Project 1: Orbital Mechanics

Due: Monday May 21, 11:59 pm

1 Assignment Overview

This project focuses on some mathematical manipulation. You will compute the orbital position of the Earth around Sol.

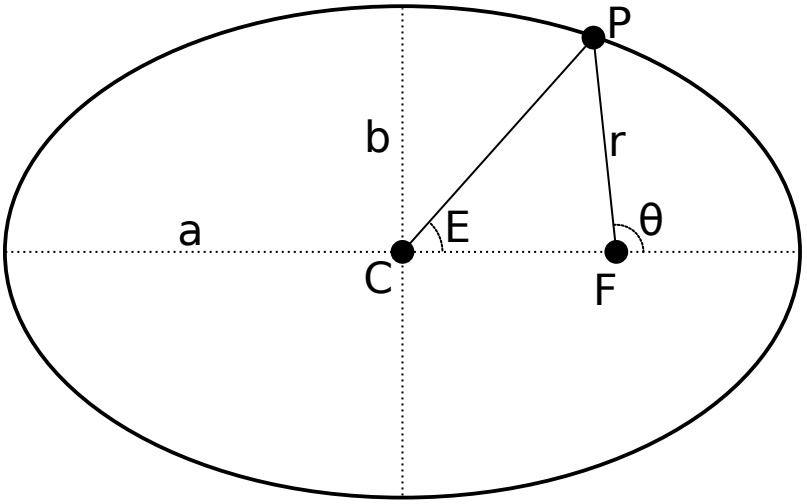
2 Assignment Deliverables

You must turn in completed versions of the following files:

- proj01/orbit.cpp

Be sure to use the specified file name and to submit your files for grading via **Mimir** before the project deadline.

3 Background



The planets orbit around Sol in elliptical orbits. Fortunatly, we have equations that describe ellipses; we will use these equations to compute the distance from Earth to Sol.

Ellipses are defined by two main parameters, the length of the semi-major axis, a, and that of the semi-minor axis b. These values are specific for each particular ellipse. Additionally, a third parameter, the true anomaly, θ, determines the location of a planet at point P on the ellipse. Sol is located at the focus F. The distance r from F to P is given by $r = a(1 - e\cos E)$ where e is the eccentricity and E is the eccentric anomaly.

The eccentricity is given by $e = \sqrt{1 - \frac{b^2}{a^2}}$. The eccentric anomaly is given by the equation

$$E = 2 \arctan(\sqrt{\frac{1-e}{1+e}} \tan(\frac{\theta}{2})).$$

We can also use the eccentric anomaly to convert from polar coordinates to Cartesian coordinates. The (x, y) coordinates about the focus are given by $x = a(\cos E - e)$ and $y = b \sin E$. You can then check your work by using the distance formula: $r^2 = x^2 + y^2$.

4 Assignment Specifications

Your program will take as input three double values, a, b and θ. Note that θ will be given in *degrees*. You will need to print the following lines:

- The eccentricity e in scientific notation with 3 significant figures
- The eccentric anomaly E in *degrees* to 1 decimal places
- The radius r in scientific notation with 3 significant figures
- x and y, separated by a space in scientific notation with 3 significant figures

Note that while your input and outputs are in degrees, the trig functions expect radians.

5 Assignment Notes

- You will receive no points if your solution does not compile on Mimir.
- Points will be deducted if your solution has any compiler warnings.
- The `<cmath>` library contains a number of useful methods including `pow`, `sqrt`, and a variety of trig functions.
- You will need to convert from degrees to radians. You will need to use the `M_PI` constant in `<cmath>`. Remember that $2\pi = 360°$.
- The `fixed` and `scientific` functions in `<iostream>` and the `setprecision` function in `<iomanip>` may be useful for helping to format your output.
- You *do not* need to check for bad input values for this project. In future projects we will explicitly indicate the errors we are looking for.
- It gets irritating to type in 3 numbers each time you test. To get around this, you can use a handy trick on the command line. You create a file with the 3 input values (on a single line they need to be space separated but 3 different lines is also fine). You can *redirect* the file to your main program. If your code is compiled as `a.out` you can do:
  `./a.out < fileOfInput.txt`
  This will automatically feed the input to the `cin` statement and produce the output. An example `earth.txt` is provided in the project directory.
- Earth at perihelion (the closest point when $\theta = 0°$ is 1.47×10^8 km from Sol and at aphelion (the farthest point when $\theta = 180°$) is 1.52×10^8 km from Sol.

6 Getting Started

1. In Mimir, I created a directory `proj01` and placed in it an empty file (no contents) with the correct name, `orbit.cpp`. You can updated the file contents there.

2. Write your code and compile it.

3. Prompt for some of the values and print them back out, just to check yourself.

4. Check that your calculations are right (as indicated above).

5. Upload a copy of your project to Mimir and submit. You don't have to pass all the tests the first time! You can add more information and pass more of the tests as your progress. Do things incrementally.

6. Now you enter a cycle of edit-run to incrementally develop your program.

7. If a version you submit and test on Mimir passes all the tests then you are done. If time runs out and you didn't pass all the tests then however many you passed indicates what grade you got for the project. It doesn't matter now since the project is relatively easy, at the end you do the best you can and pass as many tests as possible. However, thanks to Mimir you *always know* how you are doing.

8. Remember, in the end it only matters that it compiles and runs on Mimir. If it doesn't compile there then it doesn't compile at all (no matter what else you did on whatever environment you used). Mimir is the last word.