CS151 Fall 2012: 3D Turtle Notes

Stephanie R Taylor

November 28, 2012

1 3D Turtle

The 3-D turtle uses a 3D- coordinate system in which the positive X-axis extends to the right, the positive Y-axis extends up, and the positive Z-axis extends out of the screen (see Figure 1).

The turtle has its own set of three vectors which determine its orientation. Its *forward vector* points in the direction it will travel, its *right vector* points out to its right (like your right arm when fully extended), and its *up vector* points out through the top of its head. See Figure 2.

The turtle can rotate about any of its three vectors. A rotation around the up vector is called a yaw, which is equivalent to the orientation of a 2D turtle. A positive yaw causes the turtle to rotate to the left (see Figure 3).

A turtle can pitch, which is what an airplane does when it is taking off. A negative pitch causes the turtle to rotate around its right arm "backwards" (see Figure 4).

A turtle can roll, which is like tilting. A positive roll causes the turtle to rotate around its forward vector "to the right" (see Figure 5).

Note that all of these rotations are relative to the turtle's current position. In other words, a negative pitch causes the turtle to "take off" like an airplane only if it starts in a position with its forward vector parallel to the ground.

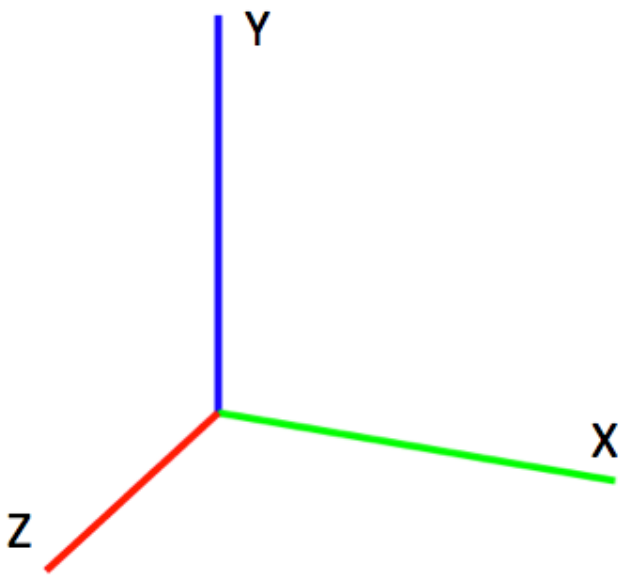

Figure 1: Positive portions of X-, Y-, and Z-axes

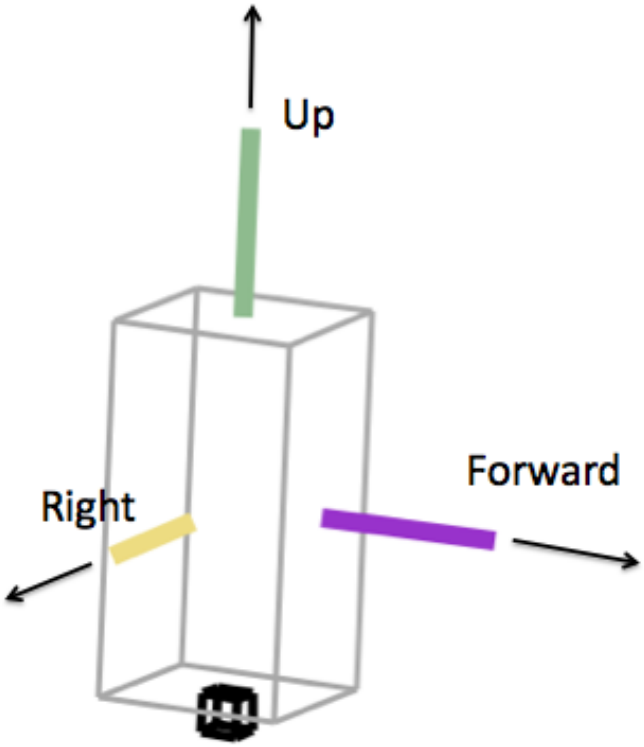

Figure 2: Stephanie's picture of a 3D turtle. The black box at the bottom is the pen. The purple "arm" represents the forward vector. The turtle moves in the direct of the purple arm. The green arm represents the up vector and the light goldenrod arm represents the right vector.

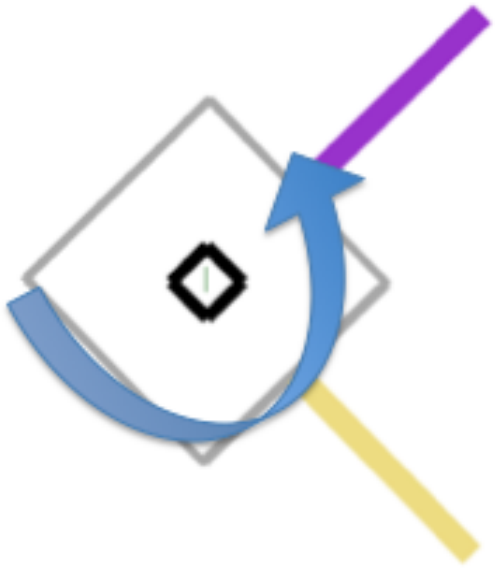

Figure 3: A view from above. A positive yaw (or orientation) is a left turn. I have drawn an arrow to show the left turn as a rotation around the up arm (which is barely visible because we are looking down on it).

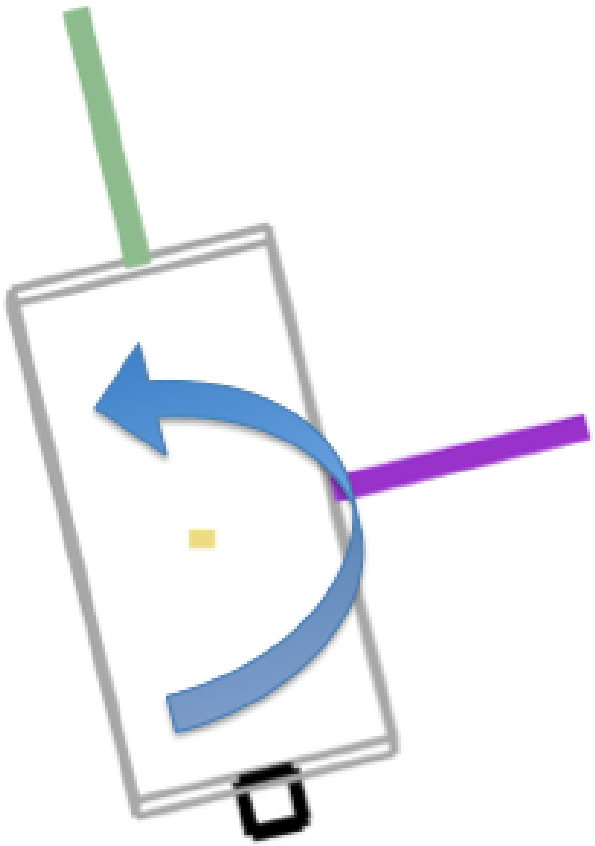

Figure 4: A view from the right-hand side. A negative pitch causes the turtle to “fall back”. I have drawn an arrow to show the fall as a rotation around the right arm (which is barely visible because we are looking straight at it).

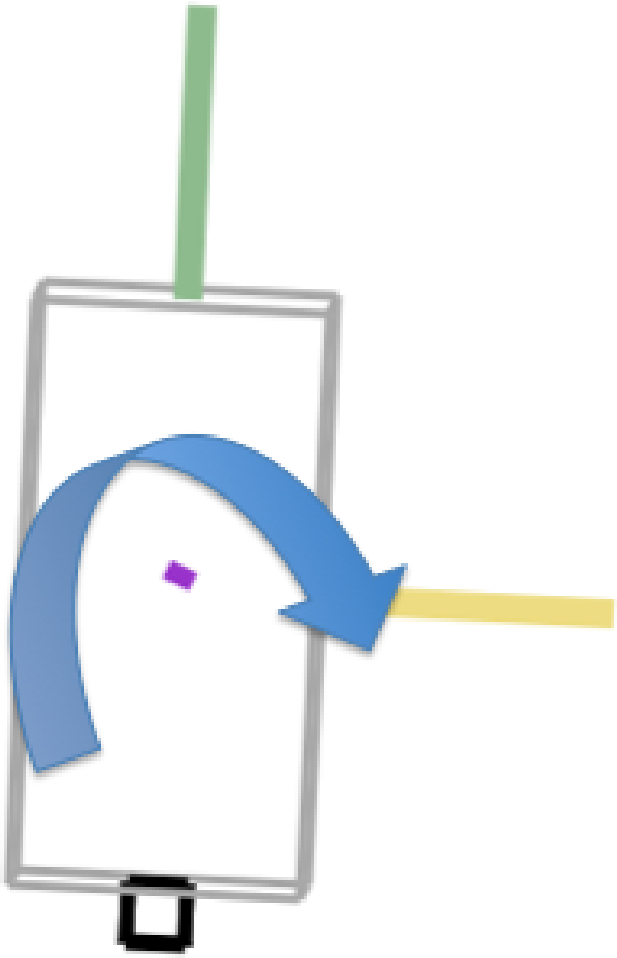

Figure 5: A view from behind. A positive roll causes the turtle to tilt to the right. I have drawn an arrow to show the right tilt as a rotation around the forward arm (which is barely visible because we are looking at it almost directly from behind).

When the turtle is first created, it is oriented so that its forward vector is aligned with the positive X-axis, its up vector is aligned with the positive Z-axis, and its right arm is aligned with the negative Y-axis. To draw the axes, I create a Line object (which simply draws the string 'F'). The draw method resets the turtle to its initial orientation, then uses the orientation argument to yaw it, then the roll argument to roll it, and finally the pitch argument to pitch it.

The X- and Y- axes don't require any change in pitch or roll, because the turtle is already set up to draw in the Z=0 plane. To draw the positive X-axis, I simply move forward. To draw the Y-axis, I turn left (positive orientation) and move forward. To draw the positive Z-axis, I need to pitch the turtle backwards so that its forward vector is heading out of the screen. Then move it forward.

Here is the code that draws the axes (I used it draw Figure 1):

```
# Draw the axes (y is blue, x is green, z is red)
# x
ln = Line(distance = 300, color=(0,0,1))
ln.setWidth(5)
ln.draw( 0, 0, zpos=0, roll=0, pitch=0, orientation=90)

# y
ln.setColor((0,1,0))
ln.setWidth(5)
ln.draw( 0, 0, zpos=0, roll=0, pitch=0, orientation=0)

# z
ln.setColor((1,0,0))
ln.setWidth(5)
ln.draw( 0, 0, zpos=0, roll=0, pitch=-90, orientation=0)
```

Now I want to confirm my understanding by lining up a lot of small 2D squares along the axes. I can control the squares' positions rather than their orientations:

```
s = Square( distance = 5, color = 'green' )
for x in range(0,200,10):
    s.draw( xpos=x, ypos=0, zpos=0, roll=0, pitch=0, orientation=0 )
s.setColor( 'blue' )
for y in range(0,200,10):
    s.draw( xpos=0, ypos=y, zpos=0, roll=0, pitch=0, orientation=0 )
s.setColor( 'red' )
for z in range(0,200,10):
    s.draw( xpos=0, ypos=0, zpos=z, roll=0, pitch=0, orientation=0 )
```

Interpreter Symbols

Now that the turtle can move in three dimension, we need to add symbols to the interpreter to allow it to handle roll, pitch, and yaw. We already have yaw – left (+) and right (-) turns. Positive roll is \ (tilt right) and negative roll is

- yaw

  +: positive yaw (turn left)

  −: negative yaw (turn right)

- roll

  \: positive roll (tilt towards right)

  /: negative roll (tilt towards left)

- pitch

  &: positive pitch (tilt forward)

  ^ : negative pitch (tilt backward)

Now we can make a shape that has each axis as a branch. As as long we have symbols to indicate the three colors (g for green, b for blue, r for red), we can do this:

```
class AxesShape(Shape):
    def __init__( self, distance=100, color = (0,0,0) ):
        Shape.__init__(self, distance=distance, angle=90,
                       color = color, istring = '[gF][+bF][^rF]' )

def drawAxes():
    s= AxesShape( distance = 200 )
    s.setWidth( 10 )
    s.draw( xpos=0, ypos=0, zpos=0, roll=0, pitch=0, orientation=0 )
```