Notes to Transcribe to Journal:

Using frames and composite objects, I was able to create the wind turbine. There are three levels of frames: the world frame, the blade frames, and the winglet frames. Each winglet and each blade has its own frame, and they are all oriented such that they rotate about the center of the "rotor."

Using the while loop at the bottom of the program, the three composite objects (the blade and winglet combinations) are rotated. What I am attempting to do is show the "wake" of each of the turbine blades as they move through space. This will demonstrate the axis about which the vortices are forming and how they are pushed backward by the wind. In the future I would like to have both the rotational speed of the blades and the rate at which the wake moves be controlled by wind speed, but for now they are both controlled by the arbitrary "number" and "cycle" variables respectively.

As for the generation of the wake, I am having a bit of trouble. My logic is as follows: I initialize a curve in the world frame with only one point, which is the position of the respective winglet converted to its coordinates in the world frame. Then, on each cycle, after each rotation, the new position of the winglet is converted to the coordinates in the world frame and stored in the variable "new_pos#," with the number corresponding to the desired winglet. After this, the new position is appended to the desired wake curve, with the z coordinate translated backwards by the amount of "cycle," to make sure that the wake moves "backwards" in space.

The code for this section is as follows:

```
while True:
      rate(100)
      cycle += .001
      blade_fr1.rotate(angle=-pi/number, axis=(0,0,1), origin=(0,1,0))
      blade_fr2.rotate(angle=-pi/number, axis=(0,0,1), origin=(0,1,0))
      blade_fr3.rotate(angle=-pi/number, axis=(0,0,1), origin=(0,1,0))
      new_pos1=winglet_fr1.frame_to_world(winglet1.pos)
      new_pos2=winglet_fr2.frame_to_world(winglet2.pos)
      new_pos3=winglet_fr3.frame_to_world(winglet3.pos)
      wake1.append(pos=(new_pos1.x, new_pos1.y, new_pos1.z-cycle))
      wake2.append(pos=(new_pos2.x, new_pos2.y, new_pos2.z-cycle))
      wake3.append(pos=(new_pos3.x, new_pos3.y, new_pos3.z-cycle))
```

But what this produces is a single straight line that emanates from the center of one of the blades. I am currently reading the documentation for Vpython and attempting to figure out what is wrong with my code.

I have found the "make_trail" parameter in the documentation, which might serve my purpose, at least in the plane of rotation. There are several problems with this, though. One problem is that the winglet doesn't actually move relative to its own frame, which I think is what is preventing any trail from appearing. The second problem is there is no way to make this move in the z-direction. An idea that I have it to make the whole wind turbine move forward in space, which would leave behind a trail in the z-direction, so I will look into doing this.

I took off the trail attribute, but instead had the program print the values of new_pos1, new_pos2, and new_pos3 with each run through the cycle, which revealed the problem to me. For some reason, it seems that the position for each of the variables is not changing, and ALSO for some reason, it says that the positions of each of the blades is the same. I am thinking that this is because it is taking the position relative to one frame up as opposed to the world frame, which means it is the position relative to the blade that it is attached to, which is (hopefully) the same for all of the winglets. So, we go back to the drawing board on this one.

What I am going to attempt to do to solve this is to use frame_to_world() again on the value of new_pos1, new_pos2 and new_pos3, but this time with the blade frame, and see if this produces the desired result.

When I added the new layer of frame_to_world, it produced something very close to what I wanted. The rotation was a bit fast and the increment too low, so the wake was not clearly defined. Additionally, there was a very weird design at the very beginning of the wake, so I am going to do three things. 1, I am going to slow down the rotation speed. 2, I am going to increase the shift in z for each of the appended points, and 3, I am going to apply the blade_fr#.frame_to_world() method when I declare the initial point of the wake, and see if that works out.

This made the wake VERY close to what I want it to be, unfortunately, every appended point is forming at the center of the the blade, while only the initial points form at the location of the winglet, so I am going to mess around with the appended points.

EDIT – I see my error – I applied the "blade_fr#.frame_to_world" to the winglet position, as opposed to the new value of new_pos#, which put the points at the center of the wings. I am going to change this and see if it corrects my issues.

That seemed to do the trick. Unfortunately, this made me realize a new issue. The additional appended points are formed at the current x and y positions, while the old points are static. This means that the wake is not correctly following the position of the blades. What I need to do is find a way for new points to be generated in the plane of the turbine rotor, while old points are pushed backward. To do this, I think that I might have to translate the entire rotor forward, and have the old points remain static, so that the relative motion is maintained, but the new points are generated at the tips of the blades instead of very far in the wake.

The code of the loop (for record keeping purposes) currently stands as follows:

```
while True:
      rate(100)
      cycle += .01
      blade_fr1.rotate(angle=-pi/number, axis=(0,0,1), origin=(0,1,0))
      blade_fr2.rotate(angle=-pi/number, axis=(0,0,1), origin=(0,1,0))
      blade_fr3.rotate(angle=-pi/number, axis=(0,0,1), origin=(0,1,0))
      new_pos1=winglet_fr1.frame_to_world(winglet1.pos)
      new_pos2=winglet_fr2.frame_to_world(winglet2.pos)
      new_pos3=winglet_fr3.frame_to_world(winglet3.pos)
      new_pos1=blade_fr1.frame_to_world(new_pos1)
      new_pos2=blade_fr2.frame_to_world(new_pos2)
      new_pos3=blade_fr3.frame_to_world(new_pos3)
      wake1.append(pos=(new_pos1.x, new_pos1.y, new_pos1.z-cycle))
      wake2.append(pos=(new_pos2.x, new_pos2.y, new_pos2.z-cycle))
      wake3.append(pos=(new_pos3.x, new_pos3.y, new_pos3.z-cycle))
```

My attempted fix of this was to add "world_fr.z+=cycle" at the end of the while loop. Unfortunately, the wake continues to be created in the same plane as the original turbine rotor as the turbine system slowly drifts forward. This is obviously not the desired result, and so I will attempt to fix it by adding the same z increment to each of the blade frames. Whether or not this works, we will see.

This got us closer to the desired result, but now, for some reason, the rotor is moving forward at a faster rate than the turbine tower, and the wake is continuously generated in the plane of the tower. Additionally, the z spacing of the triple helix formed by the wake continuously increases for some reason, so this is obviously the incorrect fix. I believe the reason for this now is that the z increment is compounding, as the blade frames are moving forward in the z direction relative to the world frame, in addition to the increment that is applied to the world frame.

The issue of the expanded spacing of the helix was because I was incrementing the z direction by "cycle," which was increasing with each pass through the turn. To solve this, I completely did away with the "cycle" variable and replaced it with an "increment" variable, which is located outside of the while loop. This value is constant, and thus the spacing of the triple helix is also constant.

To correct the issue of the rotor flying away from the turbine tower, I made the z position of each of the blade frames increase by "increment" with each cycle through the loop, but I also made the z position of the turbine tower and the rotor base increase by "increment" with each cycle through the loop. This has caused the entire turbine to move forward in z as time passes. The Triple helix now originates at the position of the winglet as well, and everything functions as desired. The scene center also increases by "increment" to ensure that the turbine is always at the center of the screen, and the number of points on each wake curve is limited to the most recent 1000 points to ensure that the wake is not unnecessarily long.

After adding this, I added a series of if statements to the while loop to handle several key presses. At the present time, different variables determine the rotational speed of the wind turbine and the rate at which the wake "moves" behind the wind turbine, which is effectively the wind speed. To account for this, there are two pairs of buttons to govern the increase and decrease of each of those variables. "A" and "d" increase and decrease the wind speed, while "w" and "s" increase the "number" variable, which decreases and increases the rotational speed respectively.

The next step in this process is making the rotational speed and the wake lengthening depend on the same variable, which in this case would be analogous to wind speed. Eventually I will determine what the precise equation of proportionality is, but for now I simply need a placeholder variable that governs both. This means that as the wake lengthening increases, the speed of the rotors increases and vice versa.

To solve for this, I simply made the rotation angle equal to 4*increment. The factor of four is to prevent the length of the wake from far outpacing the angular speed. Additionally, I added if statements to the button handler such that if increment is greater than 2*pi, then a doesn't do anything, and if increment is less than zero, d doesn't do anything. This is to prevent any reverse rotation.

FUTURE PLANS:

-Have RTICA looking at the winglet head-on, with a vector field that demonstrates the vorticity for the current wind speed. Have a variable wind speed that shows changes in the magnitude of the vorticity.

-Have RTICA of a “wing” of a given air foil, and shows the profile/parasitic drag, the lift-induced drag, and the lift, all as vectors or arrows pointing up or out from the wing. Allow the user to change the orientation of the wing and the wind speed to show how the magnitude of the arrows change.

AIRFOIL WITH FORCES RTICA

The current plan for modeling the wing is to create two curves on either side of the wing. These curves will follow the shape of the airfoil. I will then connect each of the corresponding points on the airfoils with a line, and this will give the appearance of a hollow shell.