

Konstanz, 23.09.2025

## Assignment 2

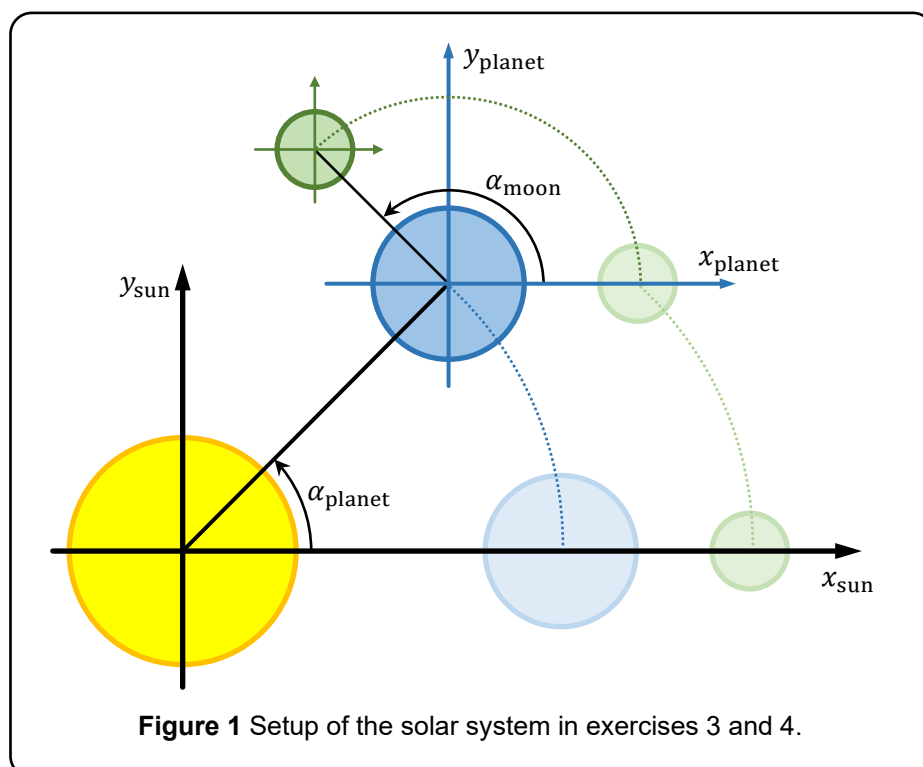
### Computer graphics

**Deadline 26.11.2025.**

#### Programming frame-work:

Download the programming frame-work from the web-page of the lecture. The zip-file contains six files `main.cpp`, `Vector.h`, `Matrix.h`, `Color.h`, `Planet.h`, `Planet.cpp`:

- The file `main.cpp` contains the programming framework, which supports the usage of the timer-callbacks, the so-called double-buffer and several display functions.
- The files `Vector.h` and `Matrix.h` provide implementations of generic vector and matrix classes. Some of their methods are already implemented. If necessary, extend the classes by further methods, following the examples in the existing code.
- The file `Color.h` provide a very simple color class. No changes required here.
- The files `Planet.h` and `Planet.cpp` provide an class representing circular object, e.g. planets or suns.



The functionality of both Exercises will be evaluated by demonstration and source code inspection!

**Exercise 3 (Affine transformations in 2d)****5 points**

Implement a visualization of an animated solar system. The solar system contains a central sun, a planet and a moon as in Figure 1. The planet rotates around the sun, while the moon rotates around the planet. Both rotations happen simultaneously.

You can use for the implementation the classes `CVec2*` and `CMat2*`, i.e. here use exclusively 2d-objects.

**Approach:**

- Adapt your implementation of the Bresenham algorithm from Assignment 1 to work with pixels. To this end, use the OpenGL methods `glBegin(GL_POINTS)`, `glEnd()` in combination with `glVertex2i` to draw an individual pixel. Integrate it to the `Planet::draw()`-method.
- Define three positions for the sun, the planet and the moon. Initially the sun is in the origin and the planet and moon on the positive x-axis. Define two angles and angle increments for the planet and the moon. Draw the three celestial bodies on the screen using the `Planet::draw()`-method.
- Implement a function to rotate an arbitrary point in affine coordinates around the origin. Test your function rotating the planet around the sun.
- Implement a function to rotate an arbitrary point around another arbitrary point in affine coordinates. Test your function rotating the moon around the planet.
- Combine both functions to rotate the planet around the sun and simultaneously rotate the moon around the planet. Realize these functions in the display function `displayExercise3`.

**Exercise 4 (Affine transformations in homogenous coordinates)****5 points**

Implement the transformations from Exercise 3 in homogenous coordinates and combine them in the display function `displayExercise4`.

Here, use only 3x3 matrices and 3d vectors, i.e. homogenous coordinates. In particular, implement the rotation around an arbitrary point as a single matrix!