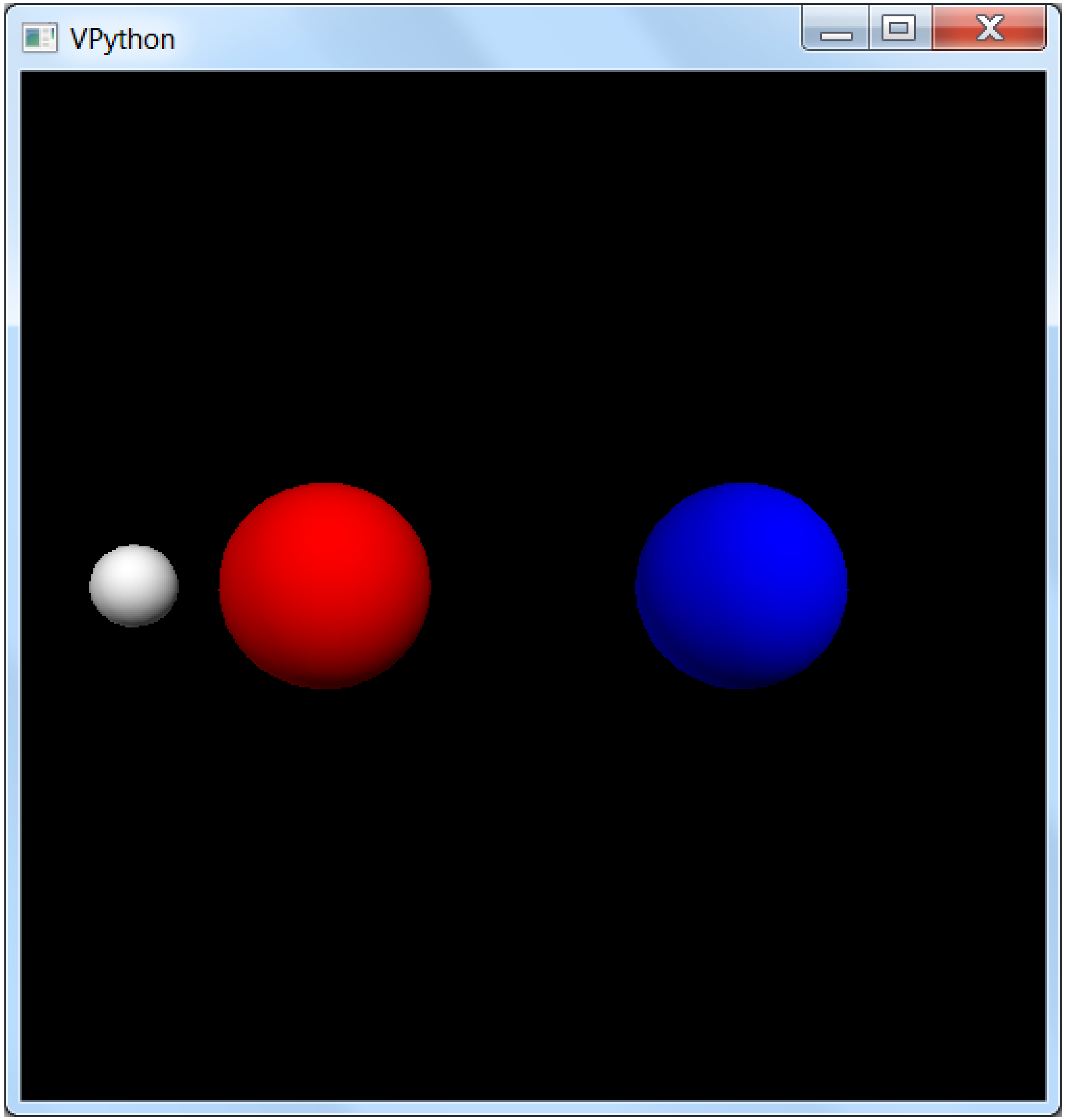

VPython

center

vp_center_2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_a_10\vp_center_2.py

```
from visual import *

scene = display(width=550, height=580, range=5)

ba1 = sphere(pos=(-2,0,0), color=color.red)
ba2 = sphere(pos=(2,0,0), color=color.blue)
ba3 = sphere(pos=(0,0,0), radius=0.4)

scene.autoscale = False

dt = 0.01
while True:
    rate(200)
    ba1.pos.x += 1*dt
    ba2.pos.x += 1*dt
    scene.center = (ba1.pos + ba2.pos)/2
```

Przydatne gdy kamerka ma śledzić środek masy


VPython

kontrola myszką

vp_no_zoom_spin.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_a_10\vp_no_zoom_spin.py

File Edit Format Run Options Windows Help

```
from visual import *

scene = display(width=550, height=580)

scene.userzoom = False    # True
scene.userspin = False    # True

sphere()
```

Ln: 11 Col: 0

label

vp_label_1.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_a_10\vp_label_1.py

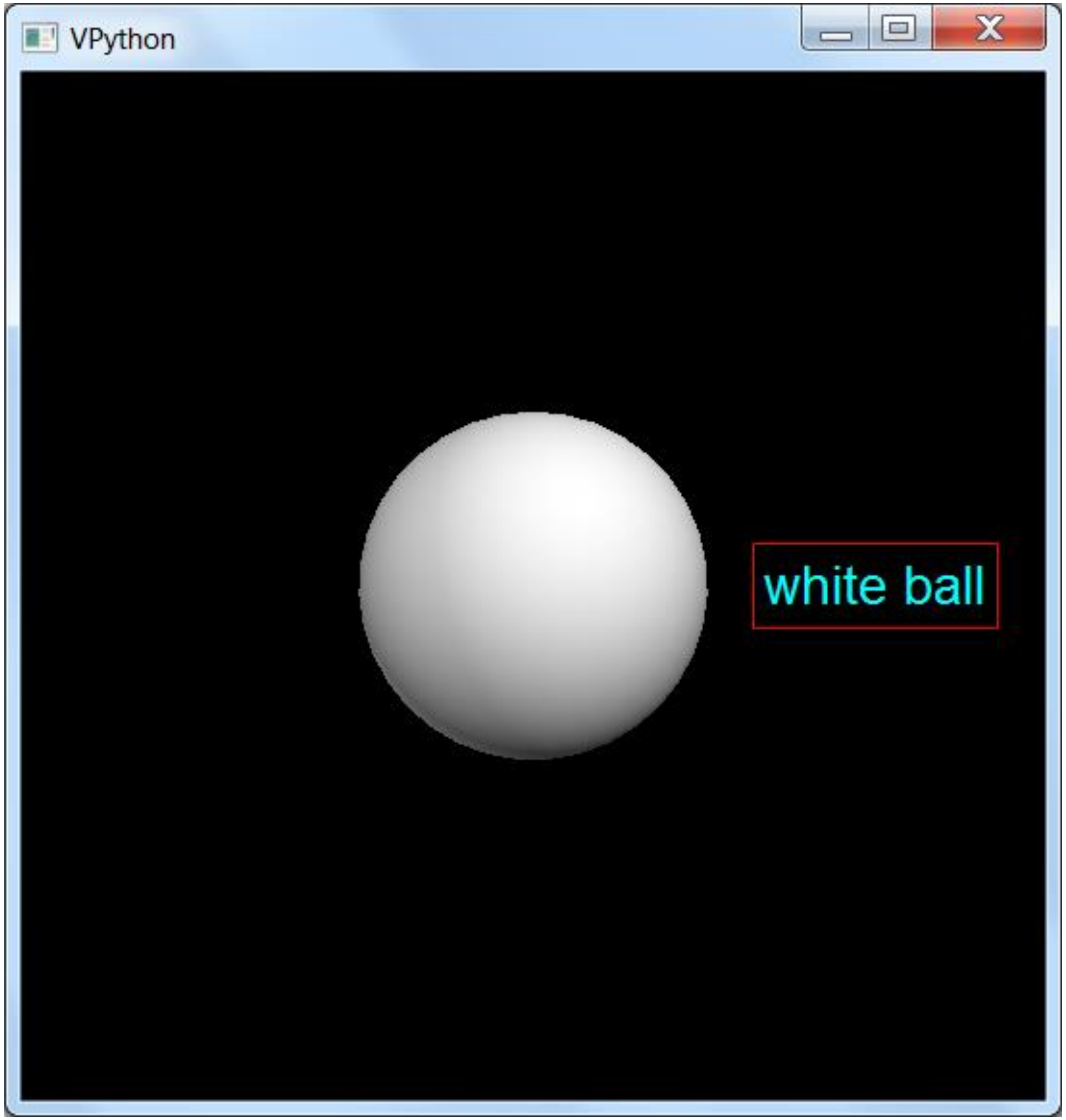
File Edit Format Run Options Windows Help

```
from visual import *

scene = display(width=550, height=580, range=3)

ball = sphere(pos=(0,0,0))

tt = label(pos=(2,0,0), text='white ball', height=20,
           color=color.cyan, linecolor=color.red)

# font = 'sans', 'serif', 'monospace'
# box = True / False
# border = 10
# opacity = 0.3
```

w pikselach

Ln: 15 Col: 0

label() zawsze patrzy się w ekran i rozmiar się nie zmienia gdy przybliżamy/obracamy kamerkę


VPython
white ball

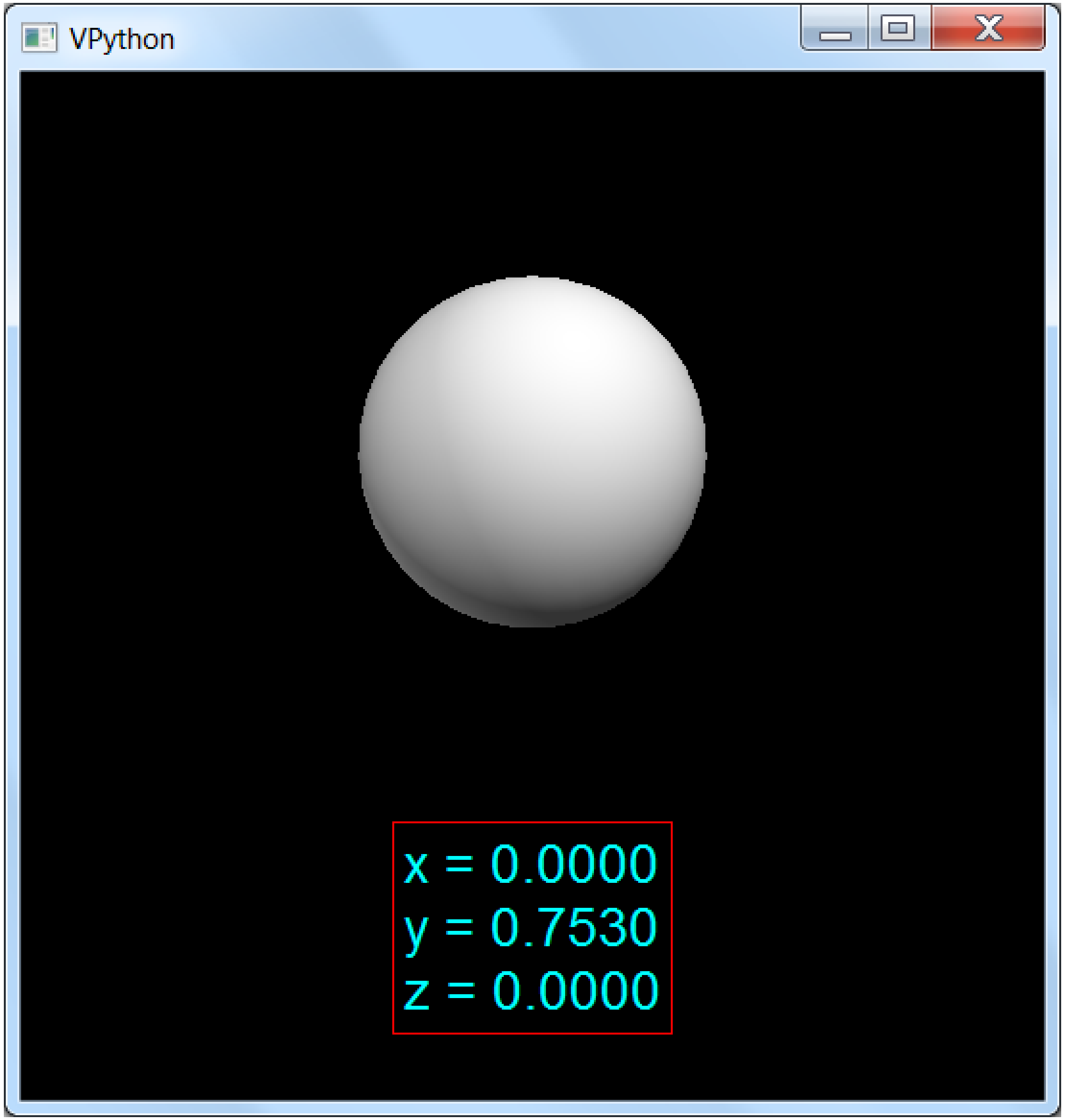
label

vp_label_2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_a_10\vp_label_2.py

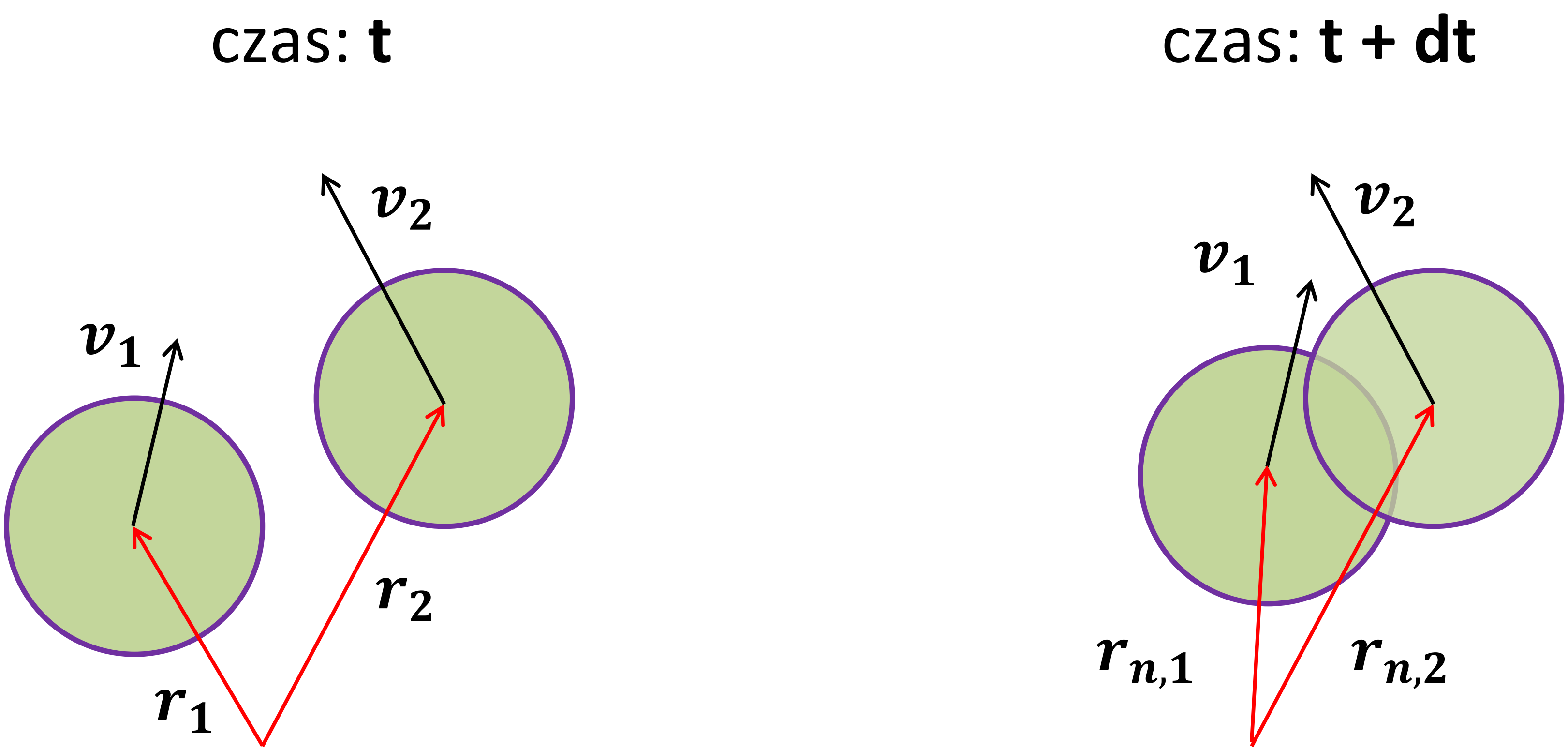
File Edit Format Run Options Windows Help

```
from visual import *

scene = display(width=550, height=580, range=3)

ball = sphere(pos=(0,0,0), radius=1)

tt = label(pos=(0,-2,0), text='ball', height=20,
           color=color.cyan, linecolor=color.red)

sleep(1)

dt = 0.01
while 1:
    rate(100)
    ball.pos.y += 0.1*dt
    tt.text = 'x = ' + '%.4f'%(ball.pos.x) + '\n' + \
              'y = ' + '%.4f'%(ball.pos.y) + '\n' + \
              'z = ' + '%.4f'%(ball.pos.z)
```

Ln: 21 Col: 0


VPython
x = 0.0000
y = 0.7530
z = 0.0000

Zderzenia kul



$$\boldsymbol{r}_{n,i} = \boldsymbol{r}_i + \boldsymbol{v}_i \cdot dt$$

Jeśli $|\boldsymbol{r}_{n,1} - \boldsymbol{r}_{n,2}| < (R_1 + R_2)$ to mamy zderzenie

R_1 , R_2 to promienie kul

Wracamy w czasie o $\boldsymbol{dt'}$ tak aby kule dotykały się powierzchniami

Łatwo policzyć dt':

$$\boxed{dt' = \frac{-b + \sqrt{\Delta}}{2a}}$$

$$a = |\boldsymbol{v_1} - \boldsymbol{v_2}|^2 \qquad a > 0, c < 0$$

$$b = -2\left(\boldsymbol{r_{n,1}} - \boldsymbol{r_{n,2}}\right) \circ (\boldsymbol{v_1} - \boldsymbol{v_2})$$

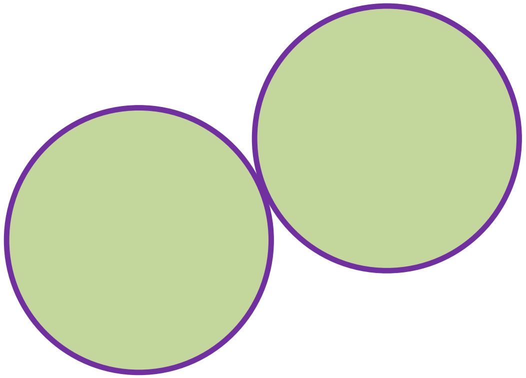
$$c = |\boldsymbol{r_{n,1}} - \boldsymbol{r_{n,2}}|^2 - \left(R_1 + R_2\right)^2$$

$$\Delta = b^2 - 4ac$$

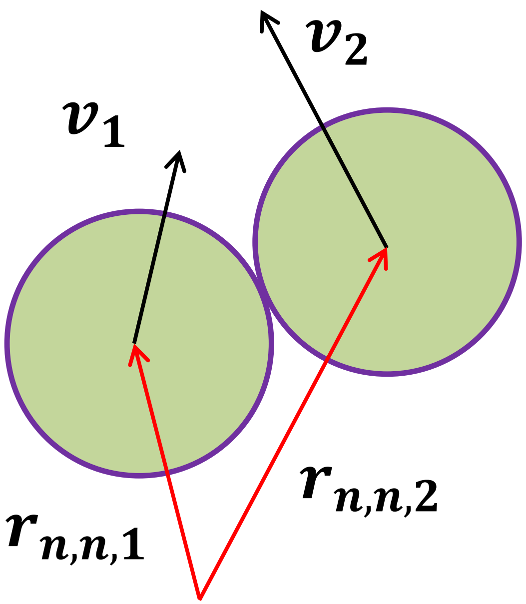
R_1 , R_2 to promienie

if a==0: continue

if $\Delta < 0$: continue

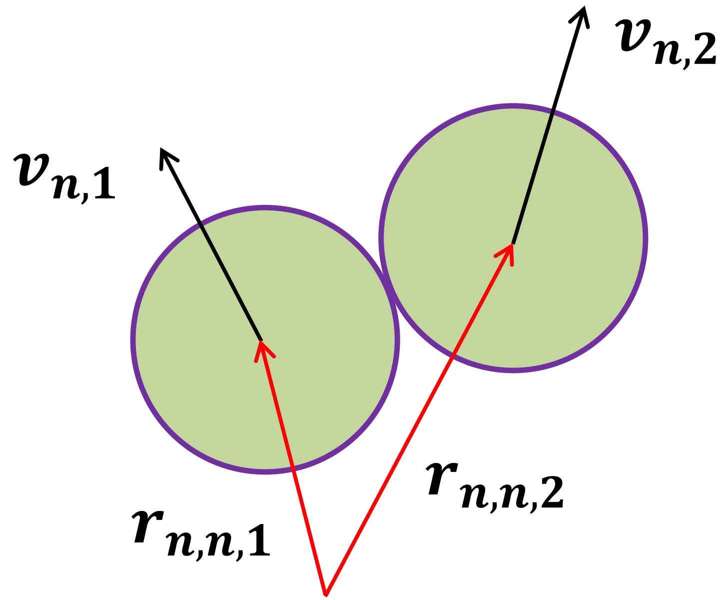


Po powrocie w czasie o $\boldsymbol{dt'}$ mamy:



$$\boldsymbol{r_{n,n,i}} = \boldsymbol{r_{n,i}} - \boldsymbol{v_i} \cdot dt'$$

Zderzenie: liczymy nowe prędkości

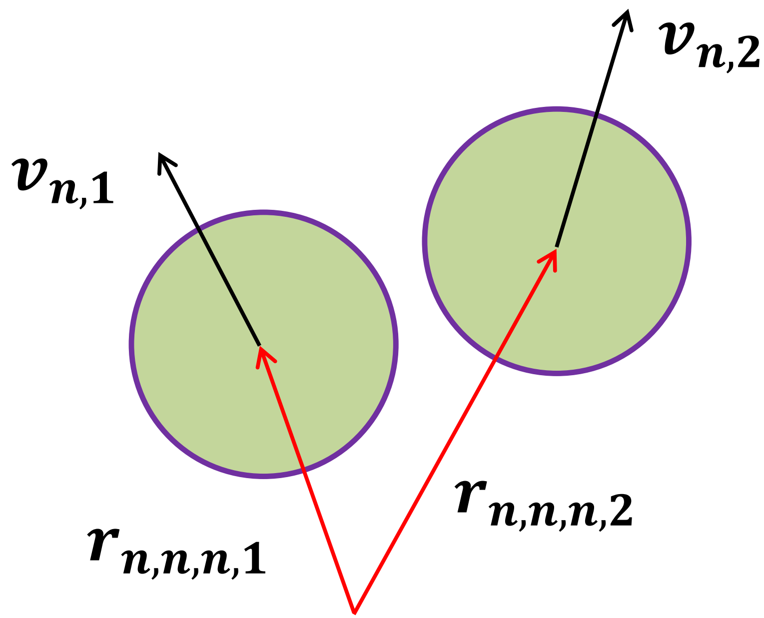


m_i – masa kuli

$$\boldsymbol{v}_{\boldsymbol{n},\mathbf{1}} = \boldsymbol{v}_{\mathbf{1}} - 2\frac{m_2}{m_1 + m_2}\left[(\boldsymbol{v}_{\mathbf{1}} - \boldsymbol{v}_{\mathbf{2}}) \circ \frac{\boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{1}} - \boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{2}}}{|\boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},1} - \boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},2}|}\right]\frac{\boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{1}} - \boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{2}}}{|\boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{1}} - \boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{2}}|}$$

$$\boldsymbol{v}_{\boldsymbol{n},\mathbf{2}} = \boldsymbol{v}_{\mathbf{2}} + 2\frac{m_1}{m_1 + m_2}\left[(\boldsymbol{v}_{\mathbf{1}} - \boldsymbol{v}_{\mathbf{2}}) \circ \frac{\boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{1}} - \boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{2}}}{|\boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},1} - \boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},2}|}\right]\frac{\boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{1}} - \boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{2}}}{|\boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{1}} - \boldsymbol{r}_{\boldsymbol{n},\boldsymbol{n},\mathbf{2}}|}$$

Idziemy do przodu o $\boldsymbol{dt'}$ (bo wcześniej się cofnęliśmy):



$$\boldsymbol{r_{n,n,n,i}} = \boldsymbol{r_{n,n,i}} + \boldsymbol{v_{n,i}} \cdot dt'$$