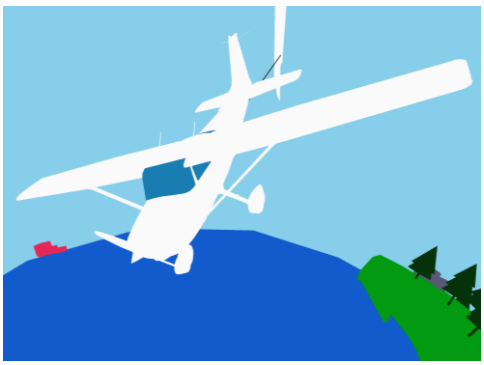
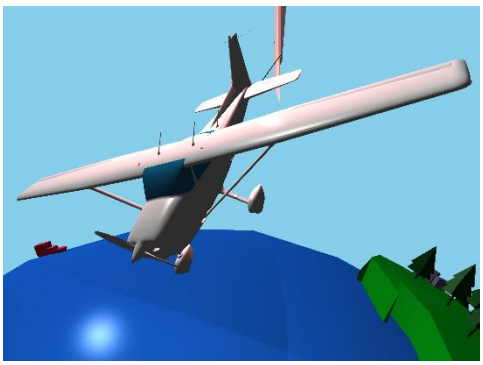
Proseminar Visual Computing
Winter Semester 2024

Assignment 5
Hand-out: December 03, 2024
Hand-in: January 06, 2025


Topics

- Texturing
- Shader Programming
- Skybox and Environment Mapping

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 enhance the visual appearance of the scene by using textures. Material properties for *Blinn-Phong* illumination should be read from textures instead of using material colors. Furthermore, a *Skybox* should be added and used to apply *Environment Mapping*.

Template code

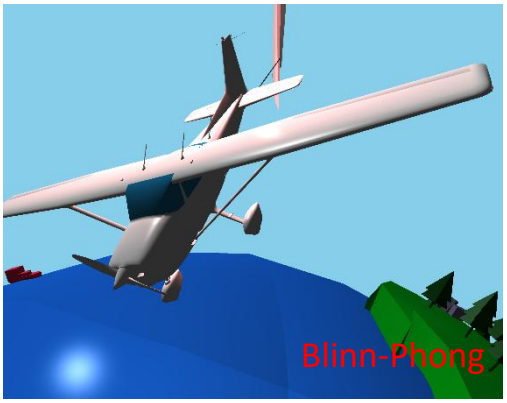
A template code is provided with this assignment. It loads the plane, flag and planet mesh, material properties, and textures from .obj files. The plane controls and all animations (planet, plane, flag) are already implemented. In the initial handout, the same render modes as in the template code for assignment 4 are available (diffuse material and normal rendering, toggle with *R*). After the submission deadline of assignment 4 (December 10), an updated template

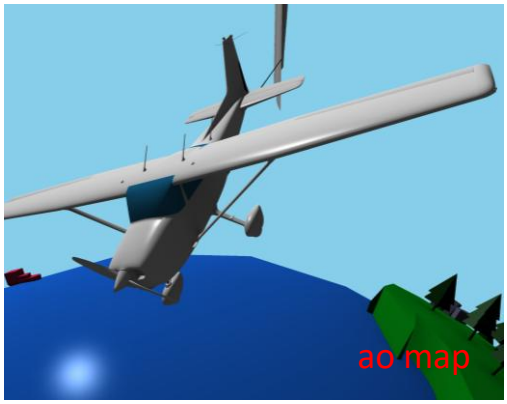
code including the flag animation and *Blinn-Phong illumination* from the previous assignment will be made available (no textures used for rendering).

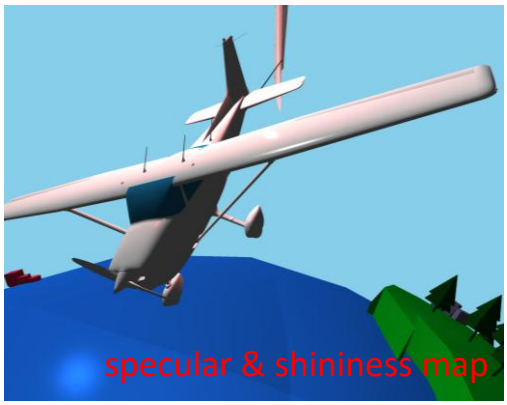
Further, code for loading a cube map (skybox) is available in both versions in the files *src/mygl/cube_map.h* and *src/mygl/cube_map.cpp*. Two example cube maps which you can use are provided in the folder *assets/skybox*.

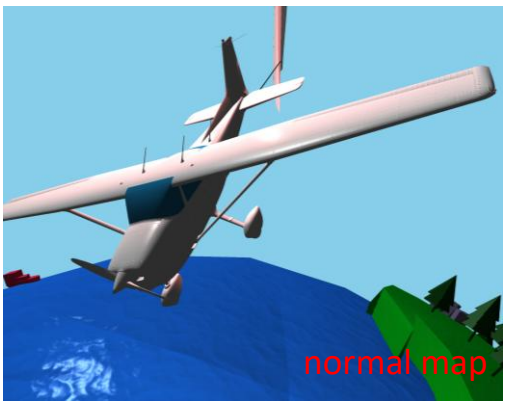
Tasks

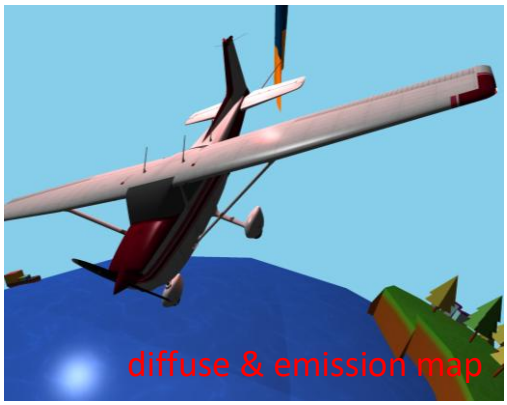
1. If required (initial template code), add *Blinn-Phong* and the flag animation (following all steps in assignment 4). Then, adapt *Blinn-Phong* illumination to use the loaded texture maps (*map_diffuse, map_ambient, map_specular, map_shininess, map_emission, map_*normal) to retrieve material properties for each fragment instead of the constant color values per surface material.

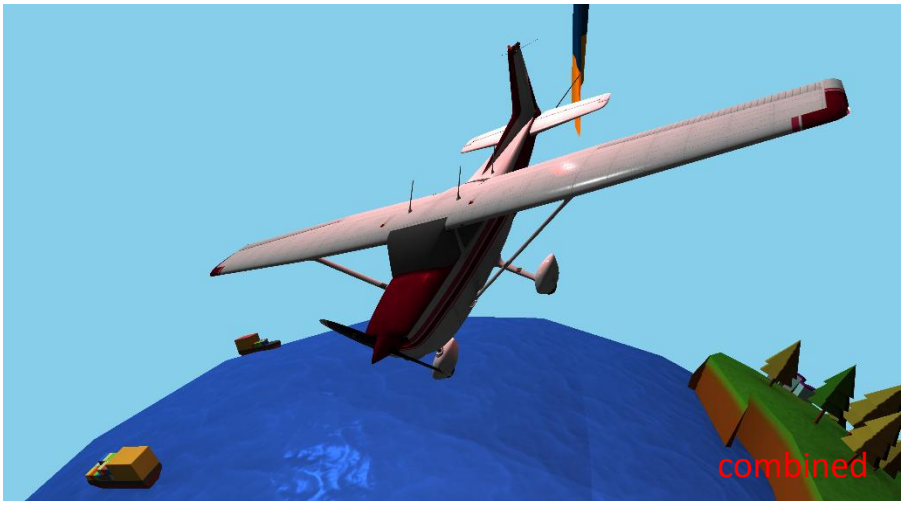












In the template code the textures are loaded so that all texels on the GPU are between 0 and 1 and all textures have RGBA channels. For images with no alpha channel in the input file, alpha is set to 1. For grayscale images (only one channel), the two missing channels (BG) are set to the same as the first one (R). To obtain the correct values that can be applied in the Blinn-Phong model, for some textures additional calculations must be performed:

- The value from the ambient occlusion map represents the strength of the ambient light reaching a certain surface position. Use this value to multiply the diffuse material to get the ambient material color (like how it is done without the texture).
- For the shininess N_s only one channel is required (*i.e.*, red). To transform the value read from the texture in the range required for the Blinn-Phong model, it must be multiplied with 1000.
- The normal maps contain normals retrieved from a high resolution model including small details. Using per fragment normals from this map for shading allows to represent details without increasing the complexity of the model. Usually, they are given in tangent space and need to be converted into object space and world space. For simplicity, the provided normal maps (planet, plane and flag) are given in **object space**. Texels of the normal maps, have three channels (*x*, *y*, *z*) ranging from 0 to 1. Since the normals are given in object space, they need to be convert to the range of -1 to 1 (encode directions over the whole sphere).
  Given a texel **t**, the object space normal $\mathbf{n}_o$ is calculated as follows:
  $$\mathbf{n}_o = \mathbf{t} \cdot 2.0 - 1.0.$$
  Further, you need to transform the normal into world space. As already discussed, given model matrix **M**, normals can be transformed as follows:
  $$\mathbf{n}_w = (\mathbf{M}^{-1})^T \mathbf{n}_o.$$
  These resulting normals can directly be used for the illumination of the plane and planet. For the flag, an additional step is required since the normals from the normal map do not change with the flag animation (here tangent space normals would be required). The actual orientations of the different vertices of the flag are given by the vertex normals $\mathbf{n}_v$ (from the vertex shader (assignment 4)). To add some cloth detail to the flag surface, a simplified estimate is to calculate the normals $\mathbf{n}_s$ as follows:
  $$\mathbf{n}_s = \mathrm{norm}(s \cdot \mathbf{n}_w + (1 - s) \cdot \mathbf{n}_v),$$
  where *s* is a scaling parameter for the flag texture (e.g. $s = 0.25$). Be aware, that the resulting normals are actually not the correct ones, but visually they look good enough. Similar as in assignment 4, the direction of the flag normal must be checked by calculating:
  $$d = \mathrm{dot}(\mathbf{n}_v, \mathbf{v}),$$
  with $\mathbf{v}$ being the view vector. If $d < 0$, $\mathbf{n}_s$ must be flipped.

Note: See assignment 4 for more details about the calculation of $\mathbf{n}_v$ and make sure that they are transformed into world space. Further, make sure to use $\mathbf{n}_v$ for the orientation check, but flip $\mathbf{n}_s$ if required.

2. Write shaders to render a *Skybox* using *Cube-Mapping*. Code that can be used to load the textures and create the geometry buffers of a *Cube Map* is available in the template code files *src/mygl/cube_map.h* and *src/mygl/cube_map.cpp*. To switch between day and night, multiply the sampled texture from the cube map with fitting colors. These might differ depending on the skybox used (*e.g.,* white for day and blueish gray for night).

3. Next, extend the *Blinn-Phong* rendering to include reflections of the environment. This is done using *Environment Mapping*: For each fragment, the reflection direction is calculated to sample from the cube map. To adjust the reflection strength of different materials, multiply the sampled color with the specular color (from specular map). Do not forget to also apply the day and night color (see previous task). Add this resulting fragment color to the color calculated with the Blinn-Phong illumination model.
   Note: See the resources below for a tutorial on cube maps and environment mapping.

4. In the next step, transform the *Skybox* so that it matches the planet's rotation. Make sure that the *Skybox* is also transformed during *Environment Mapping*.
   Note: The transformations for rendering the *Cube-Map* and for *Environment Mapping* are different.

5. Lastly, write/draw the names of the group members or something similar somewhere on the plane, by modifying its diffuse texture. It should be visible in the rendering of the scene. Feel free to redesign the flag as well, by changing its diffuse texture (optional).

Implementation Remarks

Make sure that your code is clear and readable. Write comments if 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 January 06th, 2025 (23:59). **Submit the sources** (i.e., only the content of the *src* folder) + the modified diffuse texture and a screenshot of the rendered plane with it 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: **A5_<lastname1>_<lastname2>_<lastname3>**

Archive: **A5_<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:

- Use textures to map material properties to fragment (**6 marks**)
- Skybox and environment mapping (**5 marks**)
- Skybox rotation (**1 marks**)
- Adapt the plane's diffuse texture **(1 mark)**
- Code readability, comments, and proper submission: (**2 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 with textures
  https://learnopengl.com/Lighting/Basic-Lighting
  https://learnopengl.com/Lighting/Lighting-maps
- OpenGL Tutorial for Cube Maps, Skyboxes and Environment Mapping
  https://learnopengl.com/Advanced-OpenGL/Cubemaps
- GL Framework GLFW
  https://www.glfw.org/documentation.html

Note: Be mindful of employed OpenGL and GLSL versions!