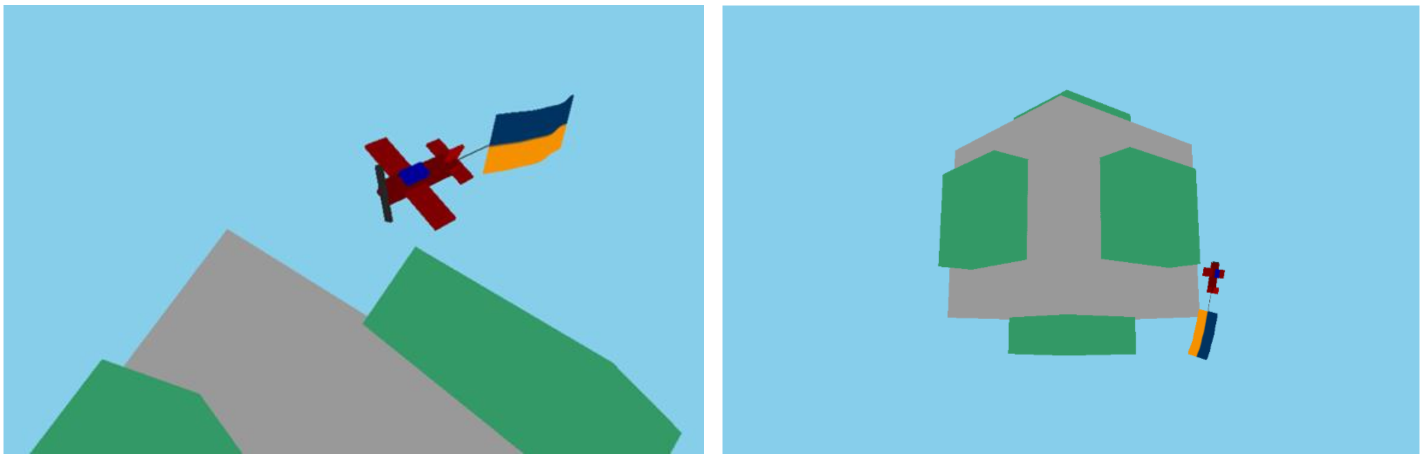
Proseminar Visual Computing
Winter Semester 2024

Assignment 3

Hand-out: November 05, 2023

Hand-in: November 18, 2023



Topics

- General OpenGL programming
- Transformations
- Basic animations and user controls
- Camera control

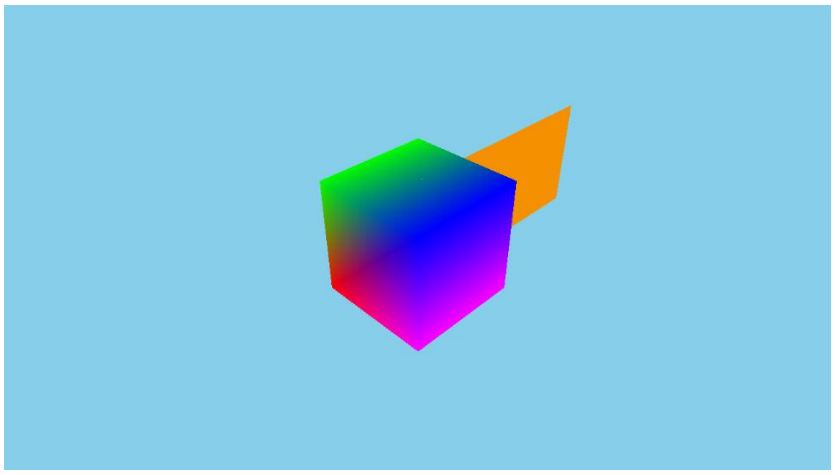
Outline

The goal of the Computer Graphics assignments of the Visual Computing PS is to build a controllable plane flying around a small planet. This work is divided into 3 steps. Each step corresponds to a programming assignment. In the first assignment, we focus on geometric transformations and animation. Its objective is to build a very simple plane made from basic geometric primitives (i.e., cubes) by applying different transformations. The plane should be controllable by user input. Behind the plane, a flag should be dragged along. Further, the flag should be animated to simulate the effect of wind on the cloth. Finally, an additional camera mode should be added. See the example video provided with this assignment for a working solution including all features.

Template code

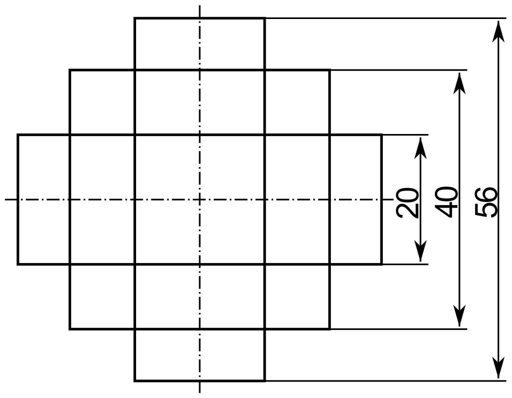
A template code is provided for this assignment. You can modify this code to build your own scene. Implementation can mainly be done in the main source file **assignment_3.cpp** and in the **flag.cpp/h** file, but feel free to create your own files (*e.g.*, it might be helpful to add a **plane.cpp/h** and **planet.cpp/h** file) and implement wherever you want.

The scene available in the template code contains a plane as basis for the flag and one cube that can be rotated around its *x*- and its *y*-axis using the W, S, A, D keys. Moreover, an orbit camera with the cube in its center is available. The camera can be controlled by holding the left mouse button; it follows the mouse movements and rotates on a sphere with a fixed radius. You can zoom in and out (*i.e.*, de- and increase the radius of the sphere) with the mouse wheel. In the figure below, an example of the template scene is depicted:



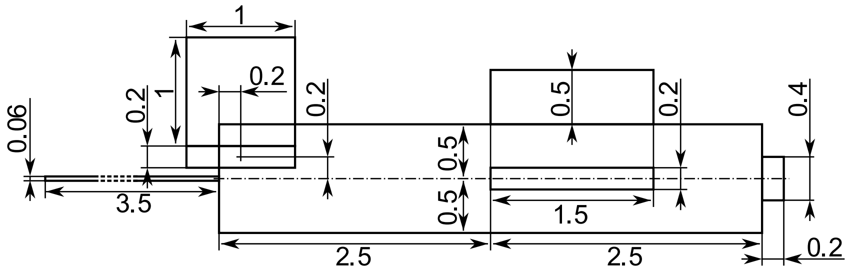
Tasks

1. Scale up the cube from the template code as the base for the **planet**. Add three cubes (one for each axis) for the protrusions (see figure below, same for each axis). As this assignment does not include any shading, change the colors of the cubes so that the difference in geometry is visible. In the end, it should look like the planet in the pictures at the top of this document. You are free to design your own planet, but it must be made up of **at least four cubes**.

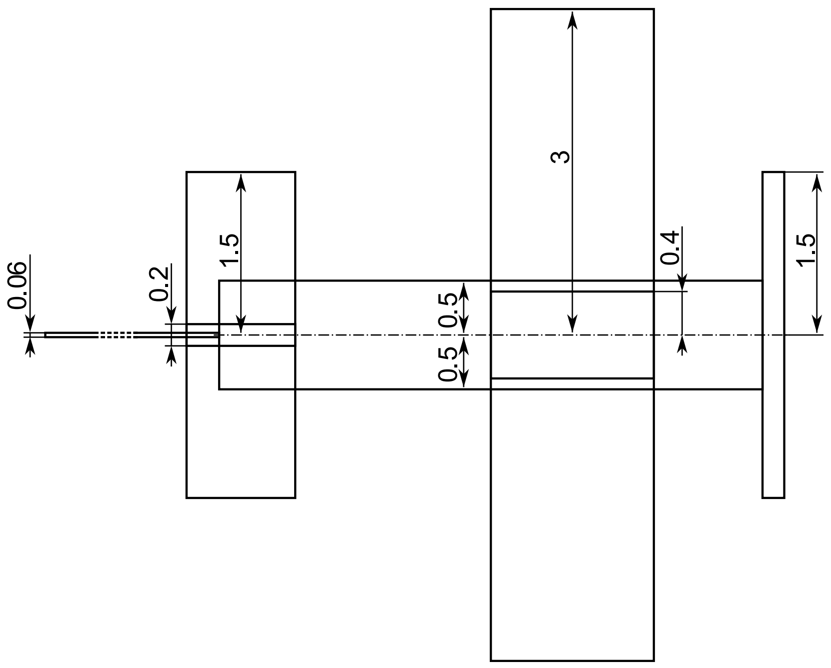


2. Setup the hierarchical geometrical **model of the plane**. For that, remove the scaled cube model from the template code. Then start with **7 cubes** (size: 2x2x2) and apply transformations to them (**scaling, rotation, and translation**) to obtain the **plane parts** representing: **body, cockpit, wing, tail wing, rudder, propeller, and the pole connecting the flag** (see introduction figure). All the necessary measurements for the various parts can be taken from the drawing below. Make sure that the given flag in the template code is connected properly to the back of the plane via the pole. Change the **flag** to contain **at least two different colors**, though the specific design is up to you. You are also free to design your own plane, but it must include **at least the same number of parts**. In addition, at least two of the three transformations **scaling, rotation,** and **translation** must be used for its creation.

side view



top view



Use **different colors for the different parts** (see example images).

3. Now, implement **basic animations** so that it looks like the plane flies around the planet. For this, there are two main approaches:
   - The planet is static, while the plane orbits the planet.
   - The plane is static. To simulate the plane's movement, the planet rotates accordingly.

   While you are free to implement the animation however you prefer, the second approach is slightly easier, as no polar coordinates are needed for the plane's movement.

   The **propeller** must be animated as well.

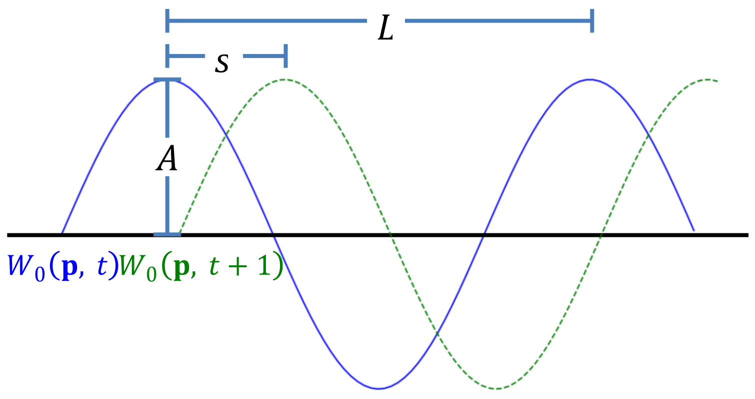
4. Next, **animate the flag** over time. In each frame, you'll need to compute the displacement (offset in x-axis) of each vertex of the flag and update the corresponding OpenGL vertex buffer with the new positions.
   The displacement of the flag (at location $\mathbf{p}$) is modeled with a sum of periodic waves:

   $$H(\mathbf{p}, t) = \sum W_i(\mathbf{p}, t)\,,$$

   each defined with varying amplitude A (wave height), phase φ (speed $s = \frac{\varphi L}{2}$), frequency ω (wavelength $L = \frac{2}{\omega}$), and direction $\mathbf{d}$ **:**

   $$W_i(\mathbf{p}, t) = A \sin(\omega(\mathbf{p} \cdot \mathbf{d}) + t\varphi)\,.$$

   In the provided video we used <u>three</u> wave functions with the following parameters (see *struct FlagSim* in **flag.h**):



$$\begin{array}{llll}
A_0 = 1.0, & \varphi_0 = 1.0, & \omega_0 = 0.25, & \boldsymbol{d_0} = [\mathbf{0}, \mathbf{1}]^{\boldsymbol{T}} \\
A_1 = 0.2, & \varphi_1 = 1.5, & \omega_1 = 0.75, & \boldsymbol{d_0} = [\mathbf{0}, \mathbf{1}]^{\boldsymbol{T}} \\
A_2 = 0.1, & \varphi_0 = 5.0, & \omega_0 = 2.0, & \boldsymbol{d_0} = \left[-\frac{\mathbf{1}}{\sqrt{\mathbf{10}}}, \frac{\mathbf{3}}{\sqrt{\mathbf{10}}}\right]^{\boldsymbol{T}}
\end{array}$$

   Furthermore, add a **linear scaling** to the **displacement**, so that it is at 0% where the flag is connected to the plane's pole and 100% at the very back of the flag.

   <u>**Note:**</u> By OpenGL convention, the flag is placed in the *yz*-plane with the x-axis representing the horizontal offsets.

5. Add **user control**: It should be possible to steer the plane **left**, **right**, **up** and **down**, and to **speed up** and **slow down**. In addition, the plane should be **animated** when changing direction or height.

   - As a first step, it should be possible to control the plane's **left** and **right movements** (*e.g.*, by pressing A and D). This turning action should combine both **yaw** and **roll** of the plane.

   - Steering **up** and **down** (*e.g.*, by pressing SPACE and CTRL) should also be possible. In this case, the **pitch** of the plane should be adapted.

   - Lastly, it should be possible to adjust the **speed** of the plane (e.g., by pressing W and S). For the speed to be more visible, also adapt the flag animations so that the waves are faster with higher speed.

     Hint: Scale the delta time between frames using a speed dependent scaling factor.

   - Make sure that the plane’s movement looks reasonable, and that the minimum height is limited so that the plane does not clip into the planet.

6. Finally, implement a **second camera mode**. There should be one camera mode in which the camera (and its *lookat* position) follows the plane (third person camera) and a second camera mode, in which the planet is fixed (not rotating) and the plane circles around it (planet is camera center). Use two different keys (*e.g.*, 1 and 2) to switch between the two camera modes.

   Hint: Depending on the approach in exercise 3, for one of the two camera modes, the camera system needs to be rotated (possible with function *setCameraRotation(…)*).

Implementation Remarks

Make sure that your code is clear and readable. Write comments if necessary. Your solution should contain a **readme file** with the names of the team members, a list of keyboard controls, and any explanation that you think is necessary for the comprehension of the code.

Submission and Grading

Submission of your solution is due on November 18th, 2024 (23:59). **Submit the sources** (*i.e.*, only the content of the *src* folder) in a ZIP archive via OLAT. Do not submit the executable and the content of the *build* folder. Do not submit the external dependencies either. Both folder and archive should be named according to the following convention:

Folder: **CGA3_<lastname1>_<lastname2>_<lastname3>**

Archive: **CGA3_<lastname1>_<lastname2>_<lastname3>.zip**,

with <lastname1>, etc. the family names of the team members. Development in teams of two or three students is requested. Please respect the academic honor code. In total there are 15 marks achievable in this assignment distributed as follows:

- Geometrical models (**5 marks**)
- Plane control and plane/flag animations (**6 marks**)
- Additional camera mode **(2 marks)**
- Code readability, comments, and proper submission: (**2 marks**)

Resources

- Lecture, proseminar slides, assignment description video, and the template code are available via OLAT.
- OpenGL homepage
  http://www.opengl.org
- OpenGL 3.3 reference pages
  https://www.khronos.org/registry/OpenGL/specs/gl/glspec33.core.pdf
- OpenGL tutorial
  https://learnopengl.com/
  http://www.opengl-tutorial.org
- GL framework GLFW
  https://www.glfw.org/documentation.html

Note: Be mindful of employed OpenGL and GLSL versions!