Assignment: A13

Air-Table: Jet forces

New concepts:

- Adding a new force to the physics engine.

Python language topics:

- Sorry, not much new here. Maybe time for you to go back and review...

Problem statement:

(Again, start with a new Python file.)

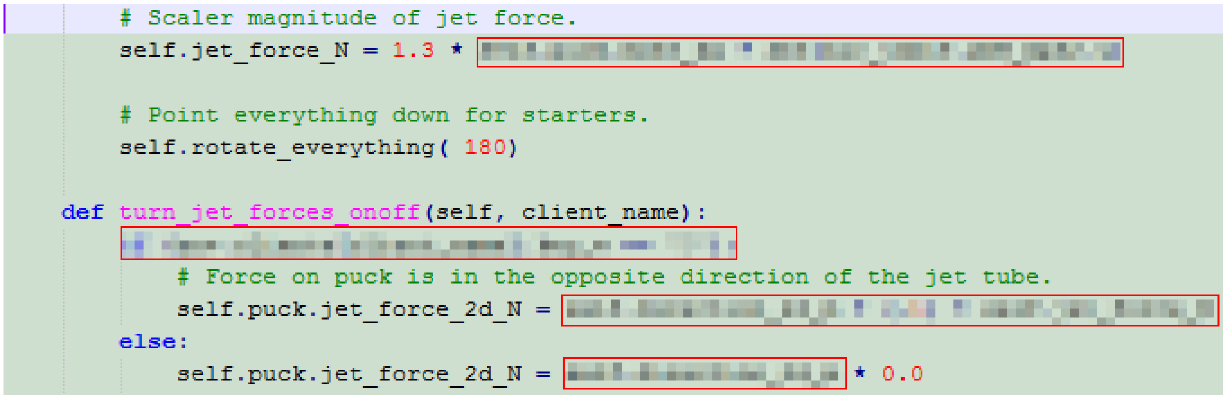
Add content to the A12 exercise to apply a jet force (thrust) to the client-controlled puck. As in A12, control the jet with the “w” key. Have the thrust force be 1.3 times the force that gravity applies to the controlled puck (1.3*m*g). This will give enough thrust to overcome gravity when you have it turned on. Try driving your jet puck with the weakest mouse-tether connected (the middle mouse button). Try driving with and without gravity on.

Algorithmic description:

- Add an attribute to the puck class that represents the jet force on the puck. Initialize this to the (0.0, 0.0) vector.
- Control this force from a method in the jet class. Call this method in the game loop AFTER the user has oriented the jet and BEFORE the call to the physics-engine methods.
    - When the jet is on, the jet force should be 1.3*m*g. The direction of this should be based on the pointing vector that’s associated with the jet.
- Modify the physics calculations to include this jet force.

Python code: (see image below)

The following code (image) is not a complete solution to the problem. It shows additional content relative to the A12 assignment. There is some obfuscation, and you have to figure out what additional code is needed to get this running. The indent levels should be a clue to you.



```
        # Scaler magnitude of jet force.
        self.jet_force_N = 1.3 * [redacted]

        # Point everything down for starters.
        self.rotate_everything( 180)

    def turn_jet_forces_onoff(self, client_name):
        [redacted]
            # Force on puck is in the opposite direction of the jet tube.
            self.puck.jet_force_2d_N = [redacted]
        else:
            self.puck.jet_force_2d_N = [redacted] * 0.0
```