

Getting Started with Tello

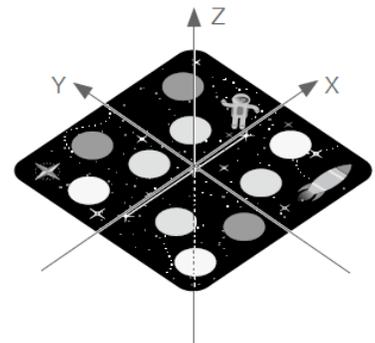
Fly the Cube

0. Tello flies forward, backward, leftward, rightward, upward, & downward. It can fly in **THREE DIMENSIONS** (3D, also known as *space*). Using a special Tello “Mission Pad” assigns Tello a 3D coordinate system so that you can tell Tello to directly `.goto()` a particular point in space by giving the point’s three coordinates as a location.



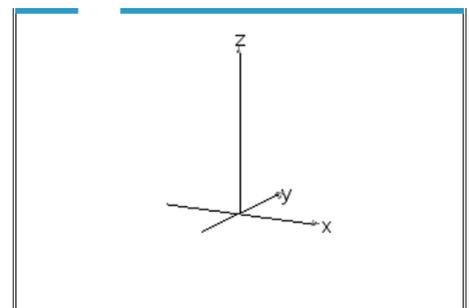
1. The eight Mission Pads look like the one shown here. The pattern on each is slightly different and each Pad has its own 3D coordinate system.

- The *origin* of each coordinate system is the center of the Mission Pad and has the coordinates $(x, y, z) = (0, 0, 0)$.
- The *positive x-axis* is in the direction that the *rocket* is facing on the Mission Pad.
- The *y-axis* is perpendicular to the *x-axis* on the Pad.
- The positive *z-axis* points upward from the center of the Pad. The *z-value* tells Tello how high to fly above (*or below*) the *x-y plane* defined by the Mission Pad.
- The coordinate system’s *unit* is centimeters (cm) in all three directions.



A Mission Pad’s coordinate system is activated when the Pad is **enabled** using the method `tello.enable_mission_pad(padID)`.

2. Yes, Tello *can* fly beyond the edges of the Mission Pad, and, if the Mission Pad is on a table, Tello can even fly *below* the pad by going beyond the edge of the table and then downward. But Tello’s minimum height is still 30cm above the surface below it at any time. The height is continuously measured by the onboard *time-of-flight sensor*. It is similar to, but more accurate than, an ultra-sonic Ranger.



TI-84 PLUS CE PYTHON

- In a new Python program **import** the Tello module and, this time, *test* the `.battery()` to be sure it is safe to fly:

```
from tello import *
if tello.battery( ) > 20:
```

First try some simple maneuvers:

- Take off**
- Enable the Mission Pad you are using**
see `[math] tello_drone...` menu and scroll down
- Go to the point (50, 50, 50)** *Where is that?*
- Go to the point (0, 0, 50)** *Where is that?*
- Land**

To 'enable' the Mission Pad, Tello needs to see it.

Remember that the Tello flying methods are on

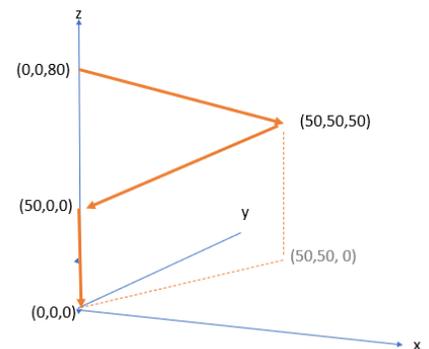
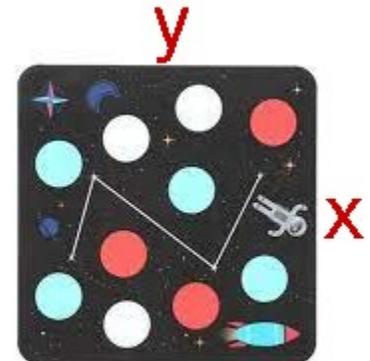
```
[math] tello drone... > Fly
```

Note this nifty trick: if the battery is 'low' (<20%) the program quits right away.

- Place Tello on the Mission pad so that Tello can see the pad (using the camera on the bottom) after `takeoff()`. The direction that Tello is facing does not matter, but let it face in the direction of the rocket on the Pad. Run the program.

The `.enable_mission_pad(n)` method activates the coordinate system for that Pad. If Tello can see several pads at once it will detect the one indicated in the method's argument ($n =$ a number from 1 to 8).

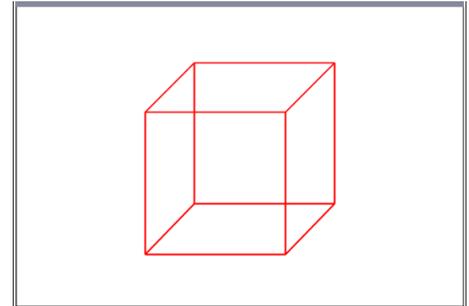
- If your flight was successful: Tello took off, then flew *down* to the left, then back over the mission pad and landed. It went *down* because `takeoff()` causes it to go up 80cm to (0,0,80) and then it had to go *down* a bit to reach the point (50,50,50). It then flew level back over the Mission Pad to (0,0,50) (or close to it) and then **landed**.



TI-84 PLUS CE PYTHON

- Now try flying to the vertices of a cube. Depending on the space you have to work with, make all edges of the cube the same number. We will choose 50cm as the side length.

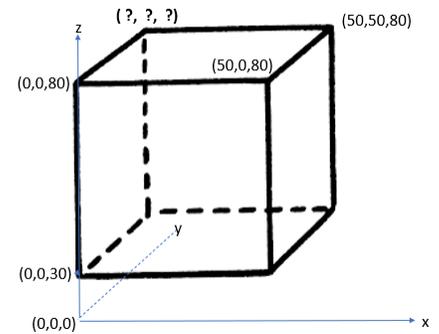
Tip: consider storing the side length as a variable and then use that variable in your flight plan to make it easy to change the size of the flight path.



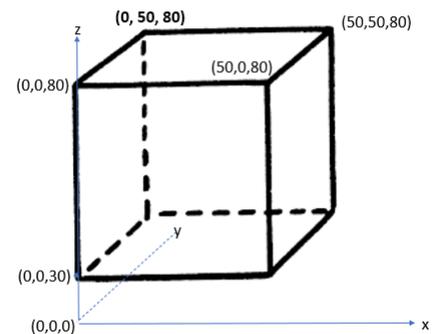
Necker Cube

- Since takeoff takes Tello up 80cm, we'll fly around the top of the cube at this level, then drop down 50cm to fly the bottom edges. After takeoff, Tello is approximately at **(0, 0, 80)**. Two other possible vertices are **(50, 0, 80)** and **(50, 50, 80)**.

What is the fourth coordinate set of the top face of the cube: (?, ?, ?)



- Answer: the fourth vertex of the cube is **(0,50, 80)**.



TI-84 PLUS CE PYTHON

9. Our code so far to complete the top face of the cube:

```
# Fly to the vertices of a cube...
from tello import *
if tello.battery()>20:
    ♦♦ tello.takeoff( ) # now at (0,0,80)
    ♦♦ tello.enable_mission_pad( 1 )
    ♦♦ tello.goto(50, 0, 80)
    ♦♦ tello.goto(50, 50, 80)
    ♦♦ tello.goto(50, 0, 80)
    ♦♦ tello.goto( 0, 0, 80) # return to z-axis
```

Note the change in the two sets of coordinates from your test flight earlier. All four z-coordinates are 80, so Tello stays at the same height for this part of the trip.

10. Now descend 50cm. You can use `.goto(?, ?, ?)` or `.down(?)` to reach the lower level of the cube.

Make another square flight at this level to fly around the bottom face of the cube. *Try it yourself first.*

11. The bottom face of the cube has the same x- and y-coordinates as the top face, so just copy/paste the code and change the z-coordinates from 80's to 30's.

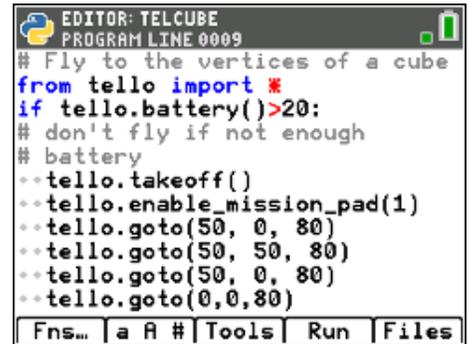
Remember to land .

Make sure the flight path is clear and try the mission now.

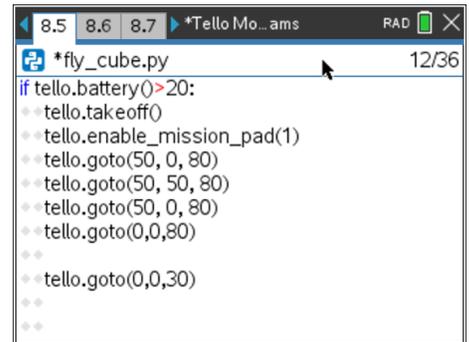
12. Congratulations on 'Flying Most of the Cube'! You've earned your wings.

Challenge: Complete the three missing vertical edges of the cube by flying up and down at each corner of the lower cube.

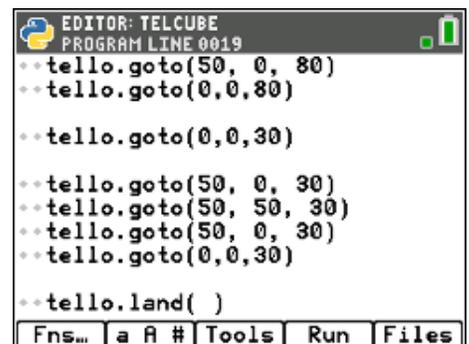
Note: `tello.jump(x, y, z, padID)` is used to transfer Tello to another Mission Pad at its (x,y,z) coordinates. Check out the Mission Pad documentation.



```
EDITOR: TELCUBE
PROGRAM LINE 0009
# Fly to the vertices of a cube
from tello import *
if tello.battery()>20:
# don't fly if not enough
# battery
-- tello.takeoff()
-- tello.enable_mission_pad(1)
-- tello.goto(50, 0, 80)
-- tello.goto(50, 50, 80)
-- tello.goto(50, 0, 80)
-- tello.goto(0,0,80)
```



```
*fly_cube.py 12/36
if tello.battery()>20:
-- tello.takeoff()
-- tello.enable_mission_pad(1)
-- tello.goto(50,0,80)
-- tello.goto(50,50,80)
-- tello.goto(50,0,80)
-- tello.goto(0,0,80)
--
-- tello.goto(0,0,30)
--
--
```



```
EDITOR: TELCUBE
PROGRAM LINE 0019
-- tello.goto(50, 0, 80)
-- tello.goto(0,0,80)

-- tello.goto(0,0,30)

-- tello.goto(50, 0, 30)
-- tello.goto(50, 50, 30)
-- tello.goto(50, 0, 30)
-- tello.goto(0,0,30)

-- tello.land( )
```

