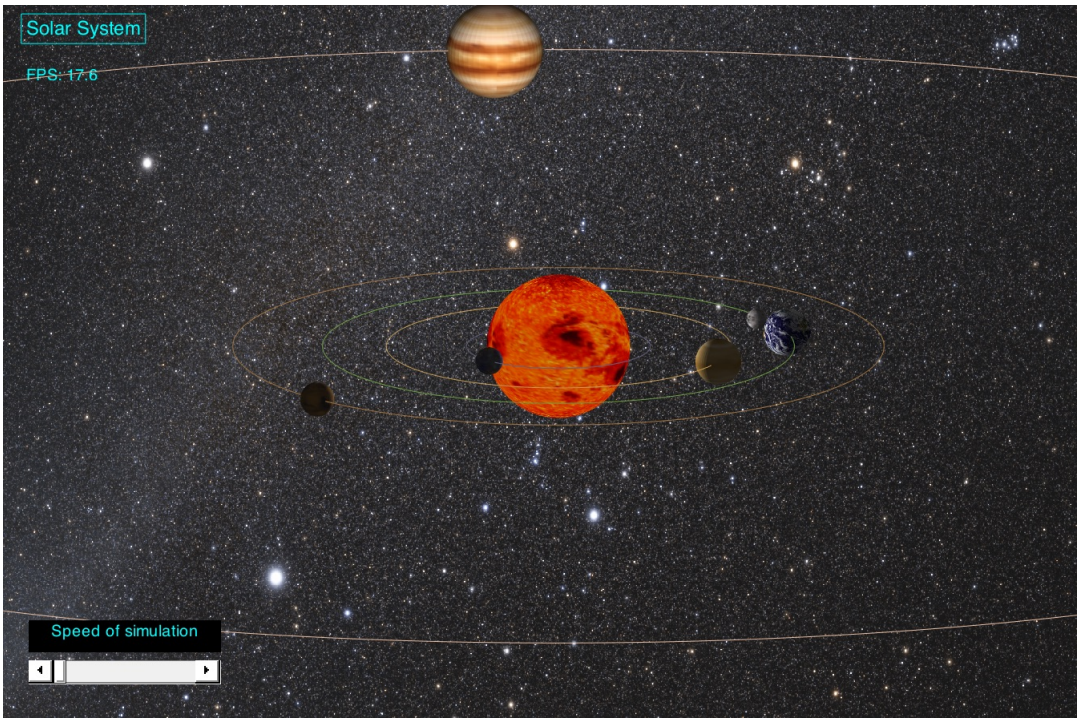



# Solar System Simulation

## Simulation of our solar system

**Applications of Matlab/Octave**

*MatLab* and *Octave* are very versatile programs. To explore the possibilities, we created a simulation of our solar system using *MatLab*. bla bla bla realism, bla bla bla to scale and some shit about not elipsies but circles

Studiengang: Micro- and Medicaltechnology
Autoren: Eric Lochmatter, Lukas Studer
Datum: 06. 01. 2016

# Inhaltsverzeichnis

# 1 Introduction

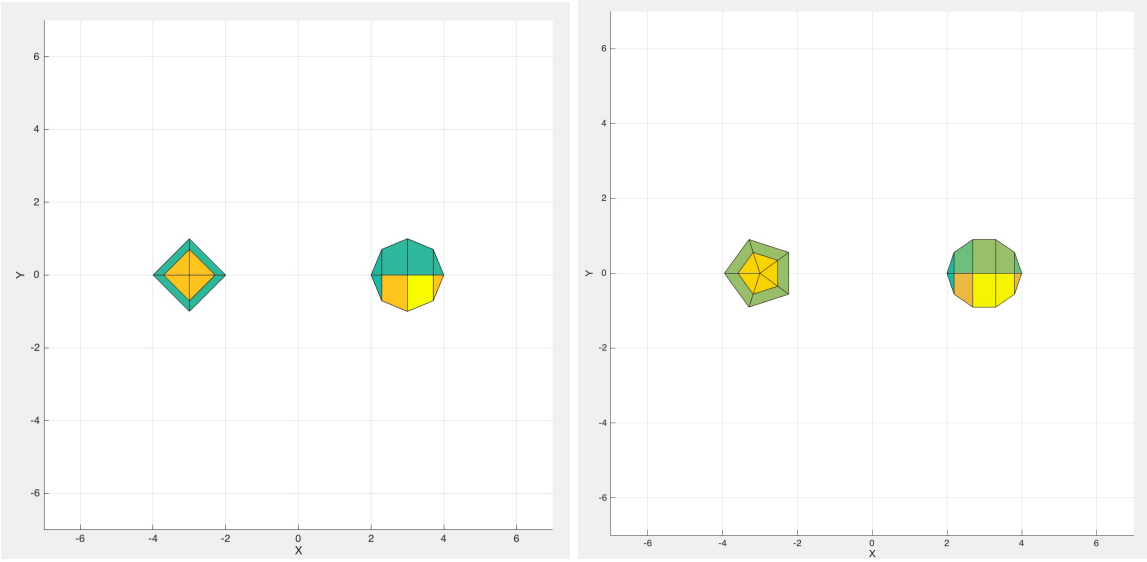
This project aims at creating a simple modle of our solar system using *MatLab*. While putting emphasis on the visuals, we kept the mathematical aspect simple.
Trajectories are not calculated and the orbits are perfect circles, due to the simple fact that the difference would be hardly noticeable. The sizes and distances are mostly to scale. We use a logarithmic scale for the sizes and a linear, though a very small scale, for the distances.

# 2 UI Elements

The modle provides two UI elements for the user, besides the common controls from *MatLab* itself. A display for the number of frames rendered pre second and a slider to control the speed at which the modle runs.

## 2.1 FPS Display

While programing the modle we soon realized that the certain changes had a massive impact on the number of frames *MatLab* could render per second. The number of frames per second, also known as *FPS* directally dictates how smooth the modle runs. Through time analysis of the modle we found out that the resolution of the planets, sun, and moon had the most noticable impact on the *FPS*. We then added the display to find the optimal resolution at which the graphics of the planets didn't suffer to much but still allowed for a smooth animation. We settled for a resolution of 50. This generates spheres composed of 50 equally wide rings with 50 equally big surfaces. This keeps the *FPS* around 20 which is only slightly lower than the number images the human eye can capture per secend. This makes the animation appear smooth. Following images shows how the sphreres react to different resolutions:




*Left: a sphere with resolution 4 (top and side view); Right: a sphere with resolution 5 (top and side view)*

CALCULATION OF FPS

## 2.2 Speed Slider

Because Uranus and Neptune are so far away from the sun, it takes them quite a while to complete an entire turn around the sun. That is why we added a slider on the bottom left which allows users to speed up the animation. Additionally the slider can also stop the animation by setting the speed to zero which allows for closer inspection of the planets. The slider basically controls the angle of rotation per tick. Faster speed results in a grater angle. Setting the speed to higher values leads to visual effects like planets appearing to jump rather than smoothly moving and planets smoothfully to be turning backwards. Alternatives like controling a delay for rendering the next frame have been tested, but didn't work out due to the *FPS* plummeting or the animation quickly reaching a maximum speed and therefore not having the desired effect.

# 3 Rotation

## 3.1 The Problem with Rotation Angles

While we tried to make things as realistic as possible