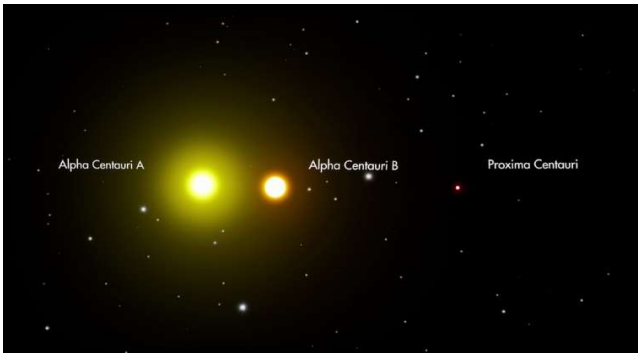
Simulation of Alpha Centauri

By Johnathan Levy and Victor Sparleanu

Introduction

α Centauri is a star system in the southern constellation of Centaurus. It consists of three stars. Two of them are sun-like stars with the third even having its very own confirmed planet orbiting it. We've decided to look at two of the three stars in this constellation, *Rigil Kentaurus* (α Centauri A) and *Toliman* (α Centauri B). In the night sky it looks as if α Centauri AB were one star when they're two. These two stars orbit around each other. That's why it always looks as if they were one. α Centauri is the brightest star in the constellation and the third brightest star in the night sky. The third star *Proxima Centauri* (α Centauri C) is a red dwarf and is therefore not visible to the naked eye. In the year of 1603, J. Bayer named the stars after the half human, half horse creature in Greek mythology. According to the myth, Hercules accidentally hurt the centaur and placed him in the sky after his death to honor him. Alpha Centauri is supposed to resemble the centaur's hoof in the sky. The star's traditional name (*Rigil Kentaurus*) is the Latin translation of the Arabic name "*Rijl al-Qinṭūrus*", which means "the Foot of the Centaur". The stars can only be seen from the south of ~40° N latitude. The reason Alpha Centauri is such a well-known constellation is because it's the closest known star system to our solar system. In our program we wanted to simulate the relation between Alpha Centauri A and B and how they orbit around each other. The two stars move at a velocity of 3686 milliarcseconds/year in 11° in north of west. For that reason, we had to adjust our program, so we would be able to actually see the stars move instead of waiting a whole year to see the stars move 3686 milliarcseconds.



NASA SVS | Alpha Centauri Stellar System

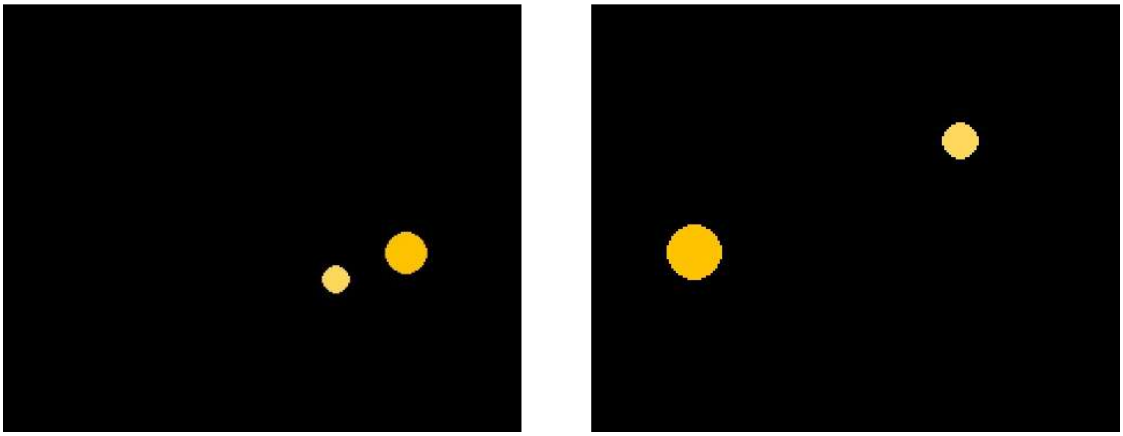
Code Description

The code employs abstraction and encapsulation using separate modules for configuring the parameters of the simulation and for providing classes for self-implemented vectors, celestial bodies, interactions, as well as a data structure to hold, manage and ensure availability across modules of various scaling-related parameters. The simulation is designed to support resizing the screen by expanding / shrinking the current view of the portion of the universe, as well as automatically resizing the viewport to ensure the celestial bodies are kept on-screen at all times. Pygame checks for user input on each tick, enabling the user to pause the simulation (space), quit (q, esc) or make the simulation full screen (f). Furthermore, the user can adjust the speed

of the simulation (FPS) using the arrow keys (up/down). The program also draws a grid to illustrate the dimensions of the celestial movements. This way, it is clear how the stars move, even if the viewport size is changed. It also gives a good overview of how the star system moves as a whole with time.

Tests

At first, we wanted to use Processing, but after some time, we figured out that it was more difficult due to the fact that things like imports, type hints or external libraries weren't implemented and there were other less customizable aspects to it. That's why we decided to use Pygame. In Processing we tried to make "q/esc" the "quit" button and "f" for full screen, but Processing wouldn't work. When we tried to make the window full screen, the resolution had to change, but in Processing it would always jump to other random resolutions. In Pygame on the other hand, it would work effortlessly the second we tried it. In the beginning we used one main file on GitHub to write all our code, but it began to pile up very quickly and was getting difficult to debug, due to the amount of code and disorder, so we split the program into multiple modules containing configurations. We used helper classes and the actual main program logic. That allowed for strict, secure and reliable code execution, which was useful when debugging errors because we would know where the code was wrong.



Photos of program in different stages

Summary

In our program we were able to show how the two stars, Alpha Centauri A and Alpha Centauri B, orbit around each other. The stars are the third brightest in our night sky. Everything in our program is accurate. The scale factor is true to size and so is the orbit itself. The only thing we didn't make accurate was the speed of the orbit, because it would've been too slow, so we made it a little faster with the help of Pygame. We used Pygame due to technical difficulties we stumbled upon while using Processing.