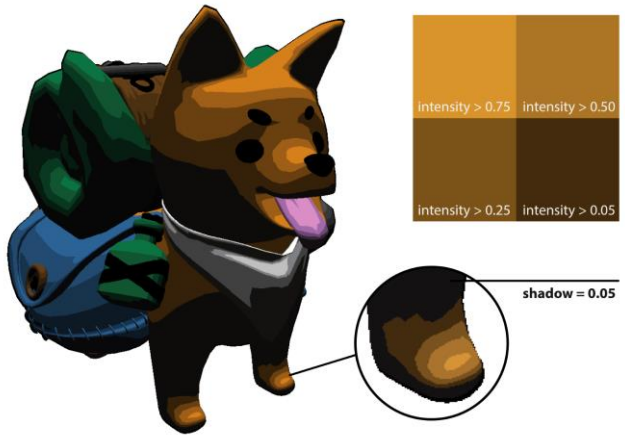
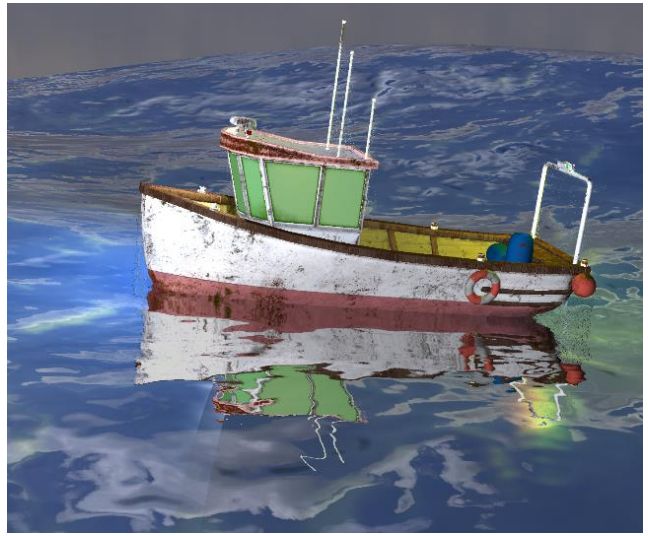
Visual Computing Proseminar
Final Projects

Computer Graphics





Final Project

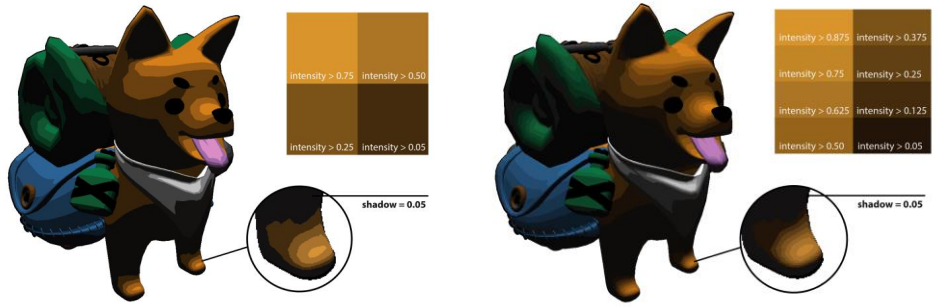
- 30 points (work in teams of 2-3 students)
- Select a topic (either Computer Vision or Computer Graphics)
  - **You can suggest your own topic**
  - Choose topic until **9. January 2024**
  - Send one mail per team to **all of us (Antonio, Niko, Stefan)** with topic and members
- Final presentation on **30. January 2024**
  - Teams with members in different groups must present together
- Final report on **06. February 2024**
  - 4 pages (double column layout provided by us)

Topics

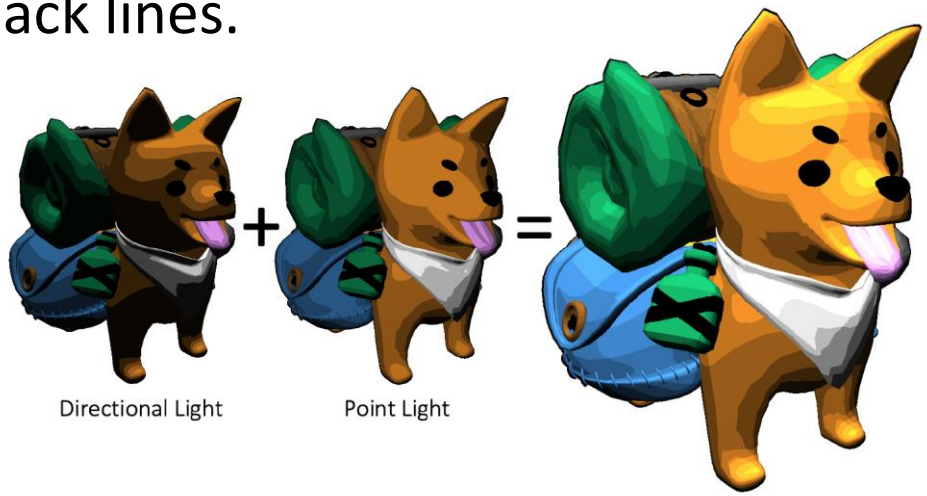
- Computer Vision Topics (more details in file ProjectsCV.pdf)
  - Image classification with SVM
  - Smile detection
- Computer Graphics Topics
  - Implement a demo of a selected algorithm (OpenGL, GLFW)
  - Cel shading with edge-detection filter
  - Shadow mapping
  - Screen-space reflection
  - Deferred rendering

1. Cel Shading with Edge-Detection Filter

- Implement non-realistic, cartoon-like rendering. The lighting of the surface is represented by flat areas of uniform colors, instead of smooth color variations. Further, outlines are highlighted through black lines.



Visual Computing Project: Stecher and Lintner 2022



Directional Light Point Light

- Technique uses <u>three</u> render passes
  1. Render scene with *cel shading* (intensity clamping) to custom framebuffer (color, depth)
  2. Render screen-filled quad to apply an edge-detection filter on the depth-buffer (*e.g.*, Sobel-filter)
  3. Render screen-filled quad and combine **color** and **edge texture** to get final image

1. Cel Shading with Edge-Detection Filter

- Links
  - https://en.wikipedia.org/wiki/Cel_shading
  - https://en.wikibooks.org/wiki/GLSL_Programming/Unity/Toon_Shading
  - https://www.lighthouse3d.com/tutorials/glsl-12-tutorial/toon-shading/

2. Shadow Mapping

- Build a 3D scene with several objects and a plane representing the ground. Use shadow mapping to cast the shadows from a directional light source in the scene.

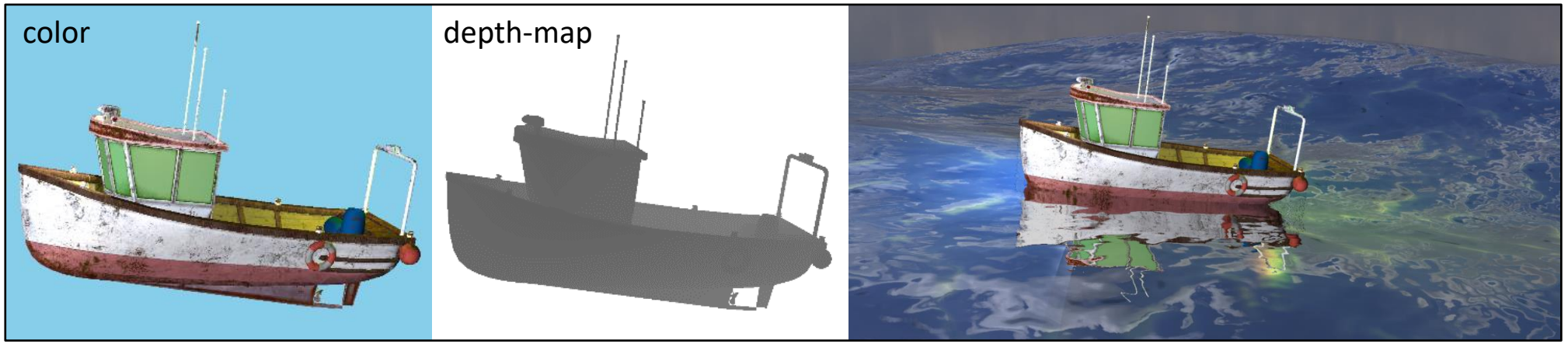
- Technique uses <u>two</u> render passes
  1. Scene is rendered from light source and depth buffer (shadow map) is stored
  2. Scene is rendered from camera; each fragment is tested if it is exposed to light or not

2. Shadow Mapping

- Links
  - https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping
  - http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/
  - https://ogldev.org/www/tutorial24/tutorial24.html

3. Screen-Space Reflections

- Extend the third assignment to include a simple implementation of screen-space reflections for the water surface. This requires to find ray geometry intersections by testing against the previously rendered depth-buffer.



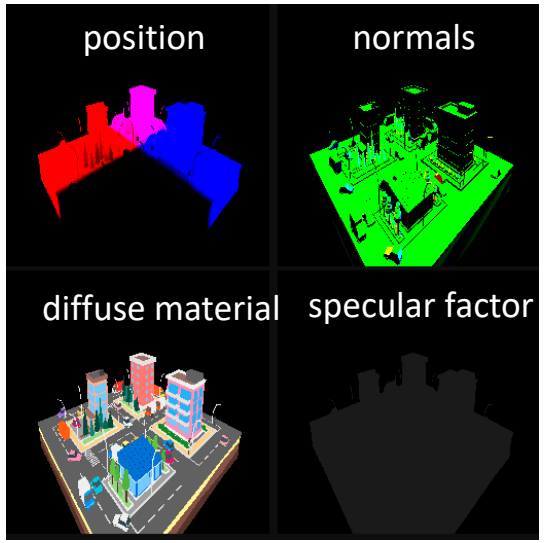
- A simple implementation could use three render passes
  1. Boat is rendered into a **custom** framebuffer (color, depth-buffer)
  2. Water is rendered and for each fragment a reflection ray is traced against this depth buffer
  3. Boat is rendered over the water surface into the **default** framebuffer

3. Screen-Space Reflections

- Links (for basic understanding; implementation can be simplified)
  - https://lettier.github.io/3d-game-shaders-for-beginners/screen-space-reflection.html
  - https://virtexedge.design/shader-series-basic-screen-space-reflections/
  - https://sugulee.wordpress.com/2021/01/16/performance-optimizations-for-screen-space-reflections-technique-part-1-linear-tracing-method/

4. Deferred Shading

- Deferred shading is an optimization for scenes with many light sources; by computing the lighting model after “collecting” visible fragments.



- Technique uses <u>two</u> render passes
  1. <u>Geometry pass:</u> render the scene once to retrieve geometric information
  2. <u>Lighting pass:</u> render screen-filled quad and compute light model per fragment using the information collected in the previous pass

4. Deferred Shading

- Links
  - https://learnopengl.com/Advanced-Lighting/Deferred-Shading
  - https://software.intel.com/content/dam/develop/external/us/en/documents/lauritzen-deferred-shading-siggraph-2010-181241.pdf

Frequently Asked Questions

- Where can I get free assets?
  - https://sketchfab.com/feed, https://casual-effects.com/data/index.html
- GUI library for our computer graphics project?
  - https://github.com/ocornut/imgui -- Example is provided in OLAT and it will be a PS topic in January.
- Could we get more details on our project?
  - Of course, please contact us if there are any questions.
    For computer vision please contact Antonio; computer graphics Niko and Stefan.
- Is it ok to copy code from tutorials and external resources?
  - Generally, make sure to develop your own solutions.
    In small parts (*e.g.*, a specific equation, data parsing) it is fine, but you **must** cite your sources.
  - Note, that we plagiarism scan against previous projects of this course.
    Do not copy text and/or code from previous years.

Frequently Encountered Problems

- The model loader in the assignments assumes triangle meshes
  - In Blender **File/export .obj** select **Geometry → Triangulate Faces**
- Some models in *sketchfab* have inverted textures
  - In texture.cpp under **textureLoad** you can modify **stbi_set_flip_vertically_on_load(true);**
- Model loader can’t find the material textures
  - Check the texture paths in the .mtl file
  - Compare the .mtl file with the assets from the assignment