Assignment: A12

Air-Table: Rotating Tubes with Jets

New concepts:

- Cross referencing objects to make a compound object.
- Client-controlled objects.

Python language topics:

- Class inheritance.

Note: Here again are the references links:

http://docs.python.org/2/tutorial/index.html

http://learnpythonthehardway.org/book/

http://www.pygame.org/docs/index.html

Problem statement:

(Again, start with a new Python file.)

Add content to the A11 exercise to build a client-controlled puck. This puck should have a rotating tube and a second tube with a jet flame (triangle) at the end. Each client should control the tubes on its puck with the "a" and "d" for the tube with the jet and the "j" and "l" keys for the raw tube. The jet is displayed when the "w" key is depressed. Test your server with the client file (A10_2D_baseline_client.py).

Algorithmic description:

Create a new Jet class that inherits attributes and methods from the RotatingTube class. Add special attributes and methods to represent the jet's flame. (This should be visible in demo #7 only.)

For each client in the clients list, instantiate a puck, a raw-tube, and a jet-tube. Associate the tubes with the puck by cross referencing their attributes. Add (append) each of these client-controlled pucks to the controlled_pucks list.

In main(), run the client_rotation_control methods of these objects. Then later, after all the physics is done, draw these using their draw method.

Python code: (see images on next few pages)

The following code (images) is not a complete solution to the problem. It shows additional content relative to the A11 assignment. There is some obfuscation and you have to figure out where these pieces of code should go. The indent levels should be a clue to you. **These are in order.** But the neighboring images are not necessarily a continuation from the image above.

```
        self.client_name = None
        self.jet = None
        self.gun = None
        self.rawtube = None
        self.hit = False
```

```
class RotatingTube:
    def __init__(self, puck):
        # Associate the tube with the puck.
        self.puck = puck

        self.color = env.clients[self.puck.client_name].cursor_color
```

```
class Jet( RotatingTube):
    def __init__(self, puck):
        RotatingTube.__init__(self, puck)

        # Degrees of rotation per rendering cycle.
        self.rotation_deg = 4.0

        #self.color = env.clients[self.puck.client_name].cursor_color
        self.color = THECOLORS["yellow"]

        # The jet flame (triangle)
        self.flame_vertices_2d_m =[
                                   ,
                                   ,
                                   ]

        # Point everything down for starters.
        self.      180)

    def client_rotation_control(self, client_name):


    def rotate_everything(self, angle_deg):
        # Rotate the pointer.
        self.direction_2d_m = self.direction_2d_m.rotated( angle_deg)

        # Rotate the tube.
        self.tube_vertices_2d_m = self.rotate_vertices( self.tube_vertices_2d_m, angle_deg)

        # Rotate the flame.


    def draw(self):
        # Draw the jet tube.
        pygame.draw.polygon(game_window.surface, self.color,
                            self.convert_from_world_to_screen(self.tube_vertices_2d_m, self.puck.pos_2d_m), 3)

        # Draw the red flame.
        if (env.clients[self.puck.client_name].key_w == "D"):
```

```
    # Make user/client controllable pucks
    # for all the clients.
    y_puck_position_m = 1.0
    for client_name in env.clients:
        tempPuck = Puck(Vec2D(6.0, y_puck_position_m), 0.45, 0.3)
        tempPuck.client_name = [redacted]

        # Let the puck reference the jet and the jet reference the puck.
        tempPuck.jet = Jet( tempPuck)
        # Same with the raw tube.
        [redacted]

        air_table.pucks.append( tempPuck)
        [redacted]

        # Draw the next one a little higher.
        [redacted]
```

```
            # Delete all the objects on the table. Cleaning out these list reference to these objects effectively
            # deletes the objects. Notice the contolled list must be cleared also.
            air_table.pucks = []
            air_table.[redacted]
            air_table.springs = []
```

```
            for controlled_puck in air_table.controlled_pucks:
                # Rotate based on keyboard of the controlling client.
                controlled_puck.jet.client_rotation_control( controlled_puck.client_name)
                [redacted]
```

```
            for eachpuck in air_table.pucks:
                eachpuck.draw()
                if (eachpuck.jet != None) or (eachpuck.rawtube != None):
                    if ((env.clients[eachpuck.client_name].Qcount < qCount_limit) or (eachpuck.client_name == 'loc
                        [redacted]
```