Proseminar Visual Computing
Winter Semester 2025

Assignment 3
Hand-out: November 4, 2025
Hand-in: November 17, 2025

Topics

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

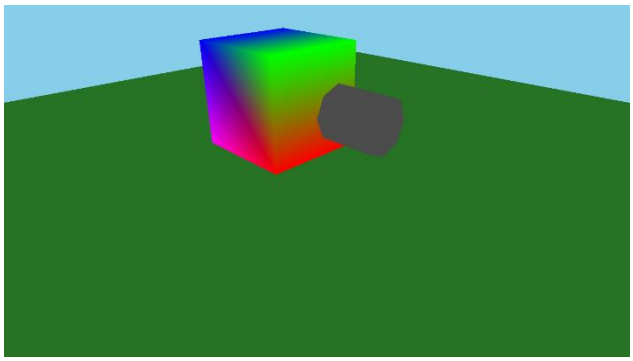
Outline

The goal of the Computer Graphics assignments of the Visual Computing PS is to build an animated pickup. This work is divided into 3 steps. Each step corresponds to a programming assignment. In this first assignment, we focus on geometric transformations and animation. The objective of this first assignment is to build a very simple pickup made from basic geometric primitives (i.e., cubes and cylinders) by applying different transformations. Further, the pickup should be controlled using user input and when driving around, the wheels should be animated. Finally, an additional camera mode should be added that follows the pickup. See the example video provided with this assignment for a working solution including all features.

Template code

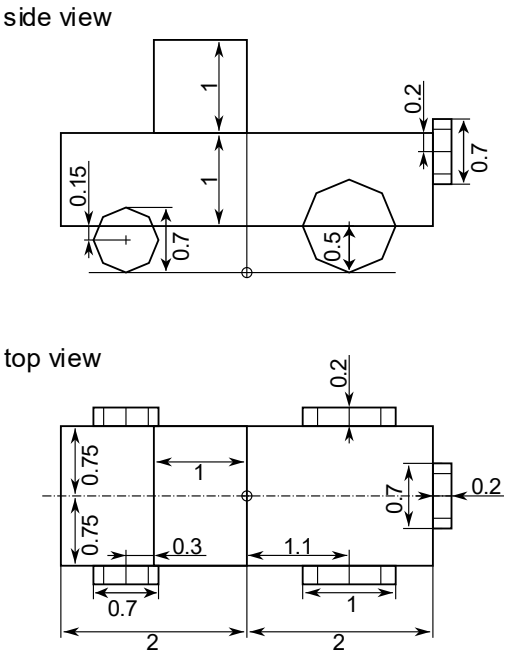
A template code is provided for this assignment. You can modify this code to build your own scene. Implementation can mainly be done in the main source file **assignment_3.cpp**, but feel free to create your own files (*e.g.*, it might be helpful to use a **pickup.cpp/h** and **ground.cpp/h** file) and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

The scene available in the template code contains a plane as basis for the ground, a cylinder, and a cube that can be rotated around its *x*- and its *y*-axis using the W, A, S, 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. Set up the hierarchical geometrical model of the pickup. For that, disable the scaled cube model from the template code. Then start with **2 cubes** and **5 cylinders** (size: 2x2x2) and apply transformations to them (**scaling, rotation, and translation**) to obtain the **pickup parts** representing: **base, cockpit, wheels, and a spare tire** (see introduction figure). All the necessary measurements for the different parts can be taken from the drawing below. You are free to design your own pickup, but it must include **at least the same number of parts**. In addition, the three transformations **scaling, rotation,** and **translation** must all be used for its creation. Use **different colors for the different parts** (see example images).



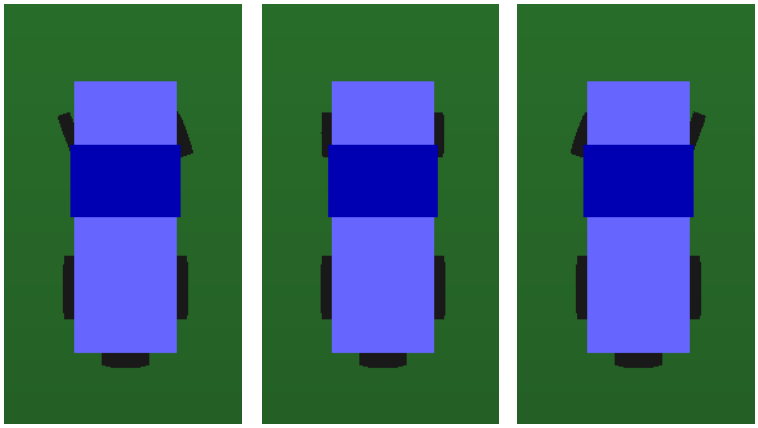
2. Add **user control** and **animation** to the pickup. It should be possible to drive the pickup around using the **W, A, S, D keys**. In addition, the **pickup wheels** should be **animated** and should **turn according to the pickup's movement**.

   - In a first step, it should be possible to control the pickup's forward and backward movement by pressing W and S, respectively. The pickup should move with a predefined velocity. Using the velocity together with the elapsed time since the last update step, the distance the pickup moves can be obtained. Further, to calculate the rotation angle α of the wheels, use the following formular:

     $$\alpha = \frac{\Delta x}{r}$$

     with r the wheel radius and Δx the travelled distance.

   - In a second step, the possibility to control the pickup steering, as well as the steering animation should be added. When pressing A/D, the front wheels of the pickup should turn left/right for a predefined turning angle θ (see example figure below). Furthermore, if the pickup moves forward or backward, it should turn. To calculate the angle that the pickup turns for every meter it travels, use the method **calculateTurningAnglePerMeter(…)**, which is in the main source file **assignment_3.cpp**. It takes as input the predefined turning angle of the pickup (**turningAngle**), the distance between the axles (**wheelBase**) and the **width** of the pickup. This constant can then be used to calculate how much the pickup turns in each update step depending on the travelled distance.



3. Next, create **wavy terrain** for the pickup to drive over. The displacement (offset in the y-axis) for every vertex of the plane has to be calculated only once, as the displacement should be constant and independent of the time. Furthermore, as this project doesn't have shading yet, **linearly interpolate between two colors** depending on the height of the ground to better visualize the wavy terrain.

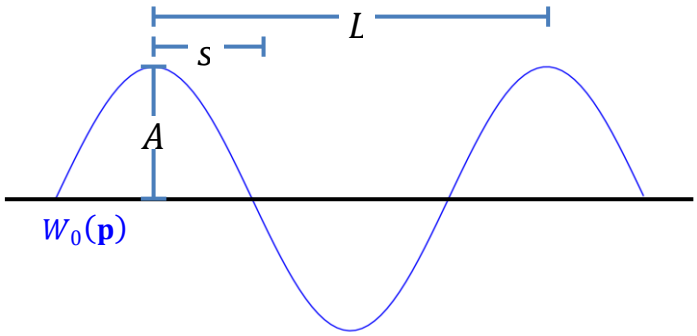
The displacement of the ground (at location $\boldsymbol{p}$) is modeled with a sum of periodic waves:

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

each defined with varying amplitude A (wave height), frequency ω (wavelength $L = \frac{2}{\omega}$), and direction $\boldsymbol{d}$:

$$W_i(\boldsymbol{p}) = A\sin\big(\omega(\boldsymbol{p}\cdot\boldsymbol{d})\big).$$

In our solution, we used <u>four</u> wave functions with the following parameters (see struct *Ground* in **ground.h**):



$$\begin{array}{lll} A_0 = 0.9, & \omega_0 = 0.35, & \boldsymbol{d_0} = [\mathbf{0},\mathbf{1}]^T/\|d_0\| \\ A_1 = 0.7, & \omega_1 = 0.4, & \boldsymbol{d_1} = [\mathbf{1},\mathbf{0}]^T/\|d_1\| \\ A_2 = 1.1, & \omega_2 = 0.1, & \boldsymbol{d_2} = [-\mathbf{2},\mathbf{1}]^T/\|d_2\| \\ A_3 = 0.3, & \omega_3 = 0.8, & \boldsymbol{d_3} = [-\mathbf{1},-\mathbf{2}]^T/\|d_3\| \end{array}$$

<u>Note:</u> By OpenGL convention, the ground is placed in the *xz*-plane with the y-axis representing the vertical offsets.

4. Now, the pickup should drive along the wavy terrain. Adapt the pickup's **rotation** and **position** according to the underlying wave displacement. The translation along the y-axis can be determined by taking the displacement at all the wheels' contact points and averaging their displacements.

   To determine the rotation of the pickup, you can use the orientation of a triangle placed at the bottom of the pickup's current position. That is, define a triangle in the xz-plane with the corners at the two back tires and one between the front tires. Calculate the height of each of the triangle corners and use the resulting 3D points to compute a rotation matrix with which the boat can be aligned with the waves.

   Hint: A rotation matrix is a square matrix whose columns and rows are orthonormal vectors. As a result, finding three orthonormal vectors that represent the target orientation can be stacked to retrieve the required rotation matrix.

5. Finally, implement a **second camera mode**. The first mode is very similar to the camera available in the template code. But instead of initially looking at the center of the cube, the camera should initially look at the origin of the world coordinate system (i.e., $(0,0,0)^T$). Further, in the second mode the camera focus should follow the pickup. Use the keys **1** and **2** to switch between the two modes.

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 17th, 2025 (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 model (**3 marks**)
- Wavy terrain (**2 marks**)
- Pickup control and animations (**7 marks**)
- Additional camera mode (**1 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!