Lesson 1 – Introduction

PV227 – GPU Rendering

Jiří Chmelík, Jan Čejka
Fakulta informatiky Masarykovy univerzity

18. 09. 2017

Outline

Outline

Course

Teaching method = seminars $\rightarrow$ **active** participation ...

Course

Teaching method = seminars $\rightarrow$ **active** participation ...

Figure: Taken from weebly.com

Course – curriculum

- Introduction, Repetition, Toon Shading
- Shadows
- Deferred shading
- SSAO, DoF
- HDR, bloom
- Particle systems, compute shaders
- Geometry shaders
- Tessallation shaders
- Microfacets
- Physically Based Rendering, IBL
- Vulkan
- Parallax Occlusion Mapping

Course

Used technologies:

- Windows
- Visual Studio 2015
- C++
- OpenGL
- Libraries
  - FreeGLUT
  - GLM
  - GLEW
  - DevIL
  - AntTweakBar

Requirements

To successfully pass the course:

- no more than 2 absences,
- two assignments:
  - individual, home work,
  - two weeks limit,
  - oral presentation.

Expectations:

- programming skills: C, C++
- knowledge of OpenGL (PV112)
- basic knowledge of basics principles of computer graphics (PB009)

Requirements

To successfully pass the course:

- no more than 2 absences,
- two assignments:
  - individual, home work,
  - two weeks limit,
  - oral presentation.

Expectations:

- programming skills: C, C++
- knowledge of OpenGL (PV112)
- basic knowledge of basics principles of computer graphics (PB009)

Contacts

- Jiří Chmelík
  - office: A412
  - e-mail: jchmelik@mail.muni.cz
- Jan Čejka
  - office: A419
  - e-mail: 324987@mail.muni.cz

Want to know more about GPUs?

PV197 – GPU Programming, Jiří Filipovič:

- Introduction: motivation for GPU programming, GPU architecture, overview of parallelism model, basics of CUDA, first demonstration code
- GPU hardware and parallelism: detailed hardware description, synchronization, calculation on GPU – rate of instruction processing, arithmetic precision, example of different approaches to matrix multiplication – naive versus block-based
- Performance of GPUs: memory access optimization, instructions perormance, example of matrix transposition
- CUDA, tools and libraries: detailed description of CUDA API, compilation using nvcc, debugging, profiling, basic libraries, project assignment
- Optimization: general rules for algorithm design for GPU, revision of matrix multiplication, parallel reduction
- Parallelism in general: problem decomposition, dependence analysis, design analysis, parallel patterns
- Metrics of efficiency for GPU: parallel GPU and CPU usage, metrics for performance prediction of GPU code, demonstration using graphics algorithms, principles of performance measurement
- OpenCL: introduction to OpenCL, differences comparing to CUDA, exploiting OpenCL for hardware not accessible from CUDA
- ...

Outline

Outline

Shader Languages

- Cg (C for Graphics), by NVIDIA – no longer under active development,
- HLSL (High Level Shading Language), by Microsoft,
- GLSL (OpenGL Shading Language), by Khronos Group.
- Vulkan + SPIR-V, by Khronos Group.

Shader Languages Comparison

- almost the same capabilities,
- conversion tools exist,
- Cg and HLSL very similar (different setup),
- HLSL DirectX only, GLSL OpenGL only, Cg for both $\rightarrow$ different platforms supported.

Chosen Language

We will use GLSL in this course:

- open standard (same as OpenGL),
- no install needed,
- all platforms, all vendors.

Will will use GLSL 4.30 for OpenGL 4.3

- newer features will be mentioned but not demonstrated,
- NVIDIA 400 family supports OpenGL 4.5
- NVIDIA 600 family supports Vulkan

Outline

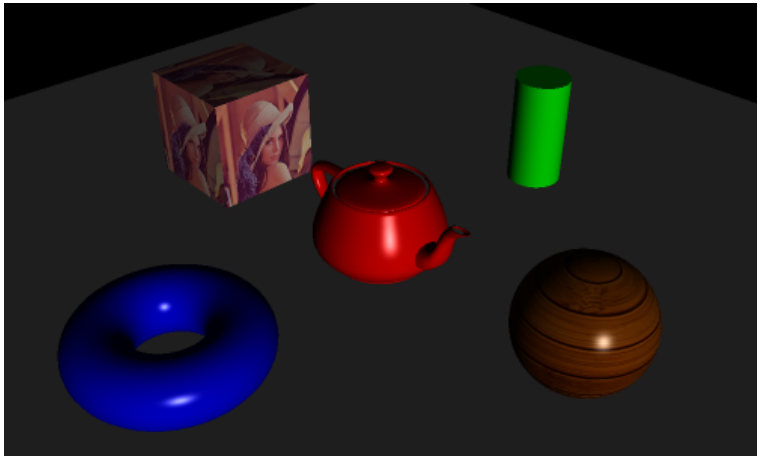
Repetition of knowlegde

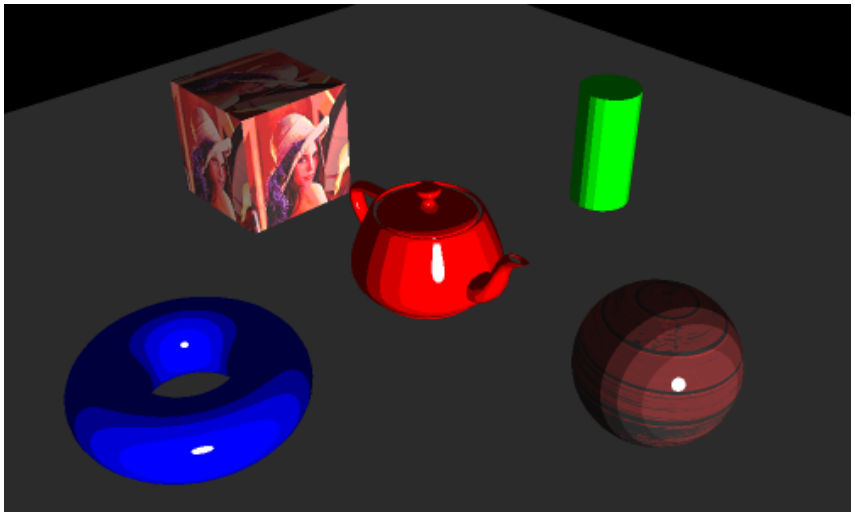
Repeat the knowledge of PV112, everyone at home as homework

- See Repetition.pdf in IS
- Understand project Repetition in IS

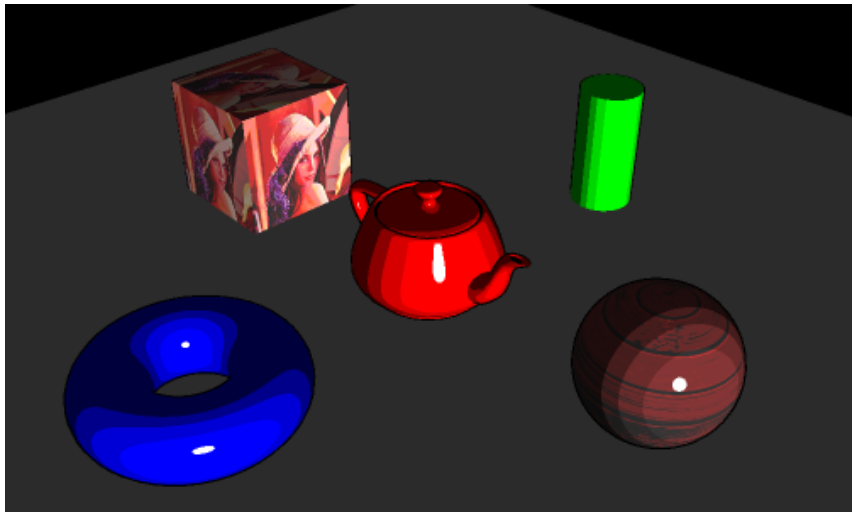
Outline

1. Lesson: Toon-shading, cel-shading





Toon-shading



Cel-shading

Source code

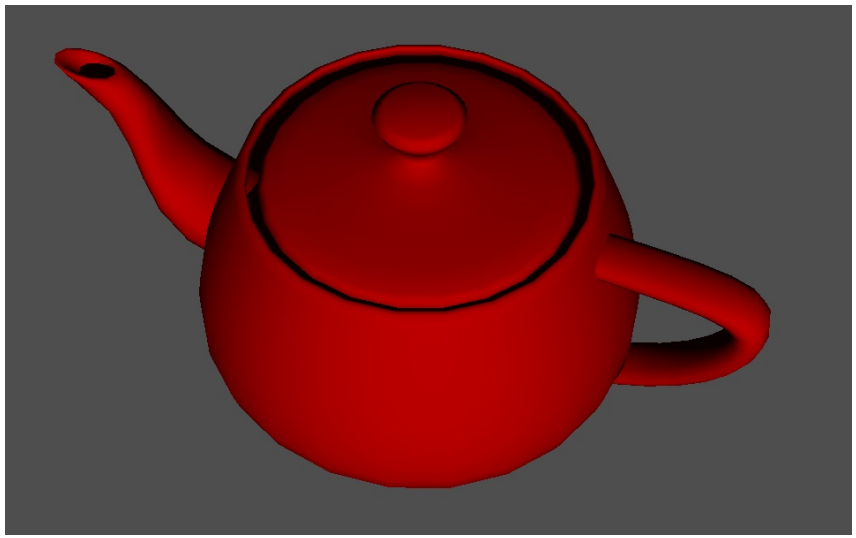
Download and prepare source code from IS:

- Download and unzip **PV227.zip**
- Download and unzip **Cv1.zip**

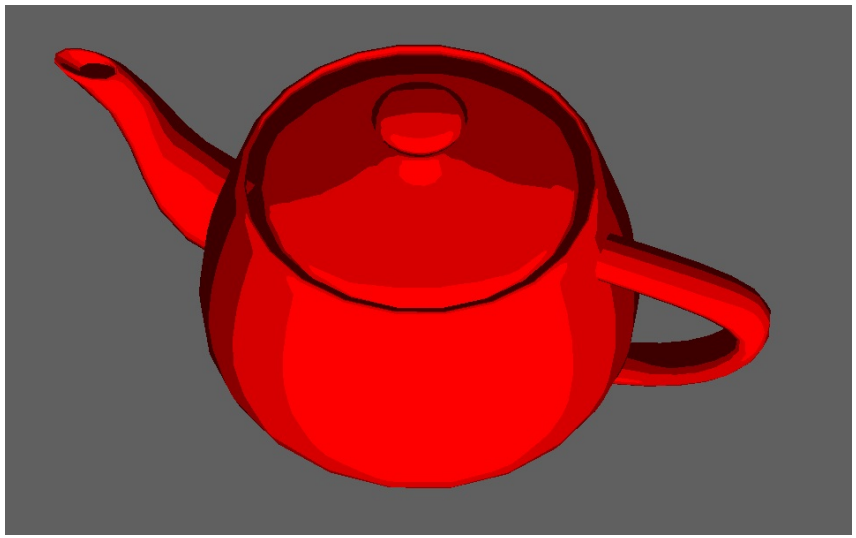
Open **Cv1.sln** in Visual Studio, and compile and run the source code.

Toon-shading

Diffuse lighting: use only several intensities of the light

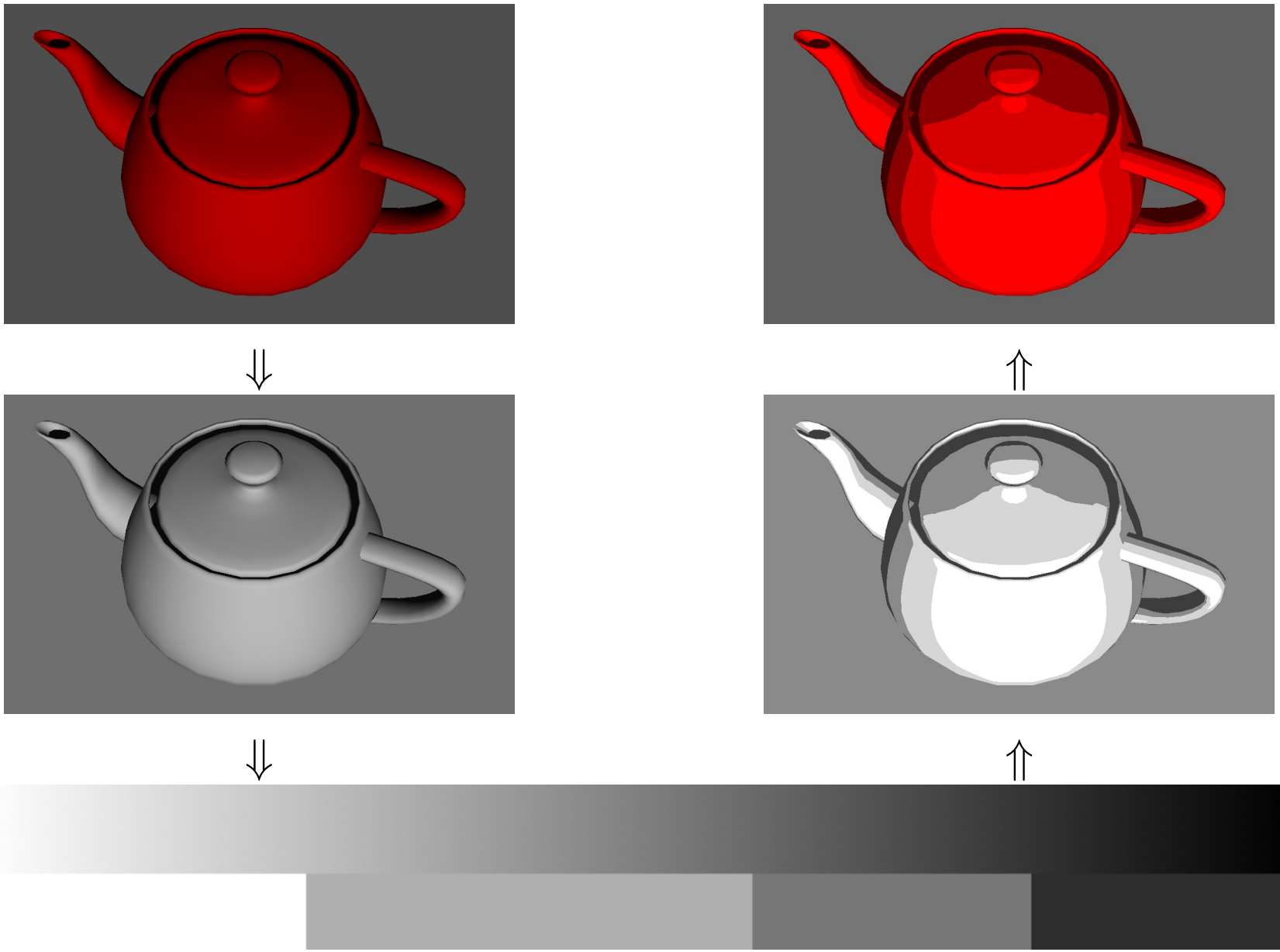


Without toon-shading



With toon-shading

Toon-shading: diffuse lighting



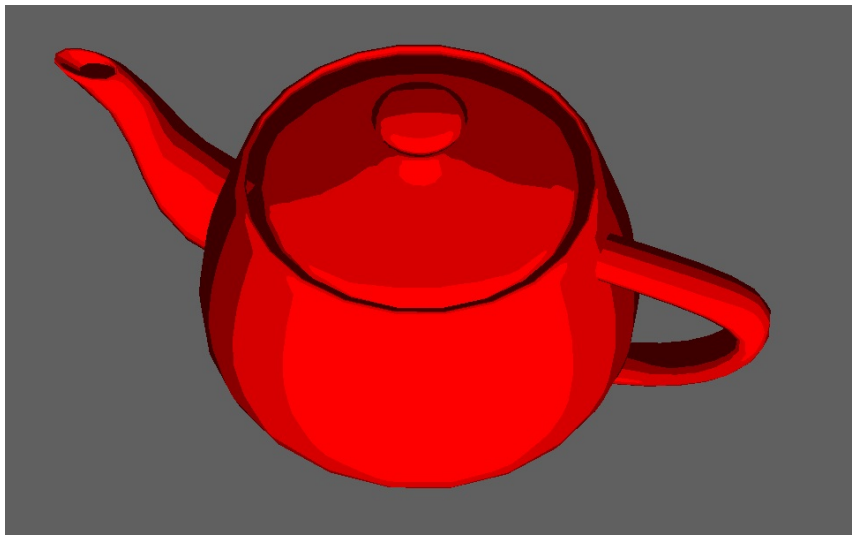
Toon-shading

- Task 1: Implement toon-shading for diffuse lighting for objects without textures.

  Hint: Look for “Task 1” in notexture_fragment.glsl

Toon-shading

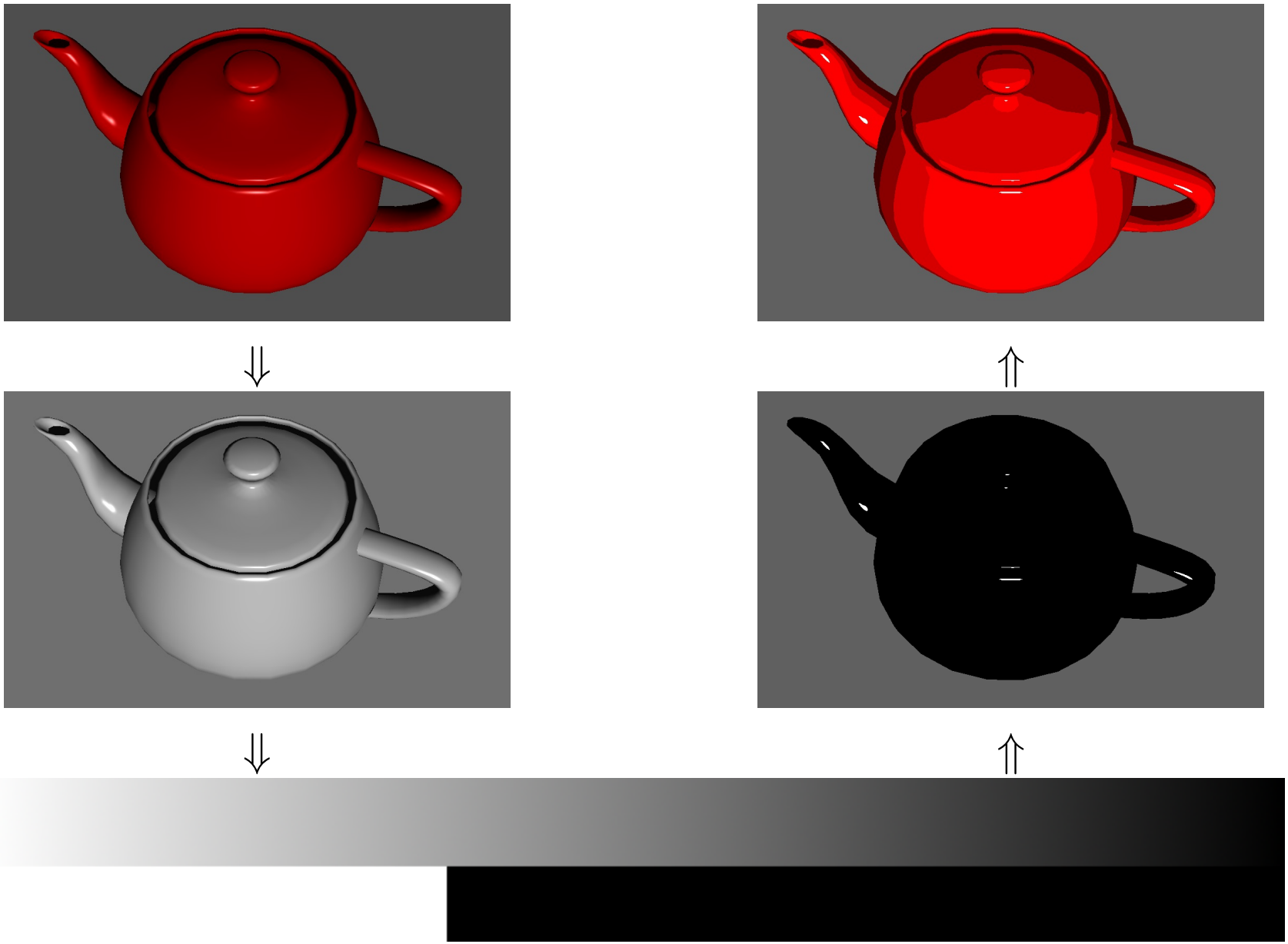
Specular lighting: the same as diffuse lighting, usually only a single white intensity.



Without specular highlight

With specular highlight

Toon-shading: specular lighting



Toon-shading

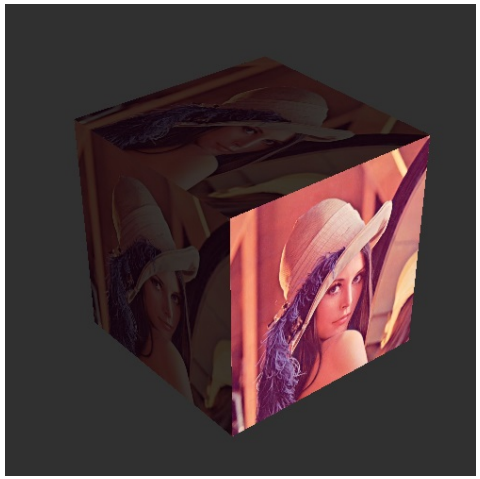
- Task 2: Implement toon-shading for specular lighting for objects without textures.

  Hint: Look for “Task 2” in notexture_fragment.glsl

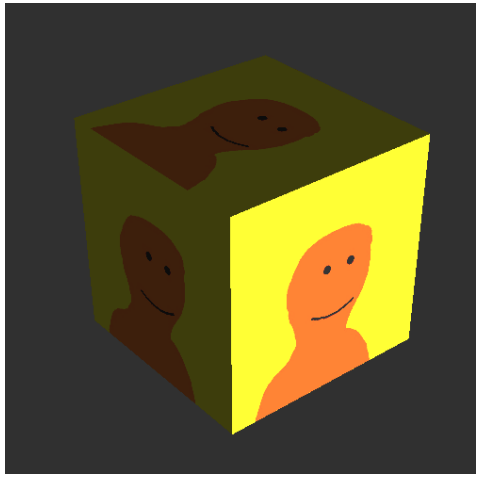
Toon-shading: textures

What about textures?

- Use the textures you have

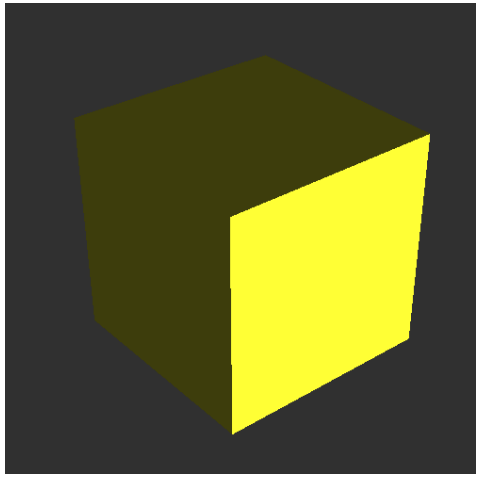


- Create special textures



- Threshold the texture

- Do not use textures



Toon-shading

- Task 3: Implement toon-shading for diffuse and specular lighting for objects with textures.

  Hint: Look for “Task 3” in texture_fragment.glsl

- Task 4: Threshold the color of the texture

  Hint: Look for “Task 4” in texture_fragment.glsl

Outline

Cel-shading

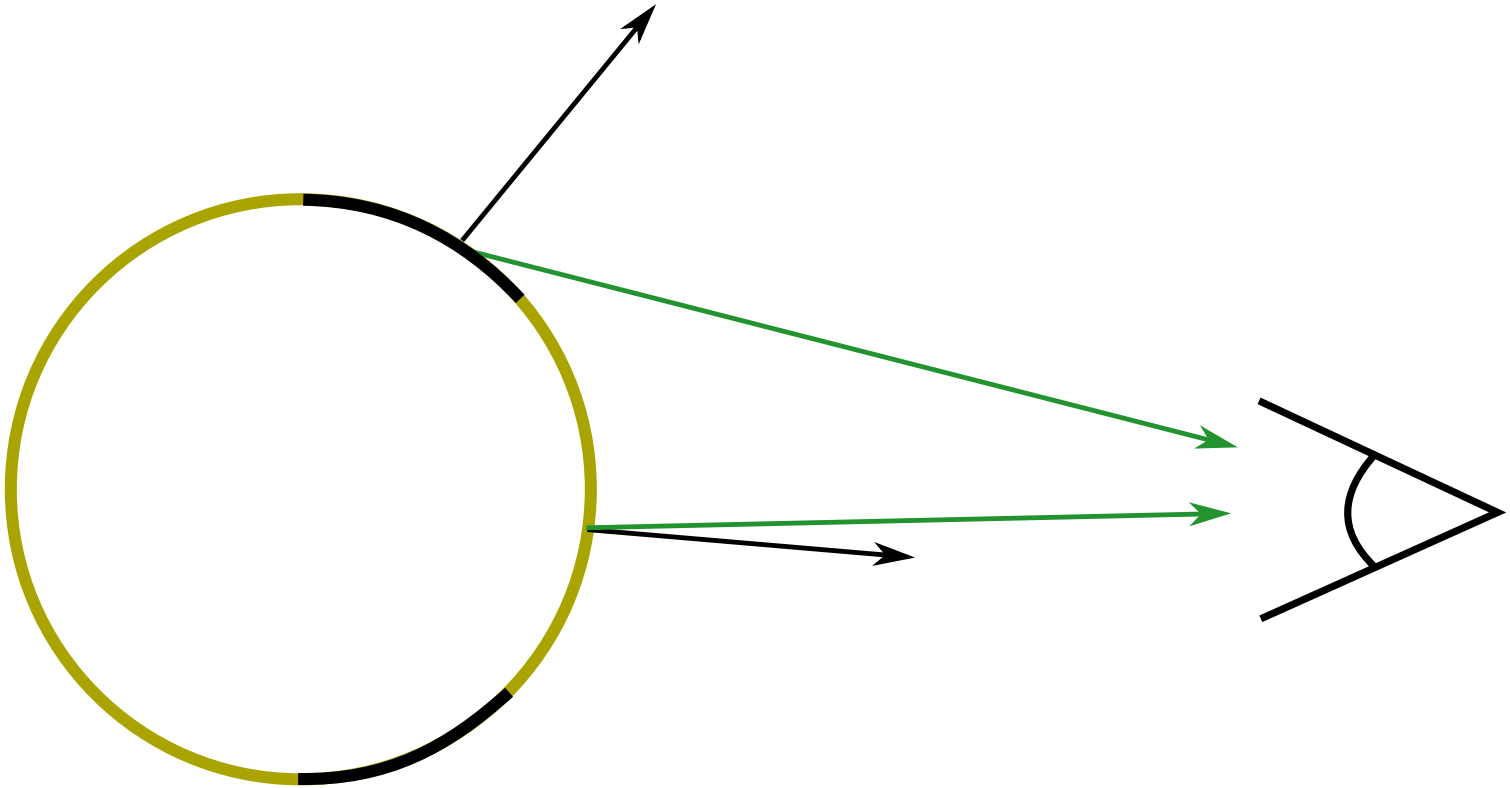
Adds contours around objects.

Without contours

With contours

Contours, method 1: view direction

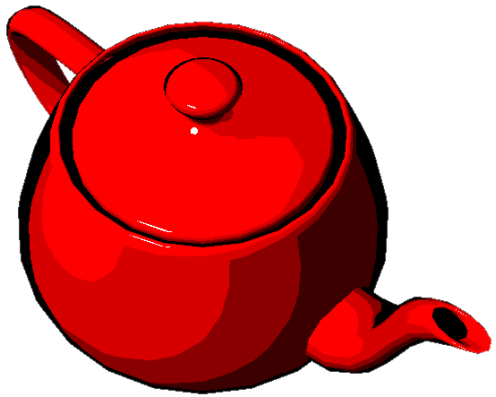
Check the angle between the normal and the view direction



black if $\vec{n} \cdot \vec{l} < threshold$

Contours, method 1: view direction

Good for round objects, bad for flat objects



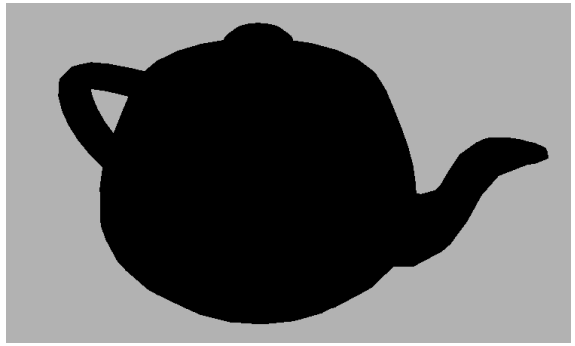
Good

Bad, no contour

Bad, left side is black

Contours, method 2: enlarge object

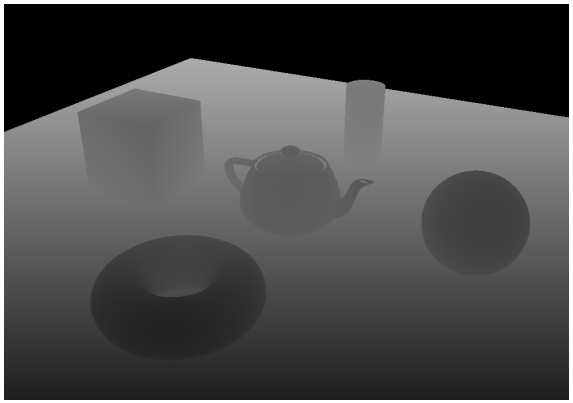
Render black back faces of objects enlarged a bit along their normals,



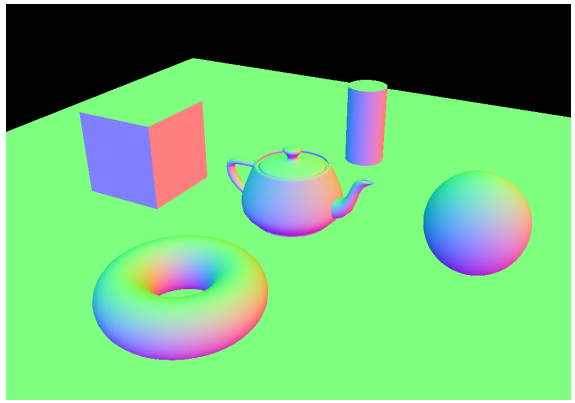
then render the front faces as usual

Contours, method 3: Postprocessing

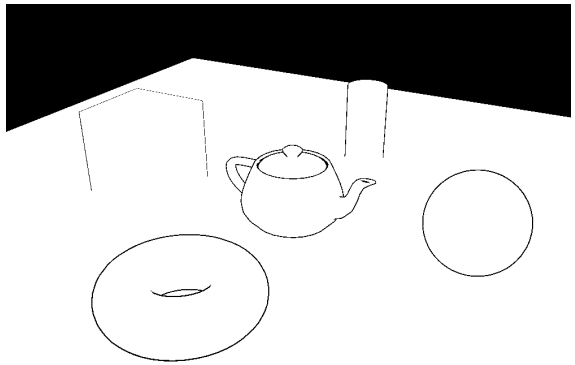
Use postprocessing to detect edges



Start with depths,



or normals,



and detect edges.

Cel-shading

Choose one (or both if you are fast):

- Task 5a: (easier) Implement the first method, i.e., inspect the view direction.

  Hint: Look for “Task 5a” in notexture_fragment.glsl and in texture_fragment.glsl

- Task 5b: (harder) Implement the second method, i.e., render the objects twice, first only the black faces in black, and then the front faces in a standard way.

  Hint: Look for “Task 5b” in Cv1_main.cpp (two places).

Outline

Food for thought

- How to solve more lights? Threshold separate lights and then add the result together? Or vice versa, first add the lights together and then threshold the result?
- How to solve transparent objects, like glass, fog, or smoke?

Borderlands 2

Food for thought

- How about different thresholds for different materials? Think about 1D textures.

- How about different styles of the lines?

Taken from gatheryourparty.com

Topic of next seminar: Shadows