Project List

Instructions: Work together in groups of 4 (with one group of 5). Each group may choose their own project, but you all must choose different projects. You job is to work together to complete that project and prepare a presentation for the rest of the class. Your presentation should cover what your project was, relevant physics, how you solved the problem, and your solutions to the problem (i.e. the animation). You may use any techniques you like to complete the projects. You are encouraged to figure things out as much as you can as a group, but if you are hopelessly stuck, please ask the instructor or TA for guidance. Should you finish your first project early, you may pick one of the other projects to work on, or you may design your own project with the instructor's approval.

Project 1: **Enhanced Projectile Motion** Suppose you are a pirate. Your pirate ship is near an island that has a castle on it. The castle has outside walls of height h, which create a square around the interior of the castle. The interior of the castle has a fragile vault. You want to fire cannon balls into the interior of the castle from your pirate ship, with the goal of hitting the fragile vault (so you can break it open and steal the gold inside). In VPython, write a program that does the following:

- Creates a visual of a ship near an island
- Fires a projectile from the ship to an arbitrary location based on the map, ignoring air resistance. You should have inputs for initial velocity and launch angle theta.
- Make a trail that shows the trajectory of the cannon ball over time
- If a cannon ball hits an outside wall, the cannon ball should stop and the simulation should end
- If a cannon ball hits the interior of the castle, but not the vault, the ball should hit the ground and the simulation should end.
- If a cannon ball hits the vault, the vault should disappear and reveal a pot of gold inside.

Once you have a program that does the above, you should factor in the following:

- Include air resistance in your simulation/animation.
- If you haven't already, assume the cannon is fired at some height h_{cannon} above the water

- Launch multiple cannon balls without starting the simulation over (i.e. in one simulation, you fire 2, 3, 4, etc. cannon balls)
- If you get the above working, implement a 'health bar' for the outside walls and vault. In other words, make it so both of these walls can break (disappear) after they are hit by a certain number of cannon balls

Project 2: **Harmonic Motion**. Write a program using VPython that simulates the motion of a simple pendulum. Your simulation should include the following:

- Visualization of a pivot, to which the pendulum is attached
- A "string" where one end attaches to the pivot, and the other attaches to a ball
- An animation that shows the system moving in accordance with the laws of physics. You should have the option to change the length of the string, gravitational acceleration, the initial point the pendulum is released from, and the initial velocity of the pendulum. You may use the small angle approximation
- A timer (including units) that shows the amount of time elapsed in your simulation. Use this timer to estimate the period of the pendulum, and compare it with its theoretical value. How close are they?

Once the above is done, you may add the following:

- Submerge the pendulum in oil, such that the motion of the pendulum is damped. Recalculate the physics/animation taking this damping into account.
- Do not use the small angle approximation. Alter your simulation such that this is taken into account
- Put the pendulum using the small angle approximation and the pendulum not using the small angle approximation on the same canvas. Give the pendulum a large initial angle, and run the animation. You should notice that the motion is noticeably different. Using your timer, what is the period of the pendulum that does not use the small angle approximation, and how does it compare to the period of the pendulum that does use the small angle approximation?

Project 3: **Planetary Motion** Using VPython, write an animation that simulates the motion of Earth moving around the Sun. Your simulation should include the following:

- A visualization of the Sun and Earth

- An option to set the initial position and velocity of the Earth as it orbits the Sun
- An option to set the mass of the Sun
- An animation that shows the Earth orbiting the Sun in accordance with the laws of physics
- A trail that traces out the Earth's orbit
- A timer that shows the amount of time elapsed (include units)

Once you finish the above, you may add the following:

- Background stars for decoration
- Decorated planets (i.e. and Earth that looks more like Earth, a Sun that looks more like a Sun)
- Comets that streak across the canvas in the distance (you do not need to program physics for them)
- A moon that orbits Earth in accordance with the laws of physics [hard]