UNIVERSITY OF LEEDS

Report for High-Performance Graphics coursework 2

Jiafei Shi

The University of Leeds

School of Computing

May 2023

Contents

Chapter 1

Prep and Debug

1.1 Shading Space

Here I'm using world space for my shading space, so in the vertex shader I don't need to transform the vertices, normals and textures that are input to the fragment shader.

In the fragment shader, the input vertex coordinates can be calculated directly when performing lighting calculations.

1.2 Descriptor Details

In the current section, only three textures are needed, base color, roughness and metalness, followed by a normal map for the subsequent sections. (Since the alpha texture is the same as base color, it can be reused for base color.)the first is to create an array of descriptors, which is determined by the number of meshes present.

For the different meshes in the model, base color, roughness and metalness have corresponding material ids, however some meshes do not have a normalMapTextureId, i.e. no normal mapping is required, so a default texture needs to be created here, for the purpose of image binding when creating descriptors. Here a default 1 * 1 image is created by rewriting the load_image_texture2d function, where the RGBA of the image is (128, 128, 255, 255).(vkimage.cpp, 93-107)

When creating a descriptor, if the current texture id is encountered as 0xffffffff, then the current descriptor will be bound to the default 1 * 1 image, and for other materials where a texture ID exists, the current descriptor will be bound to the image loaded from the path found in the texture array based on the texture id of the current mesh (main.cpp, 285-326)

For the layout of the current descriptor, the layout needs to have four bindings as each mesh has four different textures to be sampled.

In the RecordCommand function, the input descriptors are bound to the current render pass by calling vkCmdBindDescriptorSets, the corresponding descriptors can be found in the descriptor array according to the material Id of the current mesh.

1.3 Uniform Data

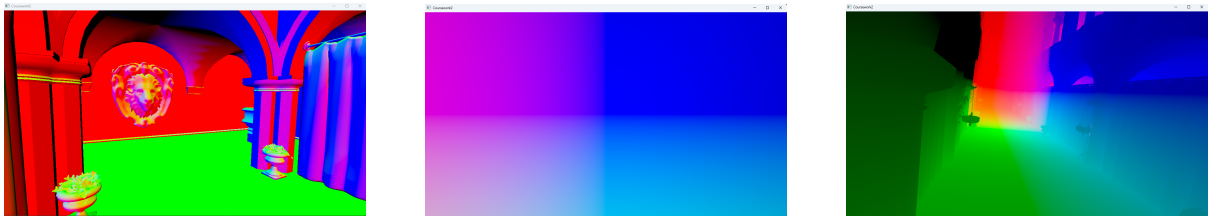
Here I have defined a SceneUniform structure to input the transform matrix, camera position and light source information to the vertex shader and fragment shader, including light source position and light source colour. Since the std140 layout rules need to be met, the camera positions and light sources are defined in vec4.(main.cpp, 83-103)

The next step is to create the SceneUBO and the corresponding descriptor based on the currently defined SceneUniform, which is of type VK_DESCRIPTOR_TYPE_UNIFORM_BUFFER since it is creating uniform data.(main.cpp, 236-265)

In the while loop, the data within the current structure needs to be continuously fetched, so here I have rewritten the update_user_state function to output the calculated camera position in the world coordinate system, and then added the update_light_state function to update the current light source information in real time, the user can stop and play with the P and O buttons on the keyboard, and finally the data is fed into SceneUniform with the update_scene_uniforms function.

In the Record Command function, a buffer_barrier is used to ensure correct synchronisation in the Vulkan pipeline, and then the scene descriptors are bound in the render pass by calling vkCmdBindDescriptorSets.

1.4 Screen Shots



(a) Normal (b) ViewDirection (c) LightDirection

Figure 1.1: Visualization of normal vector, view direction and light direction.

Chapter 2

Lighting

2.1 Screen Shots

Figure 2.1: D

D is the normal distribution function, which describes the distribution of surface normals in a microscopic surface structure.

The roughness of the material affects the distribution of light and dark areas in the scene, with the bright areas representing areas where the surface normals are more aligned with the half-angle vector H. In these areas, light is more likely to be reflected in the direction of the observer.

Dark areas represent areas where the surface normals are less aligned with the half-angle vector H. In these areas, light is less likely to be reflected in the direction of the observer.

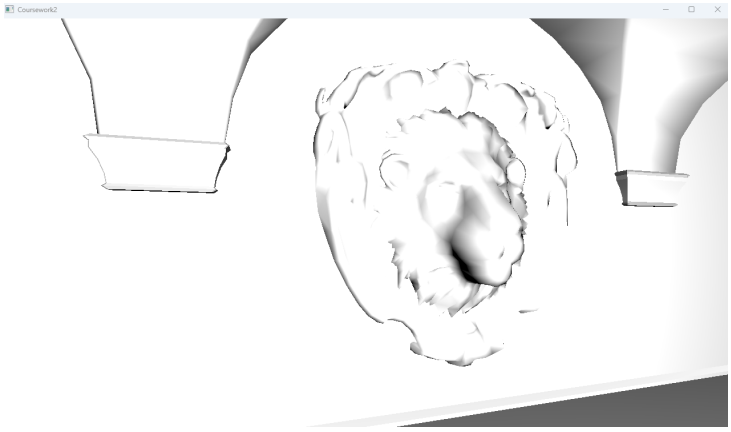


Figure 2.2: G

G is a masking function that primarily reflects the degree to which specular reflections are occluded by the geometry of the surface. Here I will use cook and torrance 1982 function.

When visualising G, a colour close to black indicates a higher degree of occlusion and a colour close to white indicates a lower degree of occlusion.

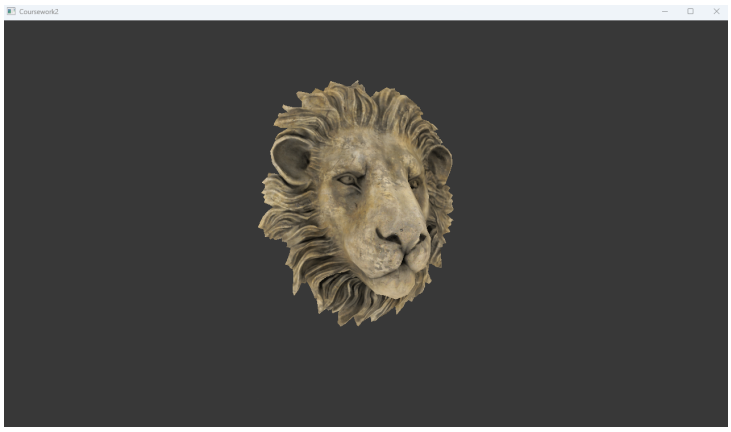


Figure 2.3: F

F is the Fresnel term, which describes the reflectance of light on a surface at different angles of incidence and emanation. When visualised, the lion's surface can be observed to be very realistic and natural.

Figure 2.4: spectular

Specular reflections focus on the glossy highlights of smooth areas of the surface. When the light shines on a smooth wall, it produces a light spot with a high contrast. When the light shines on a rough lion, it gives a softer, blurred appearance.

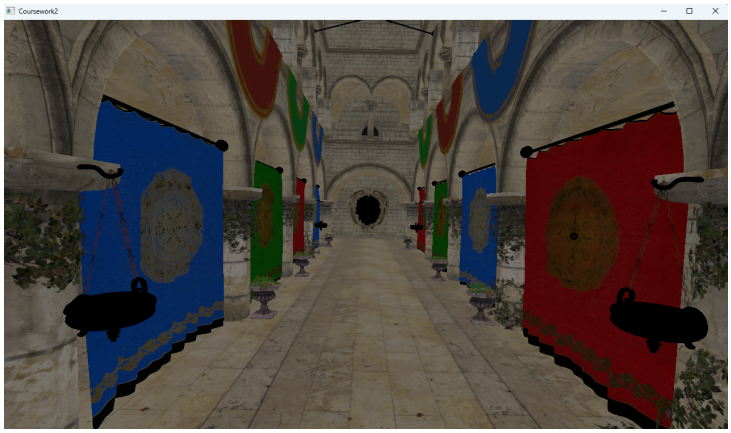


Figure 2.5: diffuse

Diffuse reflection describes the scattering behaviour of light on a surface. In visualisation, diffuse reflection causes objects in a scene to appear uniformly coloured and illuminated.

Chapter 3

Alpha Masking

3.1 Screen Shots

(a) before alpha masking

(b) after alpha masking

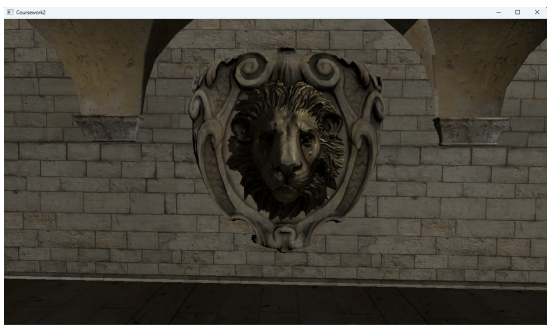
Figure 3.1: Visualization of alpha masking.

Chapter 4

Normal Mapping

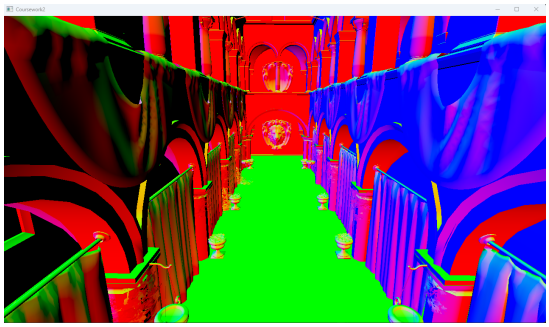
4.1 Screen Shots

(a) before normal mapping

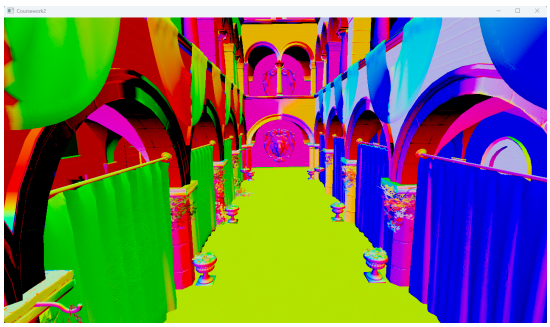


(b) after normal mapping

Figure 4.1: Visualization of lion before and after normal mapping.



(a) before normal mapping



(b) after normal mapping

Figure 4.2: Visualization of normal.

In some cases, it is necessary to retain additional information about the tangent line during the calculation, such as the chi-square coordinate system used for the transformation. The chi-square co-ordinate representation usually uses four components to represent a three-dimensional point, and it is used in graphics mainly to simplify the calculation of transformation and projection matrices.

References

cook, R. and torrance, K. (1982). “A reflectance model for computer graphics”. In: *ACM Transactions on Graphics* 1.1, pp. 7–24.