AST101: Our Place in the Universe

Lab 4: Simulating Orbits

Version for Online Or In-Person

**Submission Instructions:** when you are done with this lab, send it to [suast101labs@gmail.com](mailto:suast101labs@gmail.com) . Since this week you will be doing your lab on your own, you should send your own submission. You may send us either a shared link or an attachment. Your message should have “Lab 4” in the title.

**What is your name?**

|  |
| --- |
|  |

**Overview:**

You’ve seen me use several computer simulations in class to model planetary orbits and illustrate Kepler’s laws of orbital motion. These programs don’t contain Kepler’s laws -- ellipses don’t appear anywhere in the code, and there are no instructions that say “The planet should go faster near the star”. Instead, I have programmed the computer with some fundamental physics knowledge -- gravity and Newton’s law of motion -- and the orbits that it produces are the physically-correct ones.

In the next two labs, you'll be using a short computer program to simulate these orbits yourself. This week, you’ll be getting familiar with it on your own; next week, you will do more detailed investigations with your group. The TA’s will be on Blackboard Collaborate to help you, but you don’t need to come at your scheduled lab time; you can show up whenever you like to get help.

*Everyone, regardless of group, should do this on their own*. Normally we ask for only one submission per lab group, but this week, we’d like everyone to submit the lab separately. This is because everyone needs to be familiar with the orbit simulator. (You may, of course, work through this with your lab group, but everyone should submit it themselves.) Next week, you’ll be meeting with your lab group as normal.

**Materials:** You will need a computer or tablet. Any computer or tablet will work, since this is a web-based simulation. You could use a cellphone, too, but the small screen and lack of keyboard will likely be annoying.   
  
  
**The Orbit Simulator:** This program works in the following way:

* Choose the mass of your planet, as well as the mass of the star it orbits.
* Choose a starting position for the planet, as well as its velocity.
* Repeat the following steps many times:
  + Use Newton's law of gravity to find the size and direction of the gravitational force
  + Use Newton's law of motion to determine how this force changes the objects' motion over a tiny amount of time, and then change the motion
  + Along the way, draw some stuff so you can see how the planet moves

That's all it does.

**Describing Orbits:**

You will need to remember some terms we learned in class:

* Aphelion: The distance from a planet to its star at the furthest point in its orbit
* Perihelion: The distance from a planet to its star at the closest point in its orbit
* Period: How long it takes the planet to go around its star
* Eccentricity: A measure of how ``stretched-out'' an ellipse is. The minimum value is 0 (this is just a circle); the maximum value is 1.  
    
  Eccentricity can be calculated from perihelion and aphelion distances as   
    
    
    
  Note that the computer will do this for you; you won’t need to do any mathematics yourself.

**Part 1: Running the Code**

On the course webpage, follow the link at the top to the orbit simulator. You should see a window with computer code on the left and a blank pane on the right where the output will show up.

This window isn't big enough, so click on the “hamburger menu” (three lines) in the top left and choose “Fullscreen”.

Now, click “Run”. What happens?

|  |
| --- |
|  |

This orbit has the same aphelion as Earth's. How do the other properties (its perihelion, its eccentricity, and its period) compare to Earth's? You can either compare them numerically (by looking up those values on Wikipedia) or just look at the orbit and compare it to what you know about ours.

|  |
| --- |
|  |

Part 2: Working with Trinket

This computer program is written in the language Python[[1]](#footnote-2), and is embedded in a tool called Trinket that allows it to run in a web browser. You can modify anything you want and see what it does! Don't worry; you won't break anything. You can just choose “Reset” from the menu in the top left to reset it.

Most of the code involves things like drawing on the screen and calculating period, aphelion, perihelion, and eccentricity. You don't need to worry about that part. The things you'll need to change the planet's orbit are right at the top, outlined in a green rectangle made out of # marks. The computer ignores any code on a line after a #, so I have left “comments” in the code for you with these.

Try changing the line start\_velocity = 5 to start\_velocity = 4. This lets you change how fast the planet is moving at the beginning of the simulation. Run the program again. What happened? (Remember, if you ever break anything and want to go back, just reload the webpage.)

|  |
| --- |
|  |

Change a few other things, one at a time. What did you change, and what happened? (Note: if your planet flies away from the Sun, then the initial velocity is so high that the Sun's gravity isn't strong enough to pull it back in.)

|  |
| --- |
|  |

Why do you think you have to wait until the planet finishes a complete orbit (or more) before the simulation can show you the eccentricity and orbital period?

|  |
| --- |
|  |

**Part 3: The Earth’s Orbit**

Reload the page to go back to the default parameters. Most of these are the actual values for Earth; our aphelion, for instance, is 1.017 AU. (In the orbits simulated here, the planet always starts at either aphelion or perihelion.)

However, the initial velocity is not right.

We know that the Earth's perihelion is 0.98 AU; this gives it an eccentricity of 0.0167. (Since the perihelion and aphelion distances are so close together, our orbit is very nearly circular, and the eccentricity is very small.)

Play around with the “initial velocity” parameter; by trial and error, figure out the initial velocity that gives the correct perihelion distance. (Make small changes to the number and run the program, and find the value that gives you a perihelion of 0.98 AU.)

What initial velocity gives the physically-correct perihelion for Earth’s orbit?

|  |
| --- |
|  |

As you change the initial velocity, you'll notice the shape of the orbit change, as well as the orbital period and the eccentricity. What is the period? Is it what you expect it to be?

|  |
| --- |
|  |

In class, you learned that the force of gravity causes two objects to attract each other with a force

You also learned what forces do: they cause objects to accelerate in the same direction as the force. The acceleration on an object from a force is given by Newton's second law of motion:

Look at lines 53 and 54 of the code. Take a guess what they do. (This computer code should look a lot like one of the equations above!) What do you think \*\* means in Python?

|  |
| --- |
|  |

**Part 4: Units Used in the Program**

This code uses the following units to measure things:

* Distance is measured in AU
* Time is measured in years
* Mass is measured in multiples of the Sun’s mass

However, in the standard SI system, we typically measure distance in meters, time in seconds, and mass in kilograms. Why do AU, years, and solar masses make sense for this program to simulate orbits of planets?

|  |
| --- |
|  |

**Part 5: Changing the mass of Earth**

The mass of Earth is a tiny fraction of the mass of the Sun. Slowly increase the mass of the Earth. Try values of 0.001, 0.01, 0.1, and 0.3 solar masses.

What changes? Why do you think this happened? (You will return to this question next week.)

|  |
| --- |
|  |

**Submission Instructions:** when you are done with this lab, send it (as an email attachment or link to a shared document) to [suast101labs@gmail.com](mailto:suast101labs@gmail.com) . Since this week you will be doing your lab on your own, you should send your own submission.

1. Technically, it is a Python version called “Glowscript” that has some animation tools and is supported by Trinket. This is why you see the top line: GlowScript 2.7 VPython. It's needed to tell Trinket what kind of code this is! [↑](#footnote-ref-2)