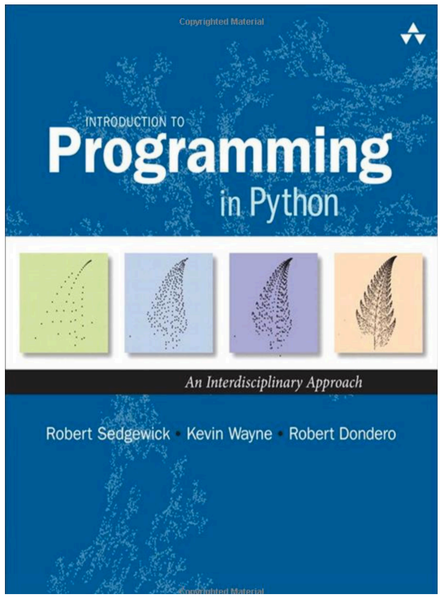
Parte II: Computación científica

Clase 12: Usando tipos de datos 3

Diego Caro
dcaro@udd.cl

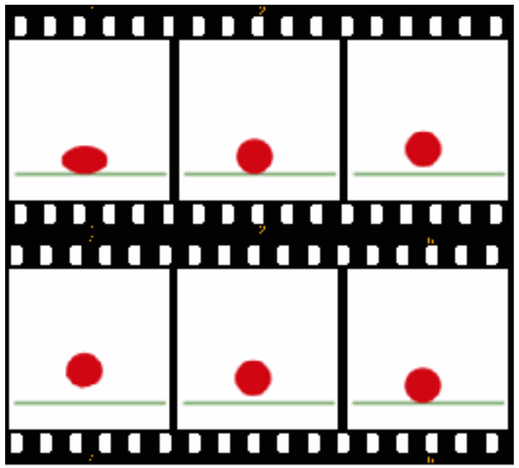


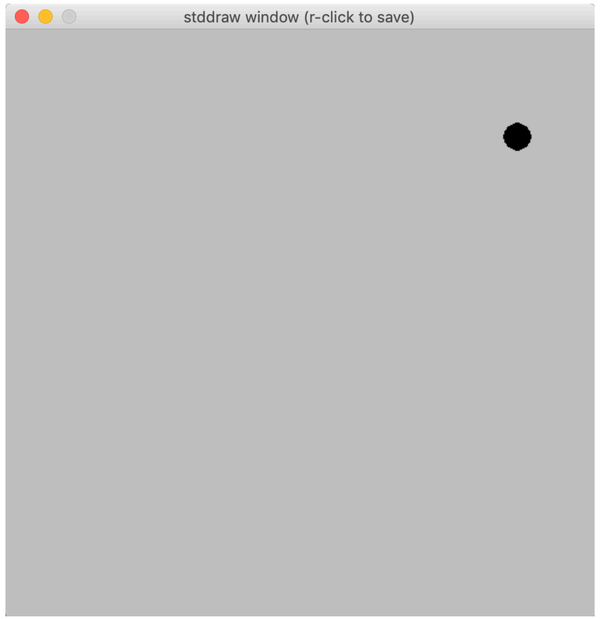
Basada en presentaciones oficiales de libro Introduction to Programming in Python (Sedgewick, Wayne, Dondero).

Disponible en https://introcs.cs.princeton.edu/python

Animación

- Movimiento puede simularse intercalando imágenes.
- Ejemplo: programar pelota que rebota en los bordes de la ventana.
- Estrategia: programar un ciclo infinito con
  - Cálculo de posición de pelota
  - Dibujar el fondo
  - Dibujar pelota (con la nueva posición)





Ejemplo: Pelota simple

```
 1 import stddraw
 2
 3 stddraw.setCanvasSize(500, 500)
 4 stddraw.setXscale(-1.0, 1.0)
 5 stddraw.setYscale(-1.0, 1.0)
 6
 7 radius = .05
 8 rx = .080
 9 ry = .060
10 vx = .015
11 vy = .013
12
13 while True:
14     # update position
15     rx = rx + vx
16     ry = ry + vy
17
18     # clear the background
19     stddraw.clear(stddraw.LIGHT_GRAY)
20
21     # draw the ball on the screen
22     stddraw.setPenColor(stddraw.BLACK)
23     stddraw.filledCircle(rx, ry, radius)
24
25     # copy buffer to screen
26     stddraw.show(0)
27     stddraw.pause(20)
```

Radio, posición y velocidad

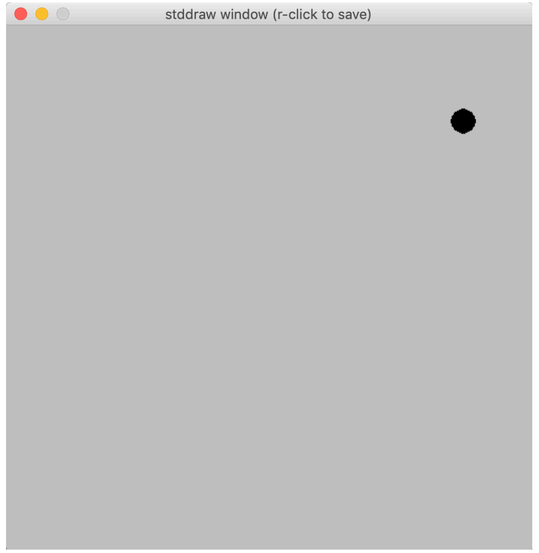
Actualización posición asumiendo velocidad constante (aceleración=0)

$$x_1 = x_0 + v_0 + \frac{1}{2}a_0t^2$$

Redibuja el fondo

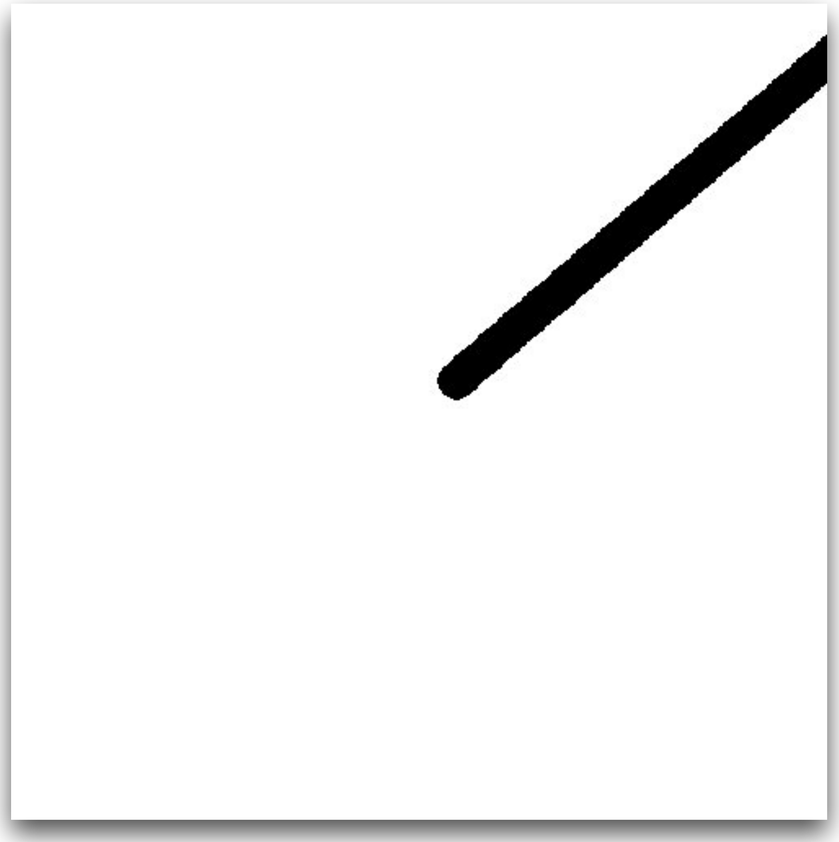
Dibuja pelota en nueva posición

Espera 20 milisegundos para dibujar siguiente frame



Preguntas

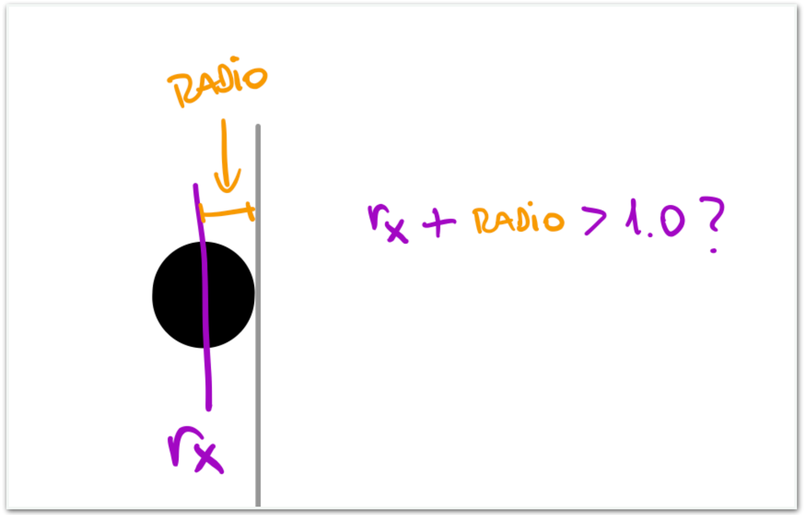
- ¿Qué sucede si no limpiamos el fondo?
  - No se borra lo que dibujamos en el ciclo anterior.



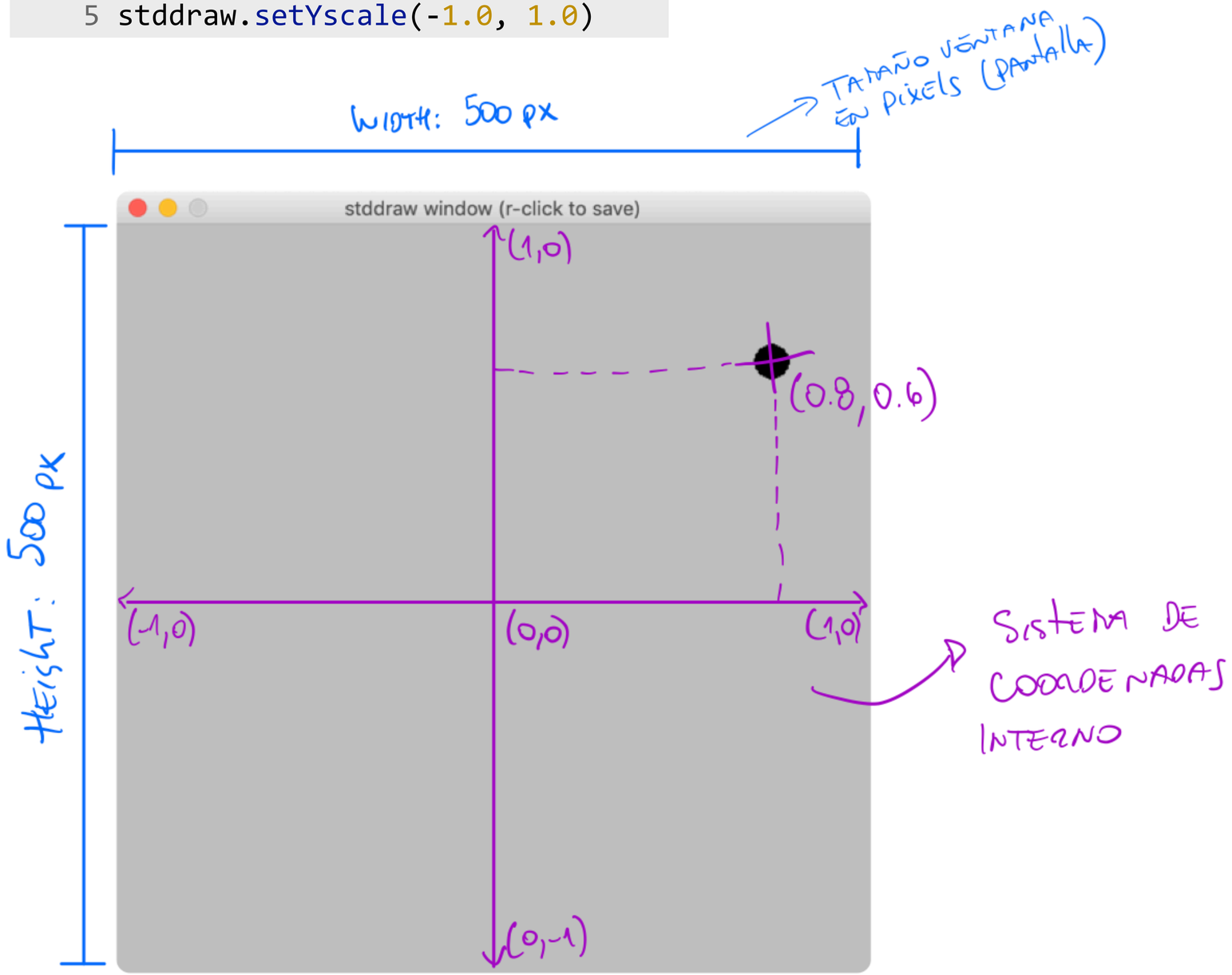
Preguntas

```
3 stddraw.setCanvasSize(500, 500)
4 stddraw.setXscale(-1.0, 1.0)
5 stddraw.setYscale(-1.0, 1.0)
```

- ¿Cómo podemos detectar que la pelota sale de la ventana?

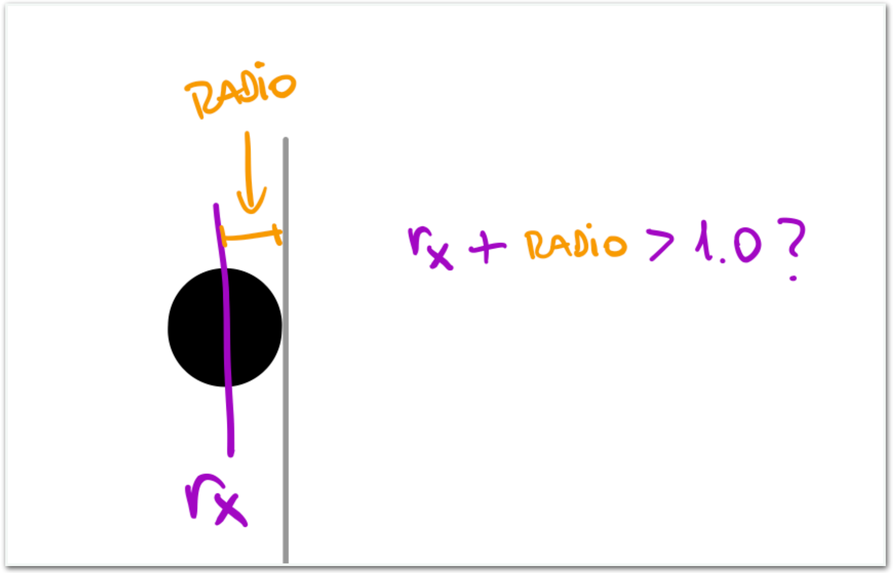



```
if abs(rx + vx) + radius > 1.0:
    print('choque con borde!')
if abs(ry + vy) + radius > 1.0:
    print('choque con borde!')
```



Preguntas

- ¿Cómo podemos hacer que la pelota rebote siguiendo las leyes de colisión elástica?
  - Asume que el borde de la ventana es de masa infinita y no se mueve.



Conservación del momento lineal:

$$m_1 v_1 + m_2 v_2 = m_1 u_1 + m_2 u_2$$

Conservación de la Energía (cinética):

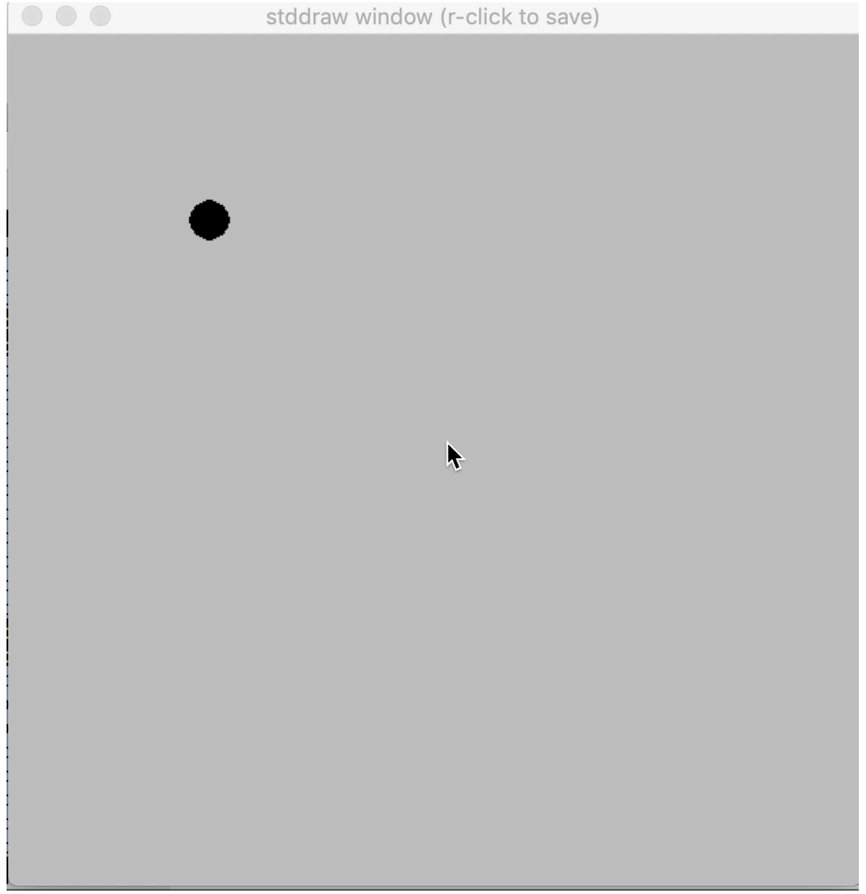
$$\frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 = \frac{1}{2} m_1 u_1^2 + \frac{1}{2} m_2 u_2^2$$

https://es.wikipedia.org/wiki/Choque_el%C3%A1stico

```
radius = .05
rx = .480
ry = .860
vx = .015
vy = .023

while True:
    if abs(rx + vx) + radius > 1.0:
        vx = -vx
    if abs(ry + vy) + radius > 1.0:
        vy = -vy

    rx = rx + vx
    ry = ry + vy

    stddraw.clear(stddraw.LIGHT_GRAY)

    stddraw.setPenColor(stddraw.BLACK)
    stddraw.filledCircle(rx, ry, radius)

    stddraw.show(0)
    stddraw.pause(20)
```

Colisión elástica con la pared

Actualización posición asumiendo velocidad constante (aceleración=0)



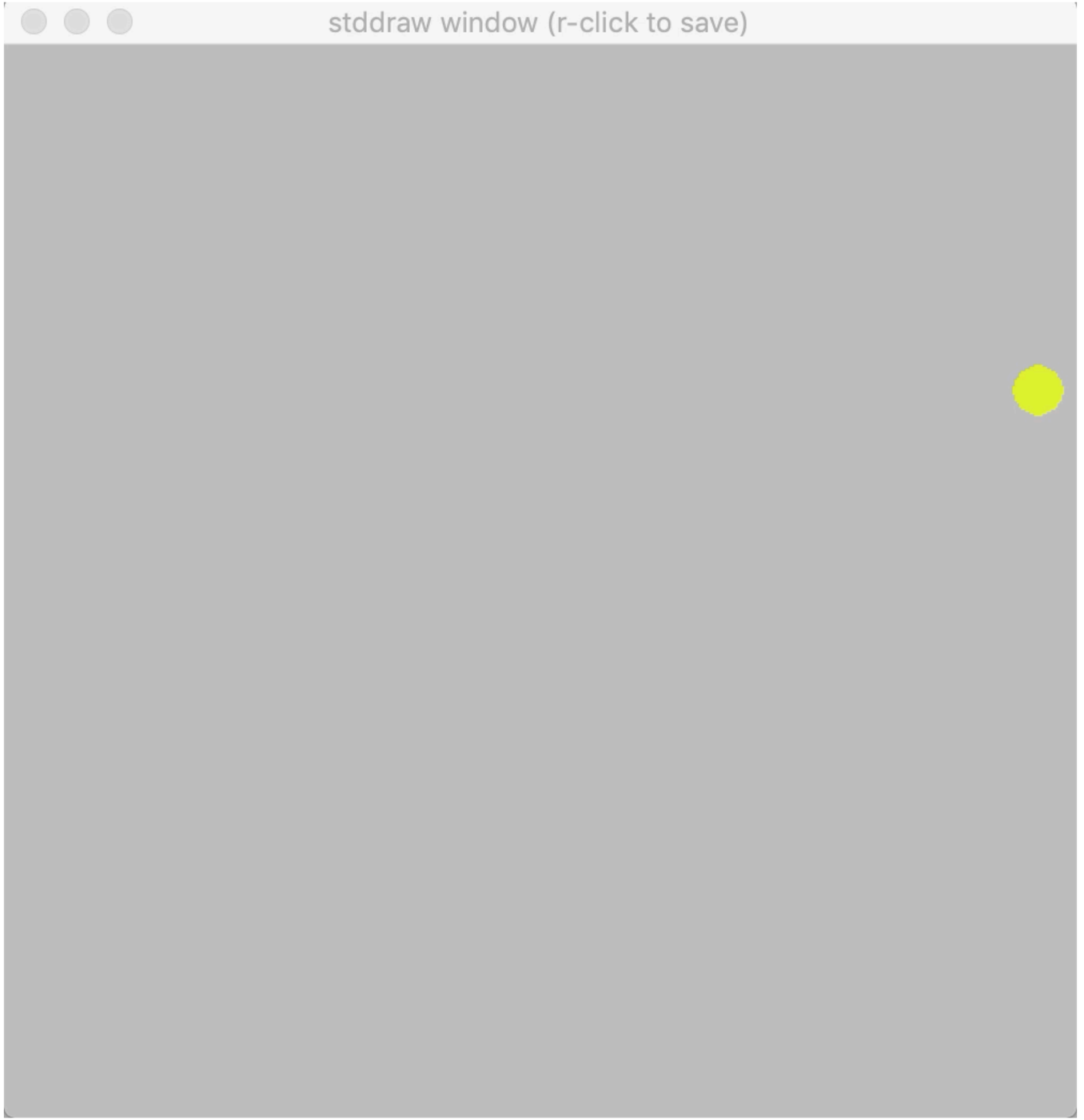
DEMO TIME

$ python3 bouncingball.py

¿Cómo hacer una pelota multicolor?

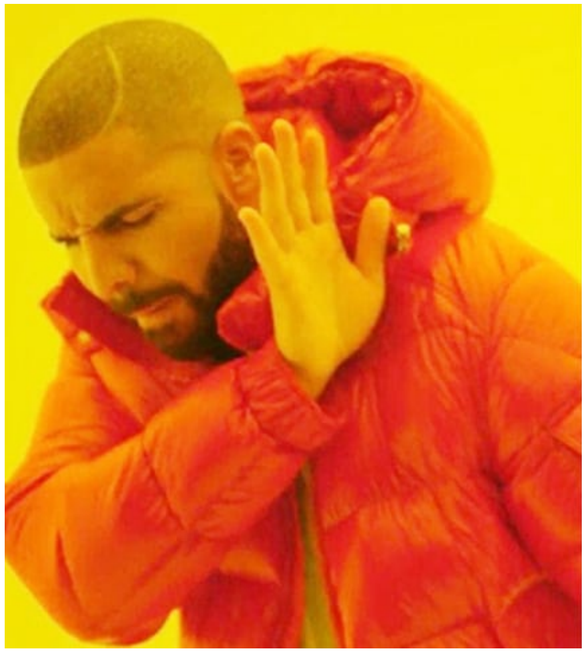
```
while True:
    if abs(rx + vx) + radius > 1.0:
        vx = -vx
    if abs(ry + vy) + radius > 1.0:
        vy = -vy

    rx = rx + vx
    ry = ry + vy

    stddraw.clear(stddraw.LIGHT_GRAY)

    stddraw.setPenColor(stddraw.BLACK)
    stddraw.filledCircle(rx, ry, radius)

    stddraw.show(0)
    stddraw.pause(20)
```

```
while True:
   if abs(rx + vx) + radius > 1.0:
        vx = -vx
    if abs(ry + vy) + radius > 1.0:
        vy = -vy

    rx = rx + vx
    ry = ry + vy

    stddraw.clear(stddraw.LIGHT_GRAY)

    r = randrange(256)
    g = randrange(256)
    b = randrange(256)
    c = Color(r, g, b)

    stddraw.setPenColor(c)
    stddraw.filledCircle(rx, ry, radius)

    stddraw.show(0)
    stddraw.pause(20)
```

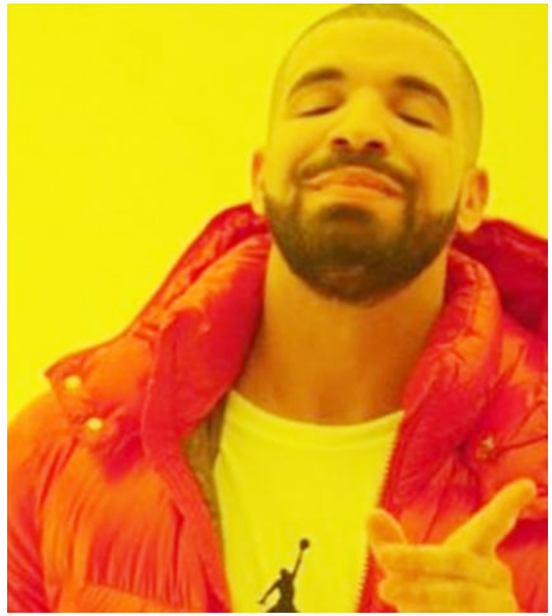

stddraw window (r-click to save)

¿Cómo mostrar 3 pelotas?

```
import
stddraw.setCanvasSize(500, 500)
stddraw.setXscale(0, 1.0)
stddraw.setYscale(0, 1.0)

radius = .05
rx = .480
ry = .860
vx = .015
vy = .023

radius2 = .05
rx2 = .480
ry2 = .860
vx2 = .030
vy2 = .063

radius3 = .05
rx3 = .480
ry3 = .860
vx3 = .01
vy3 = .013

while True:
    # bounce of wall according to elastic collision
    # update velocity
    if abs(rx + vx) + radius > 1.0:
        vx = -vx
    if abs(ry + vy) + radius > 1.0:
        vy = -vy
    if abs(rx2 + vx2) + radius2 > 1.0:
        vx2 = -vx2
    if abs(ry2 + vy2) + radius2 > 1.0:
        vy2 = -vy2
    if abs(rx3 + vx3) + radius3 > 1.0:
        vx3 = -vx3
    if abs(ry3 + vy3) + radius3 > 1.0:
        vy3 = -vy3
    # update position
    rx = rx + vx
    ry = ry + vy
    rx2 = rx2 + vx2
    ry2 = ry2 + vy2
    rx3 = rx3 + vx3
    ry3 = ry3 + vy3
    # clear the background
    stddraw.clear(stddraw.LIGHT_GRAY)
    # draw the ball on the screen
    stddraw.setPenColor(stddraw.BLACK)
    stddraw.filledCircle(rx, ry, radius)
    stddraw.filledCircle(rx2, ry2, radius2)
    stddraw.filledCircle(rx3, ry3, radius3)
    # display and pause for 20 ms
    stddraw.show(0)
    stddraw.pause(20)
```



Estrategia: crear una clase



```
class Ball:
    def __init__(self, rx, ry, vx, vy, radius, color):
        self.rx = rx
        self.ry = ry
        self.vx = vx
        self.vy = vy
        self.radius = radius
        self.color = color

    def update(self):
        if abs(self.rx + self.vx) + self.radius > 1.0:
            self.vx = -self.vx
        if abs(self.ry + self.vy) + self.radius > 1.0:
            self.vy = -self.vy

        self.rx = self.rx + self.vx
        self.ry = self.ry + self.vy

    def draw(self):
        stddraw.setPenColor(self.color)
        stddraw.filledCircle(self.rx, self.ry, self.radius)
```

Colisión elástica con la pared

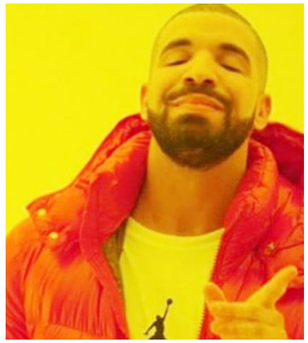
Actualización posición

Dibujar!

Código cliente para una pelota

```
ball = Ball(.480, .860, .015, .023, .05, stddraw.BLACK)

while True:
    # update velocity
    ball.update()
    # clear the background
    stddraw.clear(stddraw.LIGHT_GRAY)

    # draw the ball on the screen
    ball.draw()

    # copy buffer to screen
    stddraw.show(0)
    stddraw.pause(20)
```

Solución: crear una lista de objetos Ball



```
1  import stddraw
2  from ball import Ball
3
4  stddraw.setCanvasSize(500, 500)
5
6  stddraw.setXscale(-1.0, 1.0)
7  stddraw.setYscale(-1.0, 1.0)
8
9  balls = [
10     Ball(.480, .860, .015, .023, .05, stddraw.BLACK),
11     Ball(.480, .860, .030, .046, .05, stddraw.BLUE),
12     Ball(.180, .260, .040, .026, .05, stddraw.GREEN)
13 ]
14
15 while True:
16     # update velocity
17     for b in balls:
18         b.update()
19
20     # clear the background
21     stddraw.clear(stddraw.LIGHT_GRAY)
22
23     # draw the ball on the screen
24     for b in balls:
25         b.draw()
26
27     # copy buffer to screen
28     stddraw.show(0)
29     stddraw.pause(20)
```

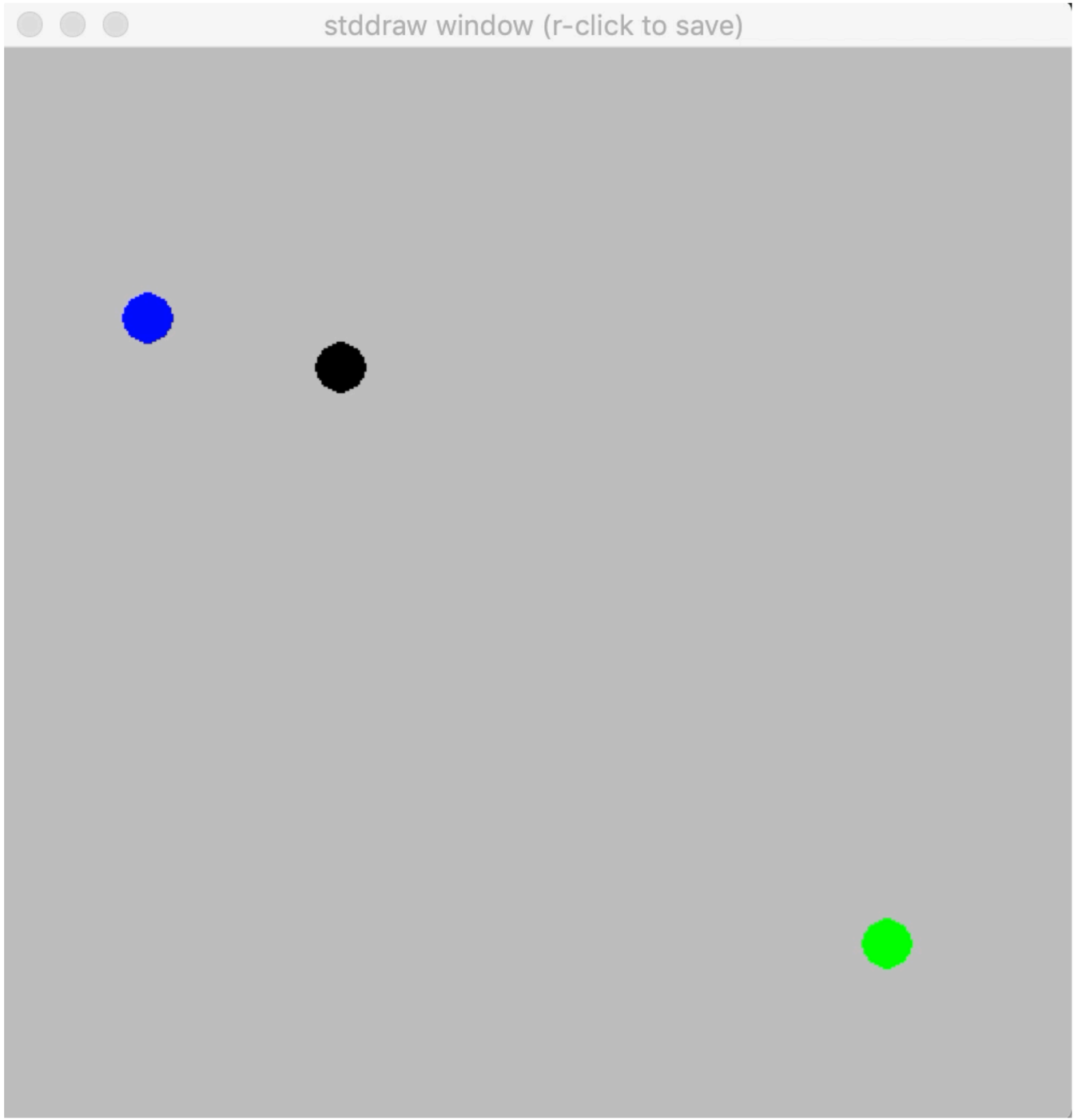
Módulo ball.py

```
1  import stddraw
2
3  class Ball:
4      def __init__(self, rx, ry, vx, vy, radius, color):
5          self.rx = rx
6          self.ry = ry
7          self.vx = vx
8          self.vy = vy
9          self.radius = radius
10         self.color = color
11
12     def update(self):
13         """
14         Bounce of wall according to elastic collition and
15         update velocity.
16         """
17         if abs(self.rx + self.vx) + self.radius > 1.0:
18             self.vx = -self.vx
19         if abs(self.ry + self.vy) + self.radius > 1.0:
20             self.vy = -self.vy
21
22         self.rx = self.rx + self.vx
23         self.ry = self.ry + self.vy
24
25     def draw(self):
26         stddraw.setPenColor(self.color)
27         stddraw.filledCircle(self.rx, self.ry, self.radius)
```

Colisión

Actualización posición

Dibujar!


stddraw window (r-click to save)

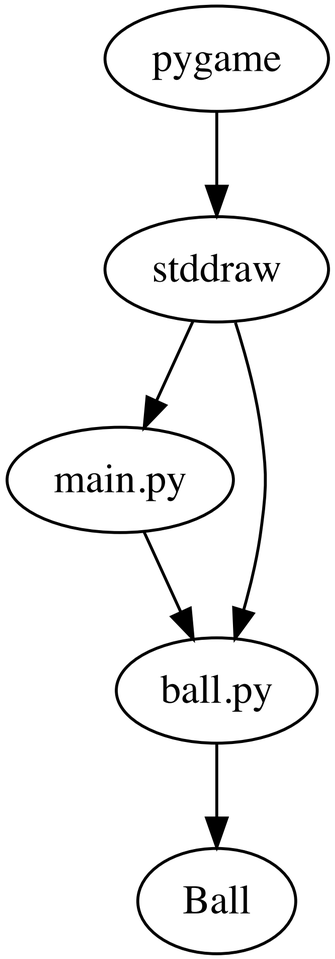
Usando el teclado

```
balls = [
    Ball(.480, .860, .015, .023, .05, stddraw.BLACK),
    Ball(.480, .860, .030, .046, .05, stddraw.BLUE),
    Ball(.180, .260, .040, .026, .05, stddraw.GREEN)
]

while True:
    # get keystrokes
    if stddraw.hasNextKeyTyped():
        k = stddraw.nextKeyTyped()
        if k == stddraw.K_UP:
            for b in balls: b.increase_speed(0.1, 0.1)
        elif k == stddraw.K_DOWN:
            for b in balls: b.increase_speed(-0.1, -0.1)

    # update velocity
    for b in balls: b.update()

    # clear the background
    stddraw.clear(stddraw.LIGHT_GRAY)

    # draw the ball on the screen
    for b in balls: b.draw()

    # copy buffer to screen
    stddraw.show(0)
    stddraw.pause(20)
```

Códigos para teclas en https://github.com/josiest/pygtails/blob/master/docs/pygstants.rst

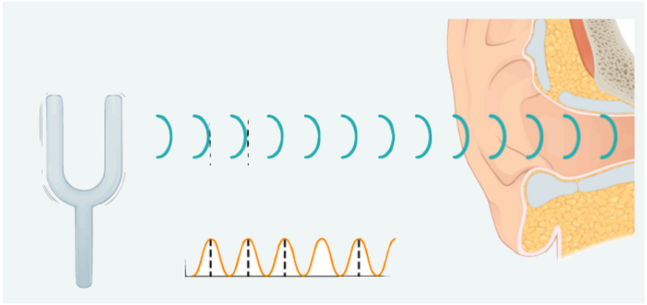
Keycode Name	Ascii	Description
K_BACKSPACE	\b	backspace
K_TAB	\t	tab
K_CLEAR		clear
K_RETURN	\r	return
K_PAUSE		pause
K_ESCAPE	^[	escape
K_SPACE		space
K_UP		up arrow
K_DOWN		down arrow
K_RIGHT		right arrow
K_LEFT		left arrow

Dependencia entre módulos



Nota: módulo pygame se escapa del ámbito de este curso. Usaremos la biblioteca introcs disponible en https://github.com/diegocaro/introcs

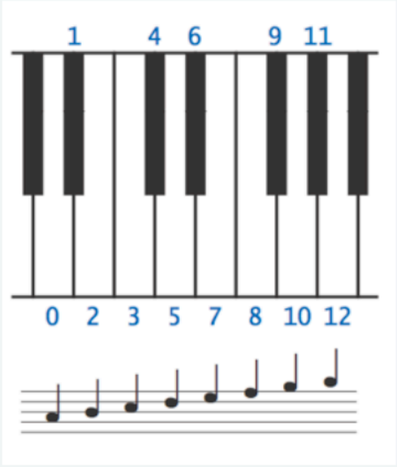
Sonido



- El **sonido** es la percepción de la vibración de moléculas.
- Un **tono musical** es un sonido periódico.
- Un **tono puro** es una onda sinusoidal.

Western musical scale

- Concert A is 440 Hz.
- 12 notes, logarithmic scale.



pitch	*i*	*frequency* $(440*2^{i/12})$	*sinusodial waveform*
A	0	440	
A♯ / B♭	1	466.16	
B	2	493.88	
C	3	523.25	
C♯ / D♭	4	554.37	
D	5	587.33	
D♯ / E♭	6	622.25	
E	7	659.26	
F	8	698.46	
F♯ / G♭	9	739.99	
G	10	783.99	
G♯ / A♭	11	830.61	
A	12	880	

http://www.cs.princeton.edu/courses/archive/fall18/cos126/lectures/CS.5.Functions.pdf

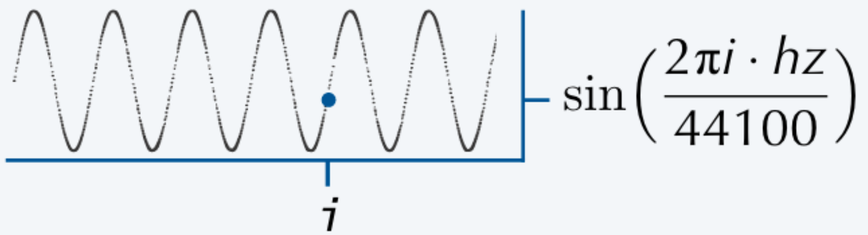
Audio digital

- Para representar una onda en el computador se debe "sample" en intervalos regulares.
- El computador solo puede representar números, "sampling" permite transformar la onda a una serie de números.

	samples/sec	*samples*	*sampled waveform*
1/40 second of concert A	5,512	137	
	11,025	275	
	22,050	551	
CD standard →	44,100	1102	

Hola mundo módulo stdaudio

```
 1 import math
 2 import stdaudio
 3 import sys
 4
 5 def tone(hz, duration):
 6     n = int(44100 * duration)
 7     note = [0.0]*(n+1)
 8     for i in range(n+1):
 9         note[i] = math.sin(2.0 * math.pi * i * hz / 44100)
10     stdaudio.playSamples(note)
11
12 hz = float(sys.argv[1])
13 duration = float(sys.argv[2])
14 tone(hz, duration)
```



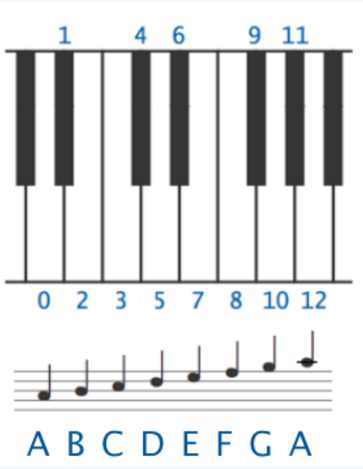

```
python3 playthatnote.py 440.0 3.0
python3 playthatnote.py 880.0 3.0
python3 playthatnote.py 220.0 3.0
python3 playthatnote.py 494.0 3.0
```

Reproducir canción

```
 1 import math
 2 import stdio # this is new!
 3 import stdaudio
 4
 5 SPS = 44100
 6 CONCERT_A = 440.0
 7 NOTES_ON_SCALE = 12.0
 8
 9 while not stdio.isEmpty():
10     pitch = stdio.readInt()
11     duration = stdio.readFloat()
12     hz = CONCERT_A * (2.0 ** (pitch / NOTES_ON_SCALE))
13     n = int(SPS * duration)
14     note = [0.0]*(n+1)
15     for i in range(n+1):
16         note[i] = math.sin(2.0 * math.pi * i * hz / SPS)
17     stdaudio.playSamples(note)
18
19 stdaudio.wait()
```

Lee desde teclado y convierte automáticamente a entero/float.

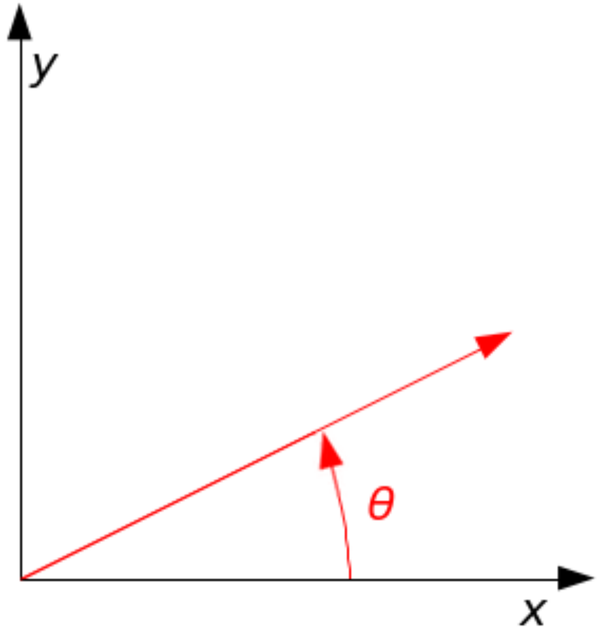
```
$ head elise.txt
7 .125
6 .125
7 .125
6 .125
7 .125
2 .125
5 .125
3 .125
0 .25
```



bouncingball-deluxe.py
balldeluxe.py
playthatnote.py

Rotaciones

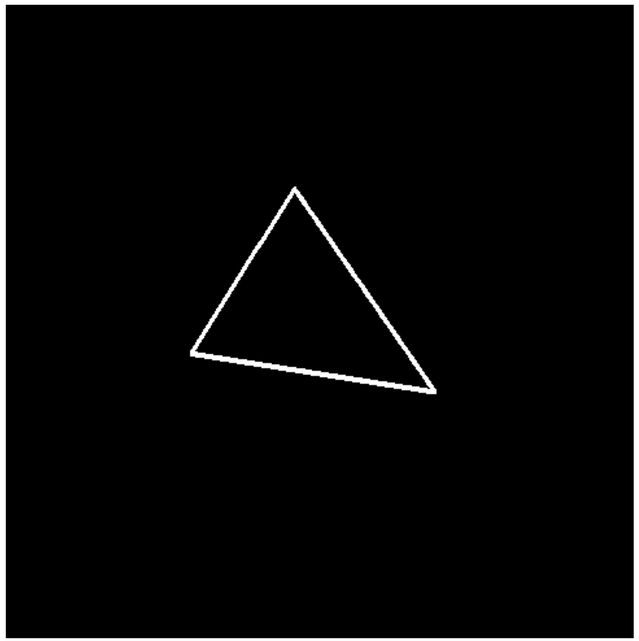
$$x' = x\cos\theta - y\sin\theta\,,$$
$$y' = x\sin\theta + y\cos\theta\,.$$

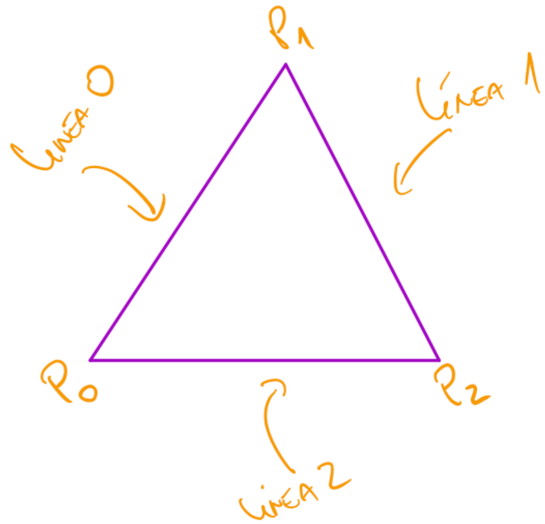


https://en.wikipedia.org/wiki/Rotation_matrix

Ejemplo: rotando una nave espacial

- La nave espacial es un triángulo.
  - ... pero stddraw no dibuja triángulos! 😱
  - Podemos dibujarla usando tres líneas
- Luego rotamos los 3 puntos del triángulo, y boom!





```python
import stddraw
from math import cos, sin

stddraw.setCanvasSize(500, 500)

stddraw.setXscale(-1.0, 1.0)
stddraw.setYscale(-1.0, 1.0)

points = [(-0.3, -0.3), (0, 0.4), (0.3, -0.3)]
n = len(points)
angle = 0.1 # in radians

while True:
    stddraw.clear(stddraw.BLACK)
    stddraw.setPenColor(stddraw.WHITE)

    # calculate rotations
    for i in range(n):
        p = points[i]
        newx = p[0]*cos(angle) - p[1]*sin(angle)
        newy = p[0]*sin(angle) + p[1]*cos(angle)
        points[i] = (newx, newy)

    # display triangle
    for i in range(n):
        stddraw.line(points[i][0],points[i][1], points[(i+1)%n][0], points[(i+1)%n][1])

    # copy buffer to screen
    stddraw.show(0)
    stddraw.pause(20)
```