一般向け・プログラミング講座（９）

益田プログラミング研究会
Masuda Programming Research Society (MPRS)
https://maro-v.jp/~nagata-iics/
永田　武

９．アプリ開発（弾むボール）

(1)本日の内容　cat 01_contents_#10

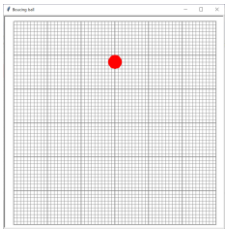
1．弾むボール（１個）bb_0.py(Boucing ball)

g bb_0.py &

```
# bb_0.py
from turtle import Turtle,Screen
import  random
from Tools import Tools

wn=Screen()
wn.tracer(False)
g = Tools()
g.grid(-300,300,600,600)
wn.tracer(True)
wn.title('Boucing ball')
#wn.bgcolor('black')
#wn.tracer(0)

balls = [Turtle()]
colors = ['red']
shapes = ['circle']

for ball in balls:
  ball.color(colors[0])
  ball.shape(shapes[0])
  ball.pu()
  ball.shapesize(2,2,1)
  x,y = 0,200
  ball.ht()
  ball.goto(x,y)
  ball.dy=0

gravity=0.1
try:
  while True:
    wn.update()
    for ball in balls:
      ball.st()
      ball.dy -= gravity
print('dy={},y={}'.format(int(ball.dy),
int(ball.ycor())))
      ball.sety(ball.ycor() + ball.dy)

      # check for collision
      if ball.ycor()<-300:
        ball.sety(-300)
        ball.dy *= -1

except KeyboardInterrupt:
  print('Interrupted!')
```

動作確認　p bb_0.py (起動 window で CTRL+C で停止)

```
dy=4,y=98
dy=4,y=103
dy=3,y=107
dy=3,y=110
dy=3,y=114
dy=3,y=118
dy=3,y=122
dy=3,y=125
dy=3,y=128
dy=3,y=132
dy=3,y=135
dy=3,y=138
```



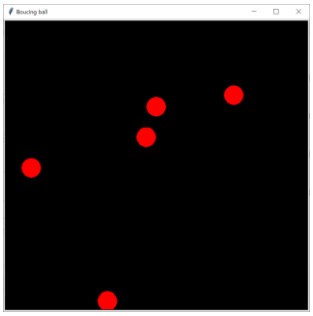
2．弾むボール（複数個）　bb_1.py

リスト balls に複数個のボールを入れる。

g bb_1.py &

```
# bb_1.py

wn.bgcolor('black')
wn.tracer(3)
N = 5
balls = []
for _ in range(N): balls.append(Turtle())
```

動作確認　p bb_1.py

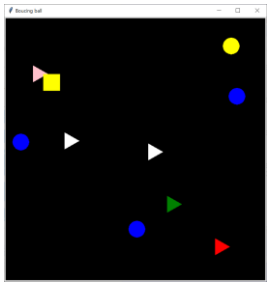


3．弾むボール（色・形）　bb_2.py

g bb_2 &

```
# bb_2.py

wn.tracer(5)
N = 10

colors=['red','yellow','blue','green','white','pink']
shapes=['circle','triangle','square']
ball.color(random.choice(colors))
ball.shape(random.choice(shapes))
```

動作確認　p bb_2.py



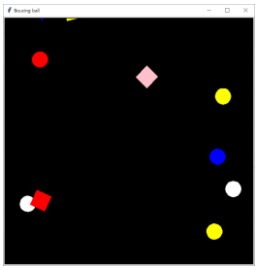
3．弾むボール（初期角度の変化・壁との衝突）
bb_3.py

g bb_3.py &

```
# bb_3.py

for ball in balls:
  …
  x = random.randint(-300,300)
  y = random.randint(200,400)

  ball.goto(x,y)
  ball.dx=random.randint(-3,3)
  ball.dy=0
  ball.da=random.randint(-5,5

try:
  while True:
    wn.update()
```

```
    for ball in balls:
      # check for collision
      if ball.ycor()<-300:
        ball.sety(-300)
        ball.dy *= -1
      if ball.xcor()<-300 or ball.xcor()>300:
        ball.dx *= -1
        ball.da *= -1
```

動作確認　**p bb_3.py**

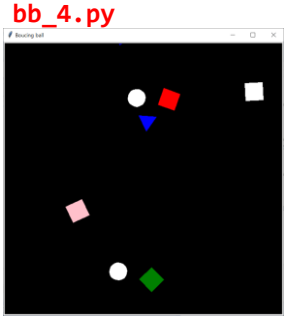


５．弾むボール（ボール同士の衝突）bb_4.py

```
g bb_4.py &
```

```
# bb_4.py


    # check for collisions between the balls
    for i in range(0,len(balls)):
      for j in range(i+1,len(balls)):
       if balls[i].distance(balls[j])<40:
         temp_dx = balls[i].dx
         temp_dy = balls[i].dy
         balls[i].dx = balls[j].dx
         balls[i].dy = balls[j].dy
         balls[j].dx = temp_dx
         balls[j].dy = temp_dy

…
```

動作確認　**p bb_4.py**



６．まとめ

- ボールの自由落下シミュレーションをするために，y 方向の速度（ball.dy）を ball.dy -= gravity で増加（ｙ座標が減少）させ，ｙ座標を更新（ball.ycor() + ball.dy）しています。
- タートルをボールとして扱い，複数のボールを扱くために**リスト**を用いています。
- また，色や形も複数**リスト**に登録しておき，その中からランダムに取り出すために，リストの random.choice()**メソッド**を用いています。
- 壁や床との衝突は，座標で判定し dx や dy の符号を反転しています。
- また，見え方を tracer()**メソッド**で調節しています。

アンケート（url.py）
http://192.168.xxx.yyy/q.php
xxx.yyy は指示します

連絡先　t.nagata.wp@gmail.com　永田　武
https://www.maro-v.jp/~nagata-iics/