1 Administrative Topics

- There is an info session about c.s. courses tomorrow evening in the robotics lab (Davis 122). Please come Thursday night at 7pm if you plan to take any future c.s. courses.

- Hopefully, I will have the quiz to return today.

2 Adding Rotation to the Physics Simulations

We are going to make it possible to rotate our objects. There is no rotate method associated with the Graphics objects, so we will need to determine where each point will need to move to, undraw any currently drawn version of the object, then draw a new one with in the rotated position.

In lab, you will be writing code to make a Line object that will rotate. Today, I am going to show you some of that code and we will together write code that uses the rotating line.

In the figure below, I show a pair of x- and y-axes with a line starting at the original. That line is at an angle of 45 degrees (we will be assuming units of degrees).
2.1 Basic Approach

When we want to draw a line in our physics simulation, we specify the position in the typical model coordinates (the origin is at the bottom left corner). We specify a point that we want to use as our anchor point, and we specify an angle that we want to rotate the line about the anchor point. I.e we take the whole line, and move it so the anchor point is at the origin. Then we rotate the line in its new location (we use sines and cosines). Finally, we “undo” the move so the anchor point is back where we want to draw it. And we have our new model coordinates for the ends of the line. Below are two examples. In the first, the anchor point (marked by an X) is at the left end of the line. In the second, it is in the middle of the line.
Move the line so the anchor point is at the origin

Move the line back to where it was

Rotate line
2.2 Labeled with fields from code

To keep track of all the positions, we use several fields, including

- pos: a tuple that stores the x and y positions of the center of the unrotated line.
- anchor: a tuple that stores the x and y positions of the anchor point
- points: a list of tuples that store the end points of the unrotated line, assuming that the center is at the origin. (this is to make our computation easier)
• angle: a float value indicating how much the line should be rotated, in degrees.

Below, I show the values of the end points as we determine what the final positions will be in the method that determines them an makes the Line object (it is called render). The values of the fields do not change. The labels on the lines represent variables in the render method. For details, see the lab instructions.
3 User interaction with Keys

We have already learned how to let the user interact with our simulations with the mouse. There are two GraphWin methods - getMouse, which waits until the user clicks the mouse in the window, and checkMouse, which returns None if the user has not “recently” clicked in the window (actually, it checks all the way back to the last time checkMouse was called) and position if it has been clicked.

There are two analogous methods – getKey and checkKey. If a key has been pressed, these methods return a string indicating which key was pressed (e.g. if the “s” key was pressed it returns “s”). If you call checkKey and no key has been pressed since you last called it, then it will return the empty string.

Our example today will control a stop-watch. “s” means “pause” or “un-pause” and “q” means “quit”. In our main loop, we will call checkKey and carry out appropriate actions if the value is s or q. Since we are paying attention to two different key values, we need to call checkKey once and store the result in a variable. We can’t just call checkKey twice in the loop. Why not?

4 Main Code

# Animates a stop-watch. You can make it stop and go
# by pressing the s key. You can make the program end
# by pressing the q key.
def main():
    win = gr.GraphWin('line thingy', 500, 500, False)
    handlength = 10
    handAnchor = (25, 25)
    hand = RotatingLine(win, x0=30, y0=25, length=handlength)
    hand.setAnchor(handAnchor)
    hand.setAngle(90)
    hand.draw()

    # Draw the ticks
    for degrees in range(0, 360, 5):
        second = (degrees % 360) / 60
drad = degrees / 180.0 * math.pi
x0 = handAnchor[0] + 1.1 * handlength * math.cos(drad)
y0 = handAnchor[1] + 1.1 * handlength * math.sin(drad)
tick = RotatingLine(win, x0=x0, y0=y0, length=2, Ax=x0, Ay=y0)
tick.setAngle(degrees)
tick.draw()

pausing = False

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

    if pausing:
        if key == 's':
            pausing = False
            time.sleep(0.1)
        continue

    minute = degrees / 360
    second = (degrees % 360) / 6
    hand.rotate(-1.0)

    if key == 's':
        pausing = True

    win.update()

    time.sleep(0.01)
    degrees += 1

win.close()