1 Adding Sound

For projects 10 and 11, you may want to add sound effects to your programs. I can teach you how to do so on a Mac - by using afplay to play a .wav file. There are lots of .wav files available on the internet. I found one called boing2.wav.

From the command line, I can do so like this
afplay boing2.wav

Notice that afplay doesn’t finish until the sound has finished. In our programs, we will want to move on to the next time step in the game before the sound has finished. That means we will want to play it in a different thread of execution - we want it to play in parallel with our other code. We can do this by telling the operating system to run afplay, but do so so in another thread. We do that by adding the & to the command line.
afplay boing2.wav &

This is all well and good you say, but what about Python? How do I do this from a Python program? Python has a package called os that will allow you to execute code as if you were using the command line. In Python, I import the os package and then I write the line
os.system(‘afplay boing2.wav’)
2 Design principles in Project 10

We briefly went over the code and some of its design. Here are the highlights:

- The purpose of the Thing class is to group together all of the code that manages fields that are common to all the non-rotating objects (e.g. the position, velocity, and mass).

- Each non-rotating object puts all of its Graphics objects in the vis field in the init method. This means that methods that change aspects of those graphics objects (e.g. their color) can just loop through the vis list and make those changes right away. This means there is no need for a color field.

- Each rotating object has to regenerate its list of Graphics objects, so we need more fields. For example, if we were to set the color of a rotating object, we would need to set the value of the color field, then redraw the object. This redraw will cause the object to be rendered again and then the color can be applied. This same color field will need to be used every time the object is rendered.

3 Making the StopWatch Scene more interesting

Let’s add a rotating block, a ball, and a leaning tower (a rotated block) to the stopwatch code. Now, it is the user’s job to start the block spinning (by pressing b). A ball will fall, if the spinning block hits the ball and directs it towards the leaning tower, then you will hear a boing sound and the simulation will pause. The timer text will show how long it took for the ball to hit the leaning tower. You unpause the simulation by pressing s. If the ball goes out of range, you need to restart the program because there isn’t any reset code in it.

Play it several times to see how early or late you can make it hit the tower.

Here is the code. It assumes RotatingLine and RotatingBlock are in rotation.py and that Text is in physical_objects.py.
import time
import math
import physics_objects as pho
import rotation
import graphics as gr
import collision as coll
import os

def main():

    win = gr.GraphWin('timer', 500, 500, False)

    handlength = 5
    handAnchor = (10, 35)
    ypos = handAnchor[1]
    xpos = handAnchor[0] + handlength/2.0
    hand = rotation.RotatingLine(win, x0=xpos, y0=ypos, length=handlength)
    hand.setAnchor(handAnchor)
    hand.setAngle(90)
    hand.draw()

    text = pho.Text(win, 10, 45, "")
    text.draw()

    # Draw the ticks
    for degrees in range(0, 360, 5):
        second = (degrees % 360) / 60
        drad = degrees / 180.0 * math.pi
        x0 = handAnchor[0] + 1.2 * handlength * math.cos(drad)
        y0 = handAnchor[1] + 1.2 * handlength * math.sin(drad)
        tick = rotation.RotatingLine(win, x0=x0, y0=y0, length=2, Ax=x0, Ay=y0)
        tick.setAngle(degrees)
        tick.draw()

    spinner = rotation.RotatingBlock(win, 35, 25, 20, 10)
    spinner.draw()
    spinner.setAngle(300)
    spinner.setAnchor((35, 25))

    leaner = rotation.RotatingBlock(win, 10, 10, 20, 3)
    leaner.setAnchor((10, 10))
    leaner.setAngle(300)
    leaner.draw()

    floor = pho.Floor(win, 0, 0, 50, 5)
floor.draw()

ball = pho.Ball( win, 25, 45 )
ball.setAcceleration((0, -10))
bball.draw()

pausing = True
spinner.pausing = True
spinner.setRotVelocity(0)

degrees = 0
dt = 0.01
while True:
    key = win.checkKey()
    if key == 'q':
        break

    if pausing:
        if key == 's':
            pausing = False
            time.sleep(dt)
            continue

    if spinner.pausing:
        if key == 'b':
            spinner.setRotVelocity(360)
            spinner.pausing = False
        else:
            if key == 'b':
                spinner.setRotVelocity(0)
                spinner.pausing = True

    if not spinner.pausing:
        spinner.update(dt)

    if coll.collision(ball, spinner, dt) or coll.collision(ball, floor, dt):
        pass
    elif coll.collision(ball, leaner, dt):
        os.system("afplay boing2.wav &")
        pausing = True
    else:
        ball.update(dt)

hand.rotate(-0.1)
degrees += 0.1
minute = \textbf{int}(\text{degrees})/360 \\
second = (\textbf{int}(\text{degrees})\%360)/6 \\

text.setText("%02d:%02d:%02d" % (minute, second, degrees\%360)) \\
if key == 's': \\
    pausing = True \\
win.update() \\

time.sleep(dt) \\

win.close() \\

if \_\_name\_\_ == "\_\_main\_\_": \\
main()