Proseminar Visual Computing
Winter Semester 2024

Assignment 4
Hand-out: November 26, 2024
Hand-in: December 09, 2024

Topics

- Shader programming
- Lighting and materials
- Light sources

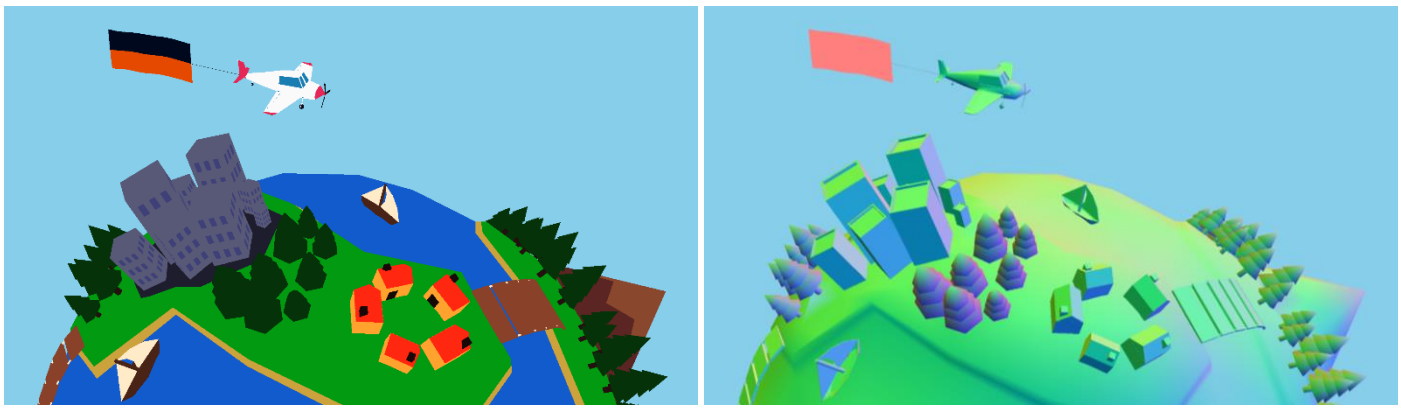
Outline

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. The objective of this assignment is to a) implement *Blinn-Phong Illumination* (with multiple light sources) in the *Fragment Shader* in GLSL using material properties defined in the .obj data format, and b) animate the flag via the *Vertex Shader*.

Template code

A template code is provided with this assignment. It loads the plane, flag, and planet meshes and their material properties from .obj files. The plane controls and all animations (planet, plane, flag) are already implemented. Currently, two render modes are available (toggle by pressing *R*).

In one mode, the fragment shader sets the fragment color according to the diffuse material, and in the other mode the fragment shader sets the fragment color according to the normals.



Tasks

1. Contrary to the first assignment, the animation of the flag should be handled on the GPU with a specialized *Vertex Shader*. The position of each vertex is given as $\mathbf{v}_{pos} = (x, y, z)^T$. Using this, the displacement of the flag (at location $\mathbf{p} = (y, z)^T$) can be 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 $L = \frac{2}{\omega}$), direction $\mathbf{d} = (d_1, d_2)^T$ and z-position z_{min} at which the flag is connected to the flag connector and therefore should stay still:

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

Implement the displacement of the flag over time in the *Vertex Shader*. That is, for each vertex calculate its corresponding wave height and translate the position (its x-coordinates) accordingly. In addition, you need to provide the surface normal to the *Fragment Shader* (for lighting calculations). This can be calculated by taking the partial derivatives of $H(\mathbf{p}, t)$ with respect to y and z (*i.e.*, the coordinates of $\mathbf{p}$):

$$\mathbf{n} = \begin{pmatrix} \frac{\partial H(\mathbf{p},t)}{\partial y} \\ 1 \\ 0 \end{pmatrix} \times \begin{pmatrix} \frac{\partial H(\mathbf{p},t)}{\partial z} \\ 0 \\ 1 \end{pmatrix}.$$

As a last step, the orientation of the normal must be determined. This should be done in the fragment shader (only for the flag). The correct direction of the normal depends on which side of the flag is seen. A simple check is to calculate the dot product between the normal $\mathbf{n}$ (in **world** space) and the view vector $\mathbf{v} = \text{normalize}(\mathbf{c}_{\text{pos}} - \mathbf{f}_{pos})$ (vector from fragment position to camera position $\mathbf{c}_{\text{pos}}$, both in **world** space). If the result is smaller than zero the normal must be flipped.

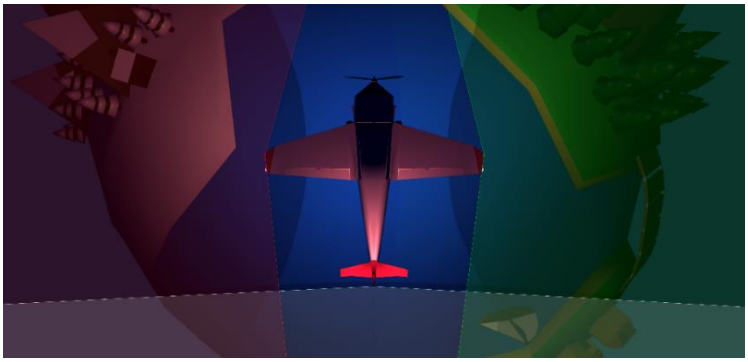
Note: By OpenGL convention, the flag is placed in the *yz*-plane with the x-axis representing the horizontal offsets. The above equations are already given in the correct coordinates. Additionally, you should normalize the vector $\mathbf{n}$ and do not forget about the transformations to the correct coordinate spaces. Finally, the render mode that uses the normals for shading might be very helpful for debugging.

2. Implement *Blinn-Phong* illumination in the *Fragment Shader* for a directional light source with the material properties (color values and shininess) given in the models (loaded from the .obj (.mat) files; already implemented – see *model.h*).

   Find color values and the direction of the light source to emulate the lighting condition during day and night. You should be able to switch between them by pressing a key. In addition, the direction of the light source should change over time (like the sun does during 24h). This means that the light direction must be rotated with respect to the coordinate system of the planet (pay attention that in the template code, the planet rotates, and the plane is fixed).

3. Next, add point light sources with decaying intensity. There should be *one red light* on the left wing, *one green light* on the right wing and on each wing *one flashing white strobe*. In addition, there should be *one white light* and *one flashing red strobe* on the rudder. The light source positions and directions are provided as constants in the *assignment.cpp* file. Make sure to translate the light source positions as the plane moves.

4. Change the *point lights to spotlights*. You can implement this without a smooth edge and simply cut the lights power if a fragment is not in the light cone (defined with a direction and a cut-off angle). The following image shows an example, with the *lights* (not blinking) on the wings and rudder, having different cones. You can specify your own setup, but make sure that at least two of the lights and/or strobes are *spotlights* (spotlight with cut-off angle of 180 degrees gives point light).



5. Enable and disable the emission of the plane lights according to the lights being on/off (*e.g.*, due to blinking, on and off toggle of all plane lights). Similarly, enable and disable emission on the planet depending on day or night settings. There is a function *setEmission* for the plane and the planet which toggles all materials with non-zero emissions.

Implementation Remarks

Make sure that your code is clear and readable. Write commentaries when necessary. Your solution should contain a readme file with names of the team members, 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 December 9th, 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: **A4_<lastname1>_<lastname2>_<lastname3>**

Archive: **A4_<lastname1>_<lastname2>_<lastname3>.zip**,

where <lastname1>, etc. are 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:

- Shader based flag animation (**4 marks**)
- A directional light source (sun/moon) (**4 marks**)
- Point light sources attached to the plane (**4 marks**)
- Spotlight sources (**1 marks**)
- Emission (**1 marks**)
- Code readability, comments, and proper submission: (**1 marks**)

Resources

- Lecture and Proseminar slides as well as code and information 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 for Blinn-Phong Illumination (Phong Shading)
  https://learnopengl.com/Lighting/Basic-Lighting
  http://www.opengl-tutorial.org/beginners-tutorials/tutorial-8-basic-shading/
- GL Framework GLFW
  https://www.glfw.org/documentation.html

Note: Be mindful of employed OpenGL and GLSL versions!