Graphics programming Exercise 12

Henrique Debarba

IT University of Copenhagen

Exercise 12

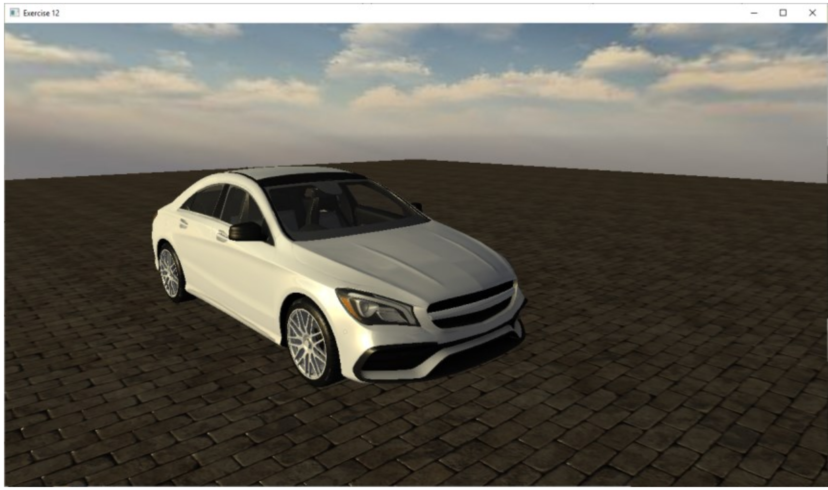
- Learning objectives
  - Implement the shadow mapping algorithm
  - Perform shadow computations in light space
  - Explore the render passes and pipeline of deferred shading.
  - Describe the implementation differences between forward shading and deferred shading.
  - Implement simple image post-processing with deferred shading.

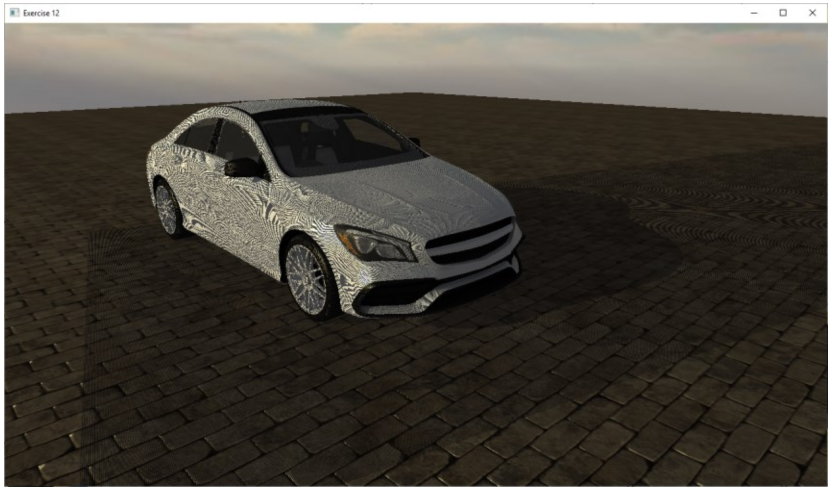
References

- Render to texture
  - https://learnopengl.com/Advanced-OpenGL/Framebuffers
- Shadow mapping
  - https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping
  - Cube map for point light in a closed environment: https://learnopengl.com/Advanced-Lighting/Shadows/Point-Shadows
  - https://docs.microsoft.com/en-us/windows/win32/dxtecharts/cascaded-shadow-maps
  - https://docs.microsoft.com/en-us/windows/win32/dxtecharts/common-techniques-to-improve-shadow-depth-maps
- Deferred shading
  - https://learnopengl.com/Advanced-Lighting/Deferred-Shading

Exercise 12.1 – depth comparison

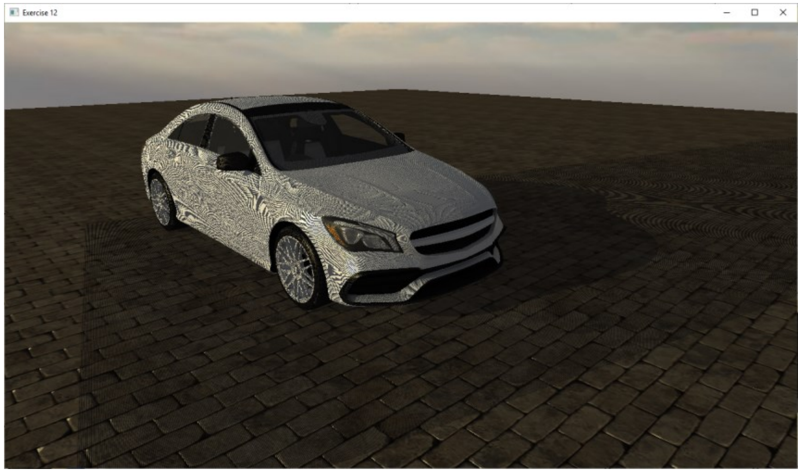
- Implement the basic shadow mapping algorithm.
- Vertex shader: Transform the vertex position from local space to light space.
- Fragment shader: Implement the ShadowCalculation function. You will need to:
  - Change the (x,y) value range from NDC [-1, 1] to uv texture coordinates [0, 1];
  - Sample the shadowmap with the coordinates;
  - Check if the current fragment depth in the scene is bigger than the depth sampled in the shadow map;
  - If it is, then the fragment is in shadow. Set shadow to 1.0.
- Add the shadow value to modulate your FragColor computation.

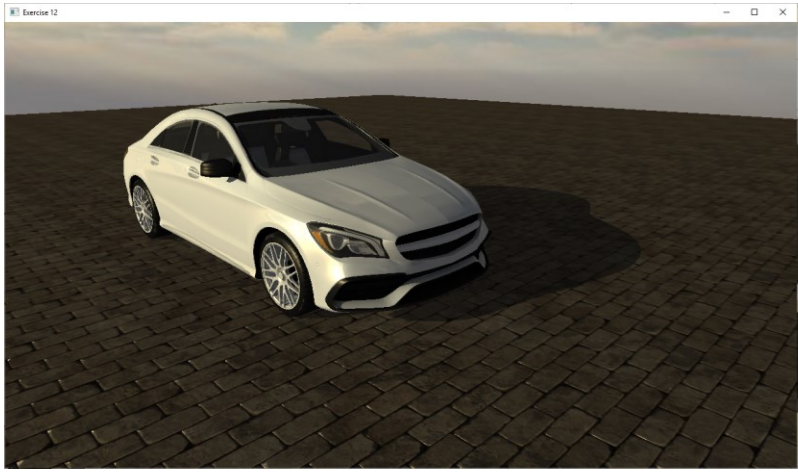




Exercise 12.2 – the acne problem

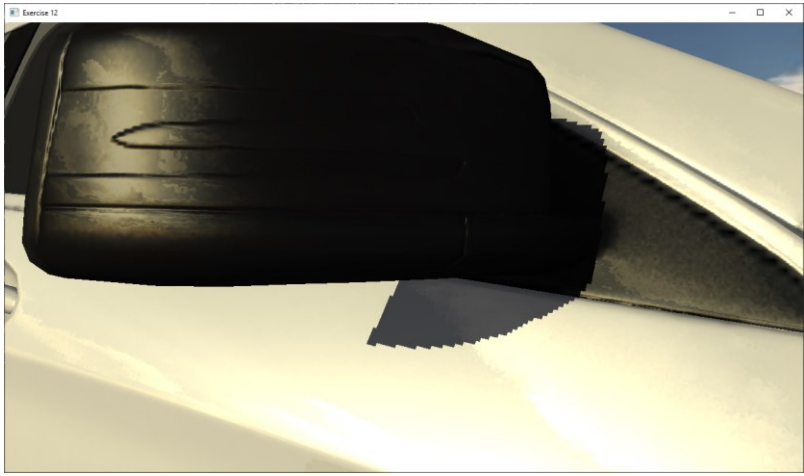
- Solve the acne problem
  - Use the shadowBias variable to offset the value stored in the shadowMap before comparing the distances.
  - Once you get the expected result, play with the bias slider in the GUI to see how the shadows behave.

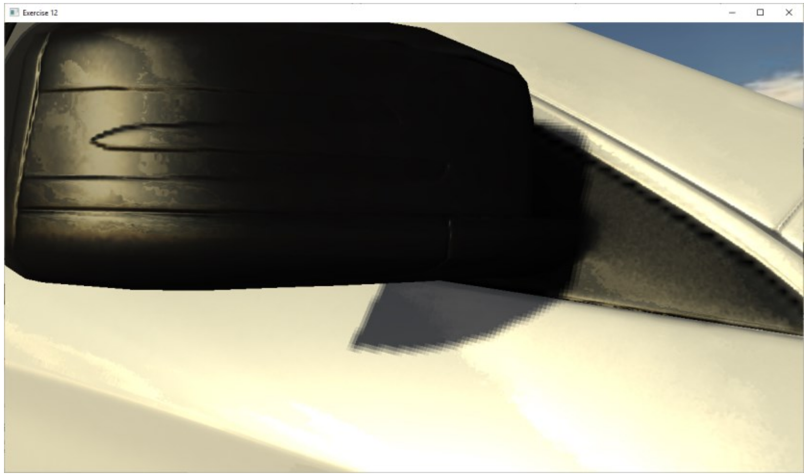




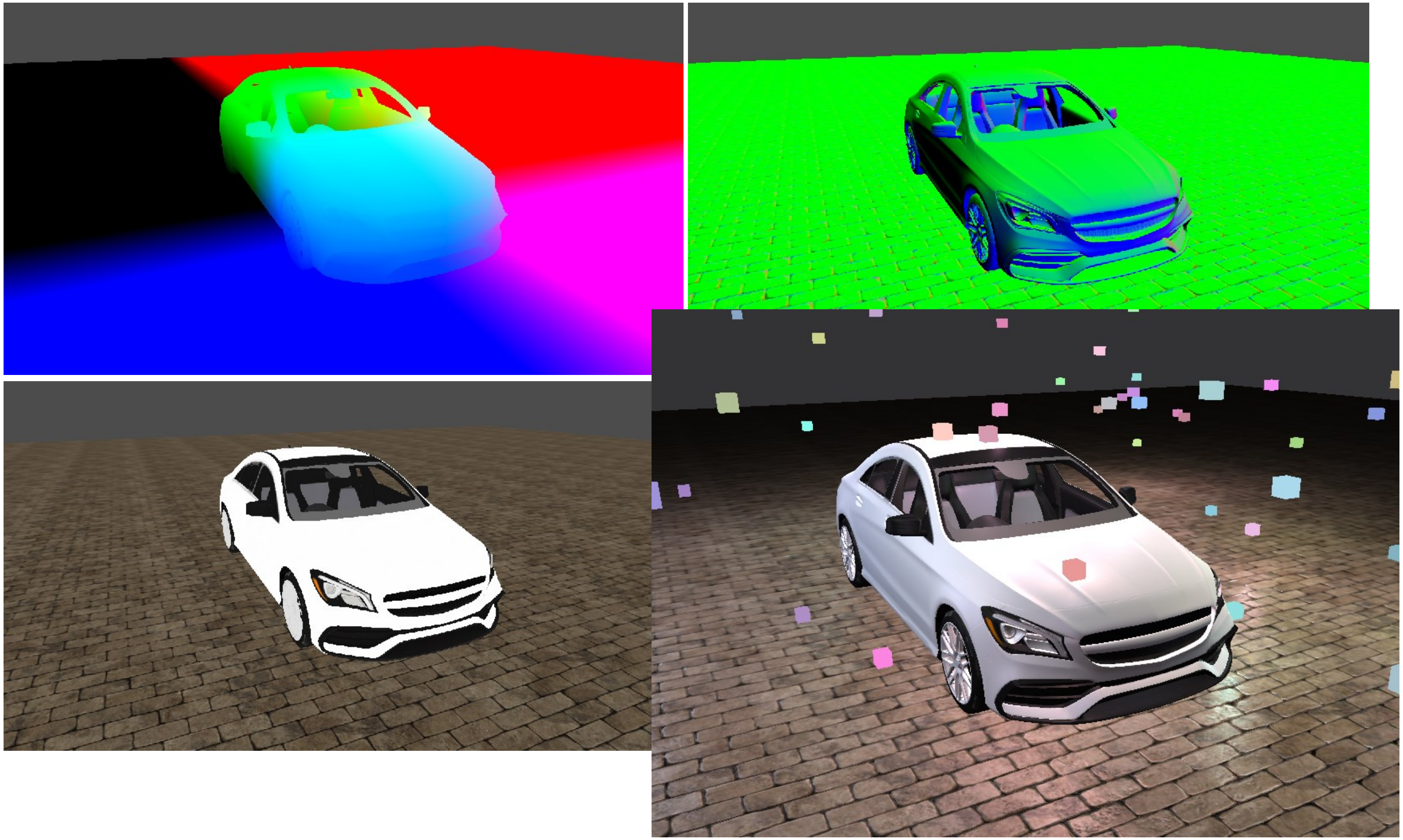
Exercise 12.3 – soft shadows

- Percentage closer filtering
- In the ShadowCalculation function:
  - Sample a 3 x 3 grid of texels (instead of single texel);
  - Set shadow as a weighted contribution of the 9 samples.





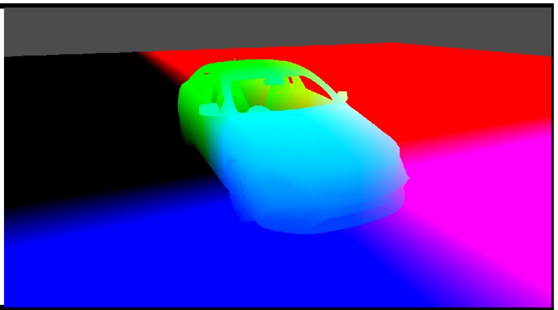
Exercise 12.4-12.6 - G-buffer



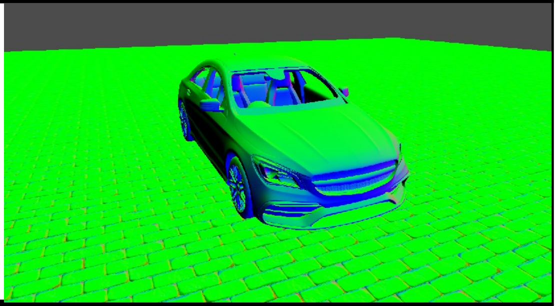
From https://learnopengl.com/Advanced-Lighting/Deferred-Shading

Exercise 12.4-12.6 - G-buffer

- Position
  - 3 channels - rgb/xyz
  - 16 bits per channel
  - World space



- Normals
  - 3 channels - rgb/xyz
  - 16 bits per channel
  - World space



- Diffuse + specular
  - 4 channels – rgba
  - 8 bits per channel
  - rgb = diffuse / a = specular map

G-Buffer in practice

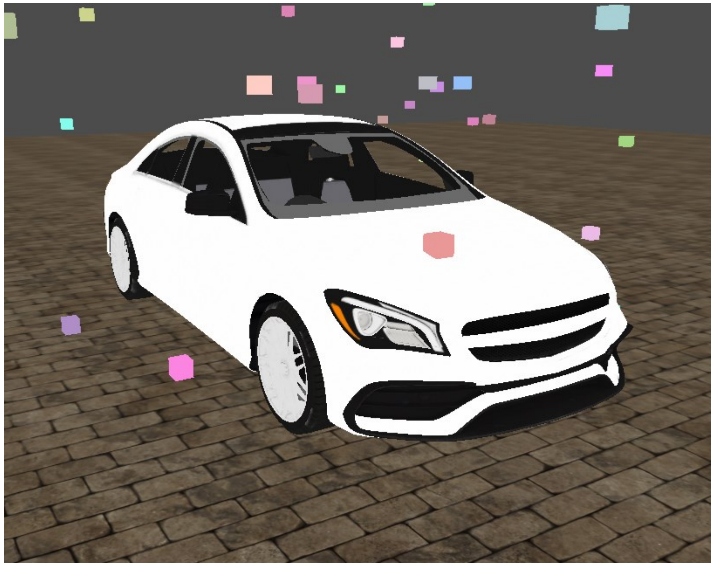
- **No world pos texture**: The world position can be computed using depth and the screen position
- **Depth buffer texture**. Using z-buffer we get the depth for free.
- **Packed normals**. If normal are in camera space, we can store X and Y and find Z in the shading pass (i.e. we assume Z is positive).
  http://aras-p.info/texts/CompactNormalStorage.html
- **Pack remaining data**: Use as little data as possible to store remaining attributes: Albedo-color, specularity-power, specularity-intensity

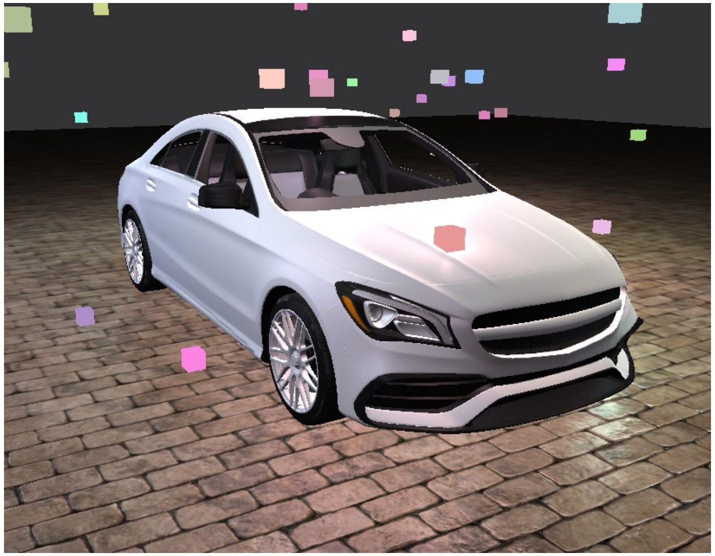
Exercise 12.4 code structure

- Explore code structure
  - The code was copied from https://learnopengl.com/Advanced-Lighting/Deferred-Shading
  - With the addition of forward shading (forward_shading.vert/frag)
  - (optional) Read through the tutorial to get familiar with the deferred shading implementation.
  - Read the render loop code and comments. Notice how, in the deferred rendering option, g_buffer.vert and .frag are used to generate the gbuffer (step 1).
  - Then, the buffers are sent to deferred_shading.vert and .frag and rendered with a quad (step 2)

Exercise 12.5 blinn-phong reflection

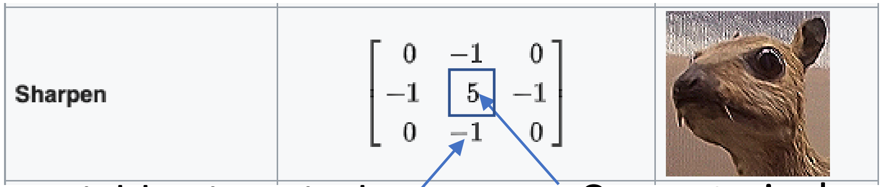
- Implementing the blinn-phong reflection model
  - Implement it in deferred_shading.frag;
  - There is an implementation in forward_shading.frag, use it as a reference;
  - Did you have to do any modification?





Exercise 12.6 post processing 1

- Apply a sharpen filter (convolution kernel) to the diffuse and specular components of the gBuffer
  - Implement it in deferred_shading.frag;
  - Use https://en.wikipedia.org/wiki/Kernel_(image_processing) as a reference;
  - Like in the shadow mapping exercise, you will need to sample the buffer multiple times.



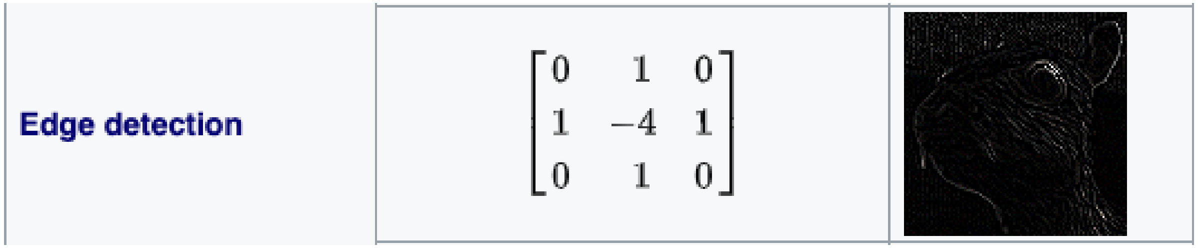
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & \boxed{5} & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

Neighboring pixels

Current pixel

Exercise 12.6 post processing 2

- Apply an **edge detection** filter (convolution kernel) to the **normal** component of the gBuffer
  - Implement it in deferred_shading.frag;
  - Use https://en.wikipedia.org/wiki/Kernel_(image_processing) as a reference;
  - You will use a Laplacian operator to obtain the weighted sum of normal vectors.
  - If the magnitude of the resulting vector is high, it indicates high curvature or surface discontinuity. You want to paint these surfaces black.



Edge detection	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	

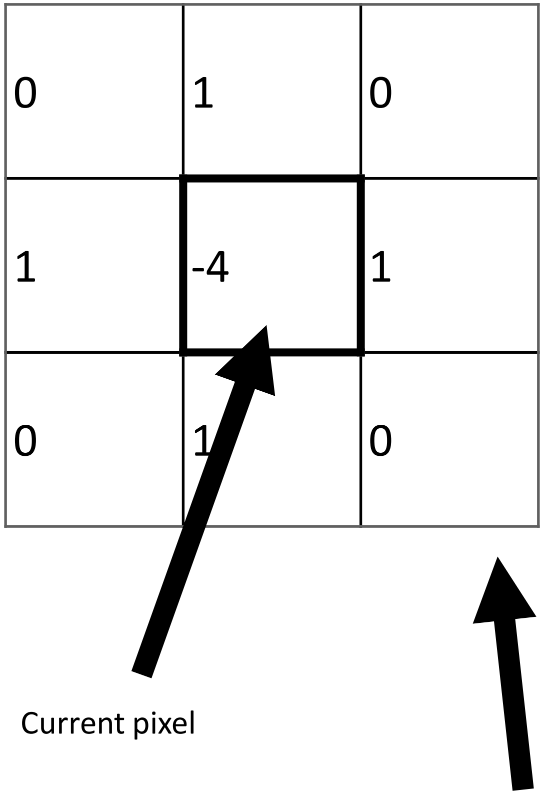
Edge detection

- Second order derivative, a discrete Laplace operator

  https://en.wikipedia.org/wiki/Discrete_Laplace_operator

- Large scalar values indicates edge

- Large magnitude values (for vectors) indicates edge



0	1	0
1	-4	1
0	1	0

Current pixel

Neighbour pixels

Result with post effects