NOTE:

In my solution I reduced the speed from the lift formula by a factor of 10 such that 120 would be 12, 115 would be 11.5 and so on. When talking about how speed was calculated it should be noted that following the calculation listed I would also divide by 10 to scale the speed, this has been left out of the report for simplicity.

Additionally information from part 1 was used throughout the entire part, that being how the current pitch was calculated. This is not noted in later parts.

Part 1:

What I did: For this I had the object calculate the amount of lift it was currently generating, and the amount of lift it needed for its current weight. It then calculated how much excess lift it had and would adjust its pitch by a function of sin.

It would then check its Y-axis location, in the instance that it was above or below it would calculate the speed needed to have it's pitch adjusted to be the opposite of its current pitch and slow to the needed speed for its pitch to be reversed at the current speed and weight. The pitch however is not adjusted until the next set of movement after the speed is lowered again.

The speed was calculated by taking the currentLift, that is the lift needed to fly straight, and then adding or subtracting (depending on whether the object was above or below its desired altitude)k the lift angle, that is the lift that is excess from the generated Lift minus the currentLift. This was then divided by the slope to give the speed needed to adjust to pitch to the desired angle.

How it worked: This actually worked excellently. The object flew a bit to straight and made me think that I had something wrong even though I didn't.

Pros and Cons: The pros of this system is that it's effective and keeps the object level. The cons are that it seems to have a tendency to drift downward slightly as it is going due to how and when the pitch is changed as a result of the speed.

What I would improve: I would improve it so that it doesn't have the slight downward drift.

Part 2:

What I did: For this part I used the angle that was calculated for what the current pitch of the object needed to be. If this angle wasn't 20 (or any entered angle if I desired to test another case) it would then convert the chosen angle (20 in this case) to radians and add it to the current lift needed to maintain a level flight path. This was then divided by the slope used to calculate the lift. The result of this was the speed of the object needed to have a pitch of 20 degrees. This was solved in a similar manner as the above as they are different sides of a similar coin.

How it worked: This worked out excellently. Similar to the last it worked almost too well that I thought it would be incorrect. The object flew at an angle of around 20 degrees upward.

Pros and Cons: The pros of this is that it works quite well and is able to be used with angles other than 20. A con would be that it has no code in place to stop it from climbing forever.

What I would improve: I would want to add this in along with the above to an alternate form of the generic movement so that its speed would change depending on what it wanted to do.

Note: I know that according to the task specs the pterrordactyl is suppose to do the circle around the origin. Unfortunately due to how the program is setup getting this to happen would require a lot of guesswork so it does not rotate around the origin.

Part 3:

What I did: For this I created a roll variable that would hold what the current roll of the object was. The angle that the object turned each step was a function of sin based on its rotation, the pitch was a function of cosine. I took the value of the pitch and multiplied it by the lift angle and adjusted the pitch by the next lift angle.

To keep level I used a similar formula from before. However this time to keep level the current lift was multiplied by 1/lift where in this case the lift was how much lift we were getting as a result of the current roll.

Currently the object would remain level, but as it traveled its circle would get tighter as it slowed down but retained the roll so it got tighter. To account for this as it slowed down it reduced its roll by a factor of the oldSpeed divided by the new speed.

How it worked: This worked out quite well as the circle would remain almost constant.

Pros and Cons: pros it worked well and had little change in elevation. The cons are that it doesn't ease into turns

What I would improve: I would add the ability for the object to ease into and out of turns if it were to go into the turns straight and then turn.

Part 4:

What I did: for this I used a method similar to the above as well as part 2 combining their work. So the amount that the speed needed to be increased by had the speeded to increase its altitude added to it's needed lift then scaled by 1/lift so that it would make up for the roll decreasing the amount of lift going into lifting the object.

How it worked: this worked out quite well. It went up smoothly as well as keeping a nice circle.

Pros and Cons: pros were that it functioned well. But cons would be that it is coded just for the test and not practical application in just having an object fly around.

What I would improve: I'd Want to make it so that the object could use this while just flying around.