Otto-von-Guericke University Magdeburg

Faculty of Computer Science

Bachelor Thesis

Applications of Shading and Illumination Algorithms to Vascular 3D Meshes

Author:

Kai Ostendorf

May 1, 2021

Advisors:

Prof. Dr. Bernhard Preim
Department of Simulation and Graphics
Otto-von-Guericke University Magdeburg

Dr. Gabriel Mistelbauer
Department of Simulation and Graphics
Otto-von-Guericke University Magdeburg

Ostendorf, Kai:
Applications of Shading and Illumination Algorithms to Vascular 3D Meshes
Bachelor Thesis, Otto-von-Guericke University Magdeburg, 2021.

Statement of Authorship

I herewith assure that I wrote the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning. I am aware of the fact that violations of copyright can lead to injunctive relief and claims for damages of the author as well as a penalty by the law enforcement agency.

Magdeburg, April 30, 2021 .....

Abstract

This thesis presents a visualization tool for rendering three dimensional vascular surface meshes. A shading pipeline is presented that contains adjustable parameters in each step to influence the final shading result. Emphasis is placed on user interactivity, with a variety of shading algorithms and techniques available that directly affect the resulting renderings starting with the parameters of the lights placed in the scene such as color and intensity. Furthermore, the ambient, diffuse and specular reflectance properties of the vessel materials can be adjusted. The shaders used for the ambient, diffuse and specular terms are separated, allowing the user to individually combine different techniques for different parts of the lighting equation. For the shaders, users can manipulate further parameters, such as the roughness for Cook-Torrance or Oren-Nayar shading. The lit-sphere approach allows users to choose from different materials which are directly applied through lighting, providing a wide range of possible shading techniques without any implementation. Shading is complemented by ambient occlusion to enhance the depth perception. Transparency in combination with the Fresnel effect keeps vessel borders visible, while allowing users to inspect the vessel interior. Finally, a vignette effect and other custom backgrounds can be loaded to enhance the overall presentation of the custom shading created.

Kurzfassung

Diese Arbeit stellt ein Visualisierungstool vor, das zum Rendern von dreidimensionalen Gefäß Meshes dient. Es wird eine Shading Pipeline eingeführt, die in jedem Schritt einstellbare Parameter enthält, welche das Endergebnis beeinflussen. Der Schwerpunkt der Arbeit liegt auf der Interaktivität durch die Nutzer, denen eine Vielzahl von Shading-Algorithmen und Techniken zur Verfügung stehen, um auf die einzelnen Schritte Einfluss zu nehmen beginnend mit den Parameter der in der Szene platzierten Lichter wie Farbe und Intensität. Des Weiteren können die Umgebungs-, Diffus- und Spiegelreflexionseigenschaften der Gefäßmaterialien eingestellt werden. Die einzelnen Komponenten der Lichtberechnung wurden auf einzelne Shader verteilt, so dass die Terme für die Berechnung des Umgebungs-, Diffusen- und Spekularenteils aus unterschiedlichen Shadern miteinander kombiniert werden können. Die Shader können wiederum durch Parameter beeinflusst werden. So kann Cook-Torrance Shading, wie auch Oren-Nayar Shading, durch die Rauheit des Oberflächenmaterials angepasst werden. Lit-sphere shading ermöglicht es dem Nutzer, aus verschiedenen Materialien zu wählen, die über die Position der Lichter platziert werden. Dies ermöglicht es, viele verschiedene Schattierungstechniken zu verwenden, ohne den entsprechenden Shader implementieren zu müssen. Ergänzend zu den Beleuchtungsmodellen kommen Techniken wie Ambient Occlusion und Transparenz zum Einsatz. Ambient Occlusion unterstützt die Tiefenwahrnehmung und hilft, Erhöhungen und Vertiefungen besser zu erkennen. Transparenz wird mit dem Fresnel-Effekt kombiniert. Dieser sorgt dafür, dass Gefäßränder sichtbar bleiben, während Nutzer das Gefäßinnere inspizieren können. Abschließend können ein Vignetteneffekt und benutzerdefinierte Hintergründe genutzt werden, um die Darstellung der erstellten Shadings zu verbessern.

Acknowledgments

We thank Kathrin Bäumler (3D and Quantitative Imaging Laboratory, Department of Radiology, Stanford University) for providing the aortic dissection meshes.

Contents

List of Figures

List of Acronyms

.ppm	portable pixmap
.vtp	visualization toolkit polygonal data
BRDF	bidirectional reflectance distribution function
OIT	order-independent transparency
RGB	red, green and blue color values
RGBA	red, green, blue and alpha color values
SSAO	screen space ambient occlusion
TBN	tangent, bitangent, normal matrix
VAO	vertex array object

1. Introduction

The visualization of three dimensional polygonal data plays an important role in clinical praxis and the analysis of clinical data. A vast amount of ways exist to render and visualize three dimensional meshes with different shading algorithms and techniques. This thesis aims at providing a visualization tool containing a balanced selection of various shading algorithms and techniques, ranging from stylistic to realistic rendering approaches to interact and analyze the polygonal data chosen by the users. The tool is supposed to enable users to render and present three dimensional meshes in multiple different ways, exploring the mesh and gather information. It mostly aims at the visualization of three dimensional aortic dissection meshes, which are the main focus of this thesis. To visualize this data, different rendering techniques are provided. Especially, depth and transparency are important cues for the understanding of anatomical and topological relationships accustomed to this data and therefore screen space ambient occlusion (SSAO) and order-independent transparency (OIT) are implemented in a customizable way.

Goal of this Thesis

The goal of this thesis is to present a software environment which allows users to display and visualize three dimensional vascular surface meshes. Different means of visualization were implemented to offer multiple ways to interact and work with their data. Offering users the possibility to customize the visualization approach combining diffuse and specular terms of various shading approaches and enabling them to use occlusion shadows and transparency to support the visualization are the main goal of this thesis.

The users are provided with a variety of customizable shading an reflectance models that can be combined with occlusion shadows and transparency. The implemented physically-based models are: Blinn-Phong Shading [36], the Oren-Nayar reflectance model [33] and the Cook-Torrance reflectance model [8]. Furthermore, stylistic rendering approaches in the form of toon shading [48] and a more advanced stylistic shading technique in the form of lit-sphere shading [41] were implemented. Lit-sphere shading applies light in form of textures, sampled from different spheres,

to meshes. Extending lit-sphere shading is the lit-sphere extension [45] that uses the light position in contrast to the view direction to obtain dynamic diffuse and specular shading. To supply users with different predefined textures, we added a material gallery.

More advanced techniques, adding occlusion shadows and transparency, were implemented in the form of SSAO [13] and Weighted Blended OIT [30]. Ambient occlusion enhances the depth perception, while transparency ensures that no important information is lost due to occlusion. In combination with transparency, we use the Fresnel effect to highlight vessel boundaries [5]. The parameters of these techniques can be adjusted in real time to individualize the resulting visualization.

Users can interact (rotation, translation, zoom) with any visualization. To improve the orientation in space and the positioning of additional lights users are provided with a view-cube [20] that represents the rendered mesh and additional light sources. To provide the users with means to highlight generated visualizations, a vignette effect and the possibility to load custom background images are offered. The implemented shaders and additional techniques enable users to visualize and present important aspects of their data sets.

Structure of this Thesis

This thesis is structured in the following way: First, state-of-the-art is described in chapter 2. This includes an overview of shading algorithms in general, containing both stylistic and realistic rendering. Furthermore, recent papers in medical visualizations employing indirect volume visualization techniques are presented.

Next, the implementation of the illumination and reflectance models is explained in chapter 3, starting with an introduction of the OpenGL rendering pipeline and continuing on to the implemented shading techniques. Here, Blinn-Phong Shading (see section 3.2) is used to introduce the different vectors and lighting terms used to calculate the incident light. Next, toon shading (see section 3.3) is presented providing a stylistic rendering approach. Afterwards, two lighting models based on micro facets are introduced in the form of Oren-Nayar shading (see section 3.4) for the diffuse term and Cook-Torrance shading (see section 3.5) for the specular term. The next discussed technique is lit-sphere shading (see section 3.6), which provides another approach for stylistic rendering. A screen space technique to calculate ambient occlusion is presented in section 3.7 and the implementation of OIT is explained in section 3.8. The methodology section concludes with a discussion of how the different implemented techniques are connected. For every technique, we discuss the theory of the original papers first. Then we present our implementation and if necessary, additions and omissions in the implementation compared to the originals are discussed.

In chapter 4, Results and Discussion, renderings combining multiple techniques are shown. The limitations of the visualization tool are explained and the different steps to create a visualization are shown. This thesis is closed in chapter 5, Conclusion and Future Work, with a short summary and promising future avenues of the visualization tool.

2. State-of-the-art

This chapter provides an overview of approaches developed in recent years for techniques used in this thesis. First, an introduction to general shading algorithms and techniques used in the visualization tool is given to provide a basic understanding of possible shading techniques. In the second part, medical visualization tools and their implementation utilizing different kinds of the introduced techniques are presented.

The shading techniques used in this thesis can be roughly split into three categories:

1. Physically-based bidirectional reflectance distribution functions (BRDF) that are based on an approximation of real light behaviour on surfaces and surroundings.
2. Stylistic shading approaches used to abstract or simplify parts of the geometry to emphasize different attributes of the rendered mesh.
3. Techniques that focus on occlusion and transparency.

Every BRDF needs lights to model its diffuse and specular term. Berbaum et al. [6] find that the accurate perception of a surface depends on the positioning of the light, with lights opposite of the camera increasing perception by a large margin. Kahrs et al. [19] go into the advantages of the different types of light: Ambient light defines the base brightness of shadow areas. A key light is used to define the brightness, shape and texture of the rendered object. Fill lights add additional diffuse lighting and can be used to control the contrast of the scene. Back lights are used to separate objects from the background and are usually placed to the direct opposite of the camera behind the object. Lee et al. [25] introduce light collages with geometry-dependent lights, which allow lights to be placed automatically inside a scene. This is done by segmenting the rendered object, based on curvature, highlighting convex regions and darkening concave regions. Additional steps are silhouette enhancement, added through a fall-off formula that depends on the view and normal vectors. Furthermore, the addition of proximity shadows added by

computing depth discontinuity curves, which are generated by comparing a pixels depth to the depth of its neighbors. For every curve a shadow light direction is created.

Recent approaches in physically-based rendering have been made using neural networks. Liu et al. [26] use a physically-based rendering network to obtain material properties, surface normals and illumination values from an input image. The network then applies a material to the input image using the surface normals and illumination values from the input image.

Toon or cel shading can be used to guide a viewers focus. Context is shaded in lower detail using toon or cel shading, while the important aspects of the data are shaded in a more complex way [37]. Traditional toon shading uses a 1D texture which maps tone to a surface orientation relative to a light source. This idea is extended on by Barla et al. [1], which replaces the 1D texture with a 2D texture, which adds a haze effect to distant objects [24]. The 2D texture is made up of a horizontal axis, which corresponds to tone, and a vertical axis, which corresponds to tone detail. An attribute has to be chosen by the developer that controls the tone detail. This can be, for example, depth or orientation. The approach of Barla and colleagues [1] was extended by Hao et al. [18]. They introduce a view-dependent attribute that is based on the orientation and curvature of the surface with respect to the observer. They use a 2D texture, where the x-component corresponds to curvature and the y-component determines shading. Van Pelt et al. [48] use cel shading and occluding contours in their 4D MRI blood flow visualization. Cel shading based on Gooch and Gooch [16] was used to simplify the surrounding vessel structure, while retaining a cue for visual depth. Furthermore, superimposed occluding contours based on the approach of DeCarlo and Rusinkiewicz [10] were employed to highlight the outline of the anatomical structure.

Ambient occlusion is a shading technique to calculate indirect lighting and therefore darkening points which are less exposed to light. This enhances depth perception [22] and helps discovering features such as holes and creases. It was first developed by the research and development department of Crytek in their CryEngine 2 and was first used in the game Crysis. Groß and Gumhold [17] employed ambient occlusion and transparency in their rendering approach of line data. Ambient occlusion is also used in direct volume rendering approaches: Ropinski et al. [38] realized dynamic ambient occlusion and Dìaz et al. [11] achieved real-time ambient occlusion. The SSAO calculation used in this thesis is based on an improved version of Cryteks approach, but since then multiple new algorithms have been created: Mantiuk [28] presented Gaze-Dependent SSAO shortening rendering time using the tracked eye position on the screen. The scene is then only rendered in maximum graphics quality around the regions the viewer is looking at and gradually reducing outwards. Park and Baek [34] developed Outline-dependent SSAO that calculates occlusion in places where it is likely to occur. This is done by detecting dense places where meshes are close to each other. The outline of the mesh is rendered and if multiple outlines are drawn on a single pixel the intersection point is stored in a stencil mask. The generated mask can then be used to compute SSAO by processing only the regions in the mask. Zhang et al. [51] generated efficient SSAO using a deep network. They built a data set containing deferred shading buffer data and ground-

truth ambient occlusion shaded images. Creating a deep neural network based on the network structure of U-Net [50], they design a Compute Shader Library containing six parameterized compute shaders to compute the shaded ambient occlusion images. To increase the accuracy of SSAO Vermeer et al. [49] use a stochastic depth map for the calculation of their Stochastic-Depth Ambient Occlusion. Their approach takes geometry into account that is not visible during rendering and is therefore not used in the computation of the ambient occlusion factor. The algorithm captures multiple scene layers per pixel and selects random samples from these different layers, rather than using only samples from the first layer as in horizon-based ambient occlusion [4].

To blend several transparent objects in OpenGL they have to be sorted according to depth. To do this manually is a cumbersome task and impossible in scenes with hundreds of moving objects. A solution to this problem is OIT. The implementation of OIT in this thesis is based on the approach of McGuire and Bavoil [30], who alter the composition operator to make it order independent. This is done to avoid the cost of sorting primitives according to depth. Everitt [14] uses depth peeling to extract each unique depth in a scene into layers, which are then depth-sorted and composited. Bavoil and Myers [3] extend depth peeling with dual depth peeling based on a min-max depth buffer, which peels one layer from the front and one layer from the back at the same time using different blend equations in the alpha blending for the front and back peeling. In addition, they introduce a sort-independent method, replacing the RGBA color in the alpha blending equation with the per-pixel weighted average over the pixel. Barta and Kovács [2] propose the usage of per-pixel linked lists. When transparent objects are processed by the fragment shader, it fills a structure with the properties of the processed fragment. The properties are: surface radiance, volumetric attenuation, albedo, distance from the camera and orientation of the fragment. The fragments are then sorted in ascending order of the distance from the camera and blended accordingly. Maule et al. [29] present an OIT approach using a dynamic fragment buffer. The buffer enables them to allocate memory in the exact amount needed for multiple per-pixel layers in each frame. The implementation runs at a frame rate high enough for the approach to be suitable for real time interaction. Münstermann et al. [32] calculate transparency based on moments. They use the logarithm of the transmittance enabling them to accumulate the depth-dependent function per pixel additively instead of multiplicatively. Using an additive rendering pass for all transparent surfaces yields moments. These are used to approximate the original function through moment-based reconstruction algorithms which are used in a second pass for the composition.

Gasteiger et al. [15] use different visualization techniques to visualize cerebral aneurysms with embedded blood flow information. Surface meshes are extracted from segmented clinical image data. Based on the segmentation, the surface morphology of the aneurysm is reconstructed. This mesh is then used to obtain a computational grid, on which they perform a computational fluid dynamics simulation to acquire blood flow data. An adapted surface visualization is needed to decrease the occlusion of the blood flow information caused by the enclosing surface. Therefore, they defined the following requirements for their solution: Visibility of internal flow information, depicting aneurysm features, surface shape, spatial relationships and

depth cues. They combine ghosted views, line rendering, flow visualization, shadows and atmospheric attenuation. Ghosted views are based on the Fresnel-reflection model [39], but in their implementation reflection is replaced by opacity to achieve more opacity at regions facing away from the viewer and more transparency at regions facing the viewer. Line rendering in the form of contours and morphological features is used to convey shape. For their application they implemented the approach of McGuire and Hughes [31]. Streamlines, glyphs and stream surfaces were used to visualize blood flow. To increase depth perception, they applied the method of Luft et al. [27] with a modified computation of the depth buffer values to achieve a constant depth cue appearance. Atmospheric attenuation is used as a second depth cue. It is introduced by applying fog to the previous visual representations, which makes the objects fade into the distance. This supports the viewer in tracing the flow and focusing on close visual information.

Díaz-Gracía and Vázquez [12] visualize human brain fibers with ambient occlusion and halos. The soft shadows created by ambient occlusion allow spatial features such as the location and relationship between grouped adjacent fibres to be more clearly visualised. The halos surrounding the fibers help to improve the understanding of the shape of the model. The ambient occlusion factor is calculated by comparing the depth difference between the target pixel and all its neighbors. The halo factor is calculated in the same way, just using a different depth difference function. In addition, a distance factor is calculated that darkens fragments that lie deeper in the scene. The final color is composed using the three factors and Phong shading [36].

Lawonn et al. [23] improve the ghosted view approach by Gasteiger et al. [15]. They modify the vessel opacity of the front faces using the suggestive contour measure. Additionally, depth blurring is used instead of atmospheric attenuation for a more natural depth perception. The resulting visualization overcomes the limitations imposed by the Fresnel effect, which renders regions facing away from the viewer opaque. Furthermore, they add visual effects in form of cast shadows to improve the understanding of the spatial relationship between overlapping vessel sections.

Behrendt et al. [5] use Phong shading [36], OIT and the Fresnel effect [39] to visualize vessels in their blood flow visualization approach. Phong shading is used to convey the vessel shape. Behrendt and colleagues render the surface semi-transparent to avoid occluding the flow visualization, where the amount of transparency can be controlled by the user. The transparency is implemented in the form of OIT following the approach of Thibieroz [44], which ensures correct image composition regardless of overlaying transparent fragments. To resolve the arising problem of less visible lighting effects on transparent surfaces they use their “glass lighting mode”, which adds a Fresnel effect to the lighting. This approach is similar to the before mentioned ghosted views of Gasteiger et al. [15]. Therefore, regions with strong lighting appear more opaque, which highlights the vessel shape and achieves an effect which is similar to looking through a glass bottle.

3. Methodology

The Visician framework is used as a base to implement the various shading algorithms. The meshes used are generated from visualization toolkit polygonal data (.vtp). We used aortic dissection meshes consisting of two flow channels and a flap. First the .vtp data is loaded into the framework and a three dimensional mesh is generated. The mesh can be freely rotated and translated in the work space. Depending on the shader used, up to two light sources are active at the same time, which can be rotated around the mesh by the user. Based on these prerequisites, all of the shaders and techniques were implemented in OpenGL and GLSL.

3.1 OpenGL Rendering Pipeline

Three dimensional meshes consist of polygons made up of points in space called vertices. Each of these vertices has a three dimensional coordinate and a normal calculated by averaging the surface normals of the faces containing that vertex. These attributes are stored as an ordered list in a vertex array object (VAO). This is the first step of the rendering pipeline called vertex specification. The VAO is further processed in the pipeline, which consists of the steps shown in Fig. 3.1.

The vertex shader processes each vertex and calculates the final position of the vertex in the scene. Tessellation is an optional step in the pipeline where patches of vertex data are divided into smaller primitives, which results in a smoother mesh. The geometry shader computes zero or more output primitives from each incoming primitive. In addition, the geometry shader can tessellate primitives or convert them to other types. Vertex post-processing includes primitive assembly, clipping and face culling. In primitive assembly, primitives are created based on the vertex data and the primitive type defined by the user. This can be either a point, line or triangle. Primitives that lie outside the viewing volume are discarded, which is called clipping. Culling describes the process of not rendering triangles that are occluded by other geometry. Rasterization converts a primitive into a sequence of fragments. Rasterization results in at least one fragment being generated for every pixel covered by the primitive on the screen. The fragment shader processes the fragments created

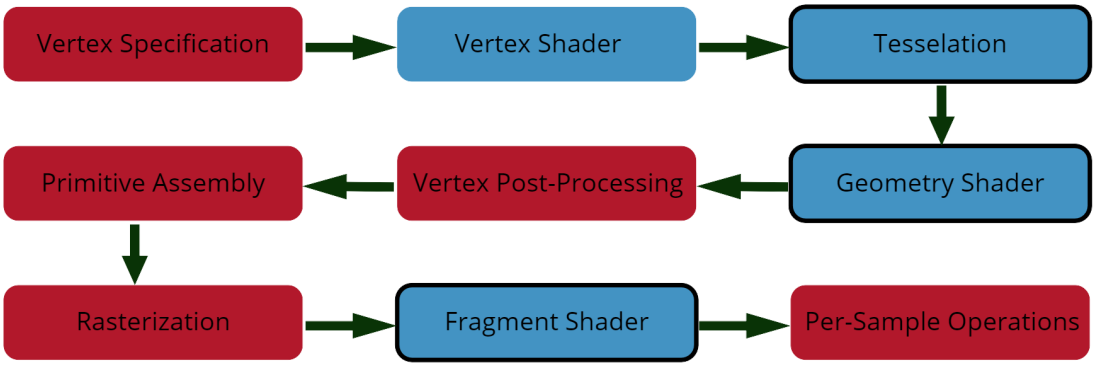


Figure 3.1: Sequence of steps in the OpenGL rendering pipeline. The pipeline consists of the steps OpenGL performs to render objects. The steps are vertex specification, consisting of the vertex shader, tessellation and the geometry shader followed by vertex post-processing, primitive assembly and rasterisation. After that, fragment shaders and per-sample operations are used. The steps marked in blue are programmable. The black outlines mark optional steps.

in the rasterization step and calculates the final color of each fragment. The final step in the pipeline are per-sample operations, which include the culling test, scissor test, stencil test and depth test. This thesis mainly focuses on fragment shaders to implement the different shading techniques.

Shaders are programmable stages of the OpenGL rendering pipeline that run on a graphics processor. A vertex shader is used to process individual vertices provided by a VAO. An input vertex consists of different attributes that define, among other things, the location and orientation of the vertex. For each input vertex, an output vertex is generated.

Unless otherwise stated, it can be assumed that all of the vertex shaders implemented in this work receive the same inputs. These are given in the form of a VAO containing a three dimensional vector for each vertex coordinate and a three dimensional vector for each vertex normal. In addition, a structure is set for the camera. This structure contains the required matrices (model matrix, view matrix, projection matrix, normal matrix) to perform the various spatial transformations for further processing of the vertices. The vertex shader then calculates three outputs: The position of the output vertex in clip space, the position of the vertex in world space and the normal of the vertex. The position of the corresponding output vertex in clip space is calculated by transforming the input coordinates from local space to clip space using the model-view-projection matrix.

Depending on the implementation, either the position of the vertex in world space or in eye space is used to calculate lighting in the fragment shader. It is calculated by transforming the input coordinates of a vertex with the model matrix. To calculate the output normal of a vertex, the input normal is transformed with the normal matrix. The world space position and the normal are passed on as inputs to the fragment shader. In addition, the fragment shader receives different inputs used for the lighting calculation.

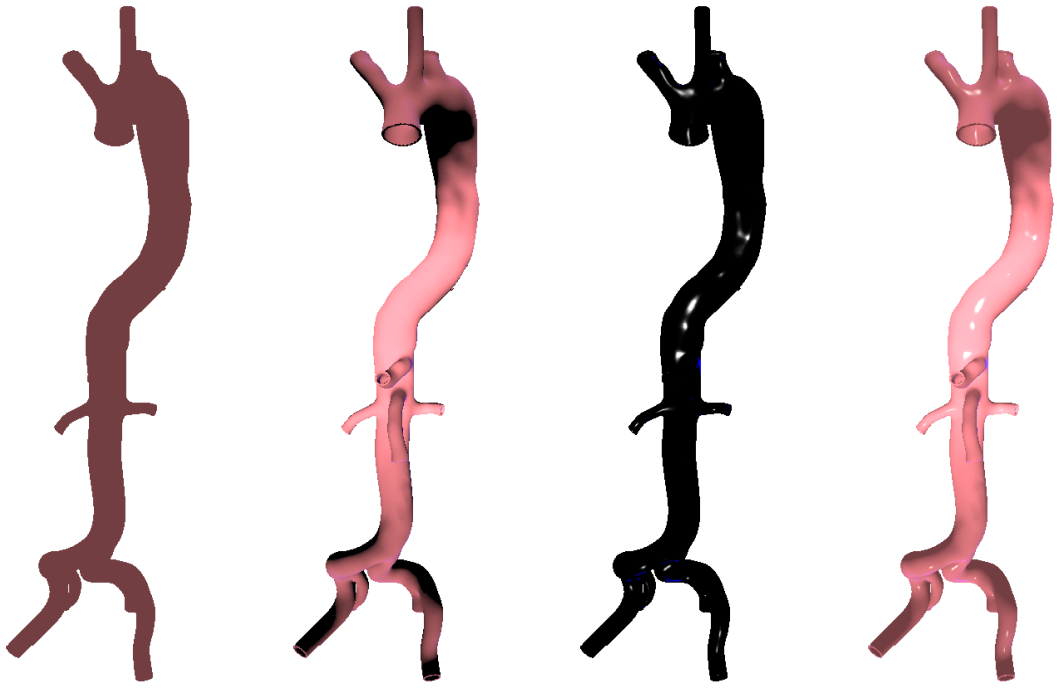


Figure 3.2: Contribution of the individual lighting terms to the final composition: (a) shows the ambient term, (b) the diffuse term and (c) the specular term. (d) shows the combined color values.

Fragment shaders compute the final color, based on the implemented lighting model. Lighting models can employ a single fragment shader or use multiple fragment shaders in combination to add effects such as SSAO or OIT. The inputs can vary, but all of the implemented fragment shaders receive as inputs the camera, material layout and light layout. The camera is the same as the vertex shader camera. The material layout contains the colors of the different light types and a shininess factor, this is explained in more detail in section 3.2. The light layout contains the position of the lights, color and intensity.

3.2 Blinn-Phong Shading

Phong lighting [36] consists of three different types of light, which contribute to the final color of the fragment: ambient, diffuse and specular lighting. The contribution of each lighting term is shown in Fig. 3.2

The ambient term is a fixed value p_a which describes the color and intensity of light falling uniformly on the surface from all directions. The diffuse term follows Lambert's cosine law, which states that the radiance of ideally diffusing surfaces is directly proportional to the cosine of the angle between the surface normal N and the light direction vector L. The amount of diffuse reflection d is calculated by:

$$d = \max(0, N \cdot L). \tag{3.1}$$

The specular term is based on the law of reflection, which states that the exit angle of a reflected light beam is equal to the entry angle of the incoming light beam. In

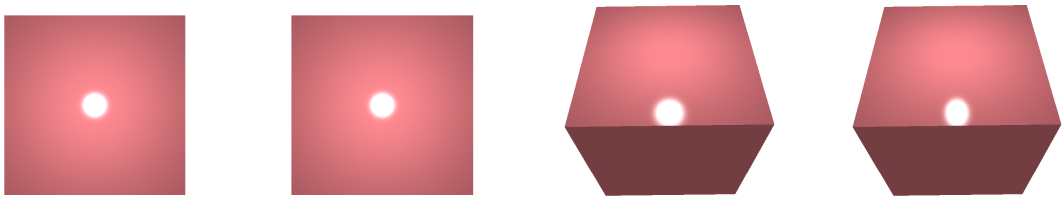


Figure 3.3: Comparison of specular highlights viewed from high and low viewing angles. Phong's approach (a), (c) produces round highlight shapes when viewed from any angle. Blinn's approach (b), (d) produces elliptical shapes at low angles relative to the surface.

case of the surface being a mirror this leads to a highlight, if the angle between the viewing direction V and the reflected light direction vector R is smaller than a certain threshold value. The threshold is defined by using a shininess factor c as a power to the dot product of V and R. The amount of specular reflection s is calculated by:

$$s = \max(0, R \cdot V)^c. \tag{3.2}$$

The calculation of specular reflections was improved by Blinn [7], who showed that instead of using the dot product of V and R, one can calculate a halfway vector H that lies midway between the viewing direction and the light direction to calculate the amount of specular reflection, where:

$$H = \frac{L + V}{\|L + V\|}. \tag{3.3}$$

To calculate the specular reflection with the halfway vector, the dot product of H and N is calculated and taken to the power of c:

$$s = \max(0, H \cdot V)^c. \tag{3.4}$$

This changes the specular highlight shape, especially at small viewing angles with respect to the surface. Conventional Phong highlights always maintain a round shape, whereas Blinn highlights become elliptical and thus more realistic at smaller viewing angles, as shown in Fig. 3.3. The vectors for calculating Phong and Blinn-Phong shading are shown in Fig. 3.4.

To calculate the perceived intensity i and thus the final color of a fragment, all three types of light are added together. The amount of diffuse reflection d is multiplied by the proportion of diffuse reflection p_d and the amount of specular reflection s is multiplied by the proportion of specular reflection p_s:

$$i = p_a + dp_d + sp_s. \tag{3.5}$$

The implementation additionally takes into account the light properties. A lighting factor l_f is calculated, which depends on the color of the light l_c, the intensity of the light l_i and the distance l_d of the light from the object:

$$l_f = l_c \frac{l_i}{l_d}. \tag{3.6}$$

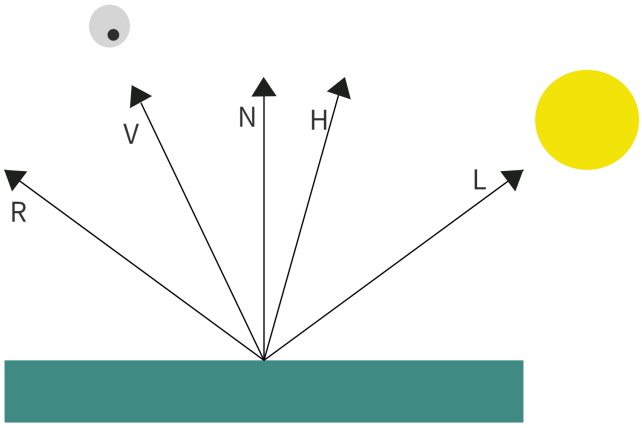
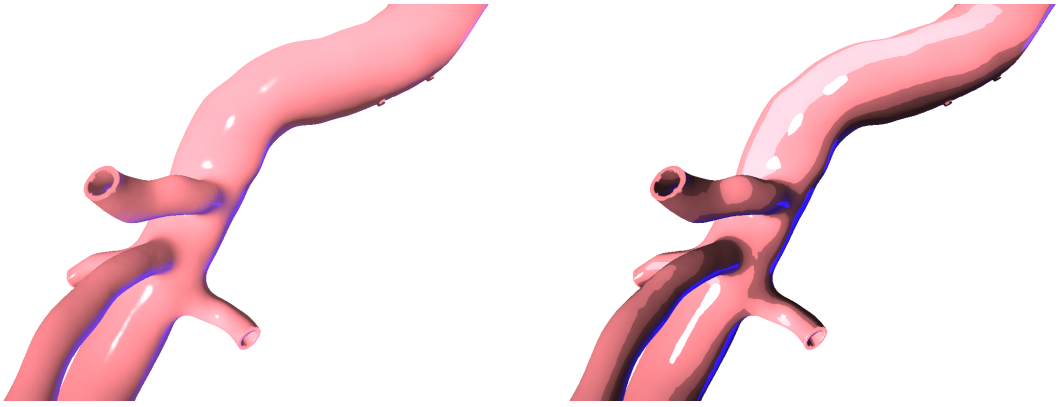


Figure 3.4: Vectors used to calculate Phong and Blinn-Phong shading. L: Vector from the surface the light source. V: Vector from the surface to the viewer. R: Vector of perfectly reflected light from L. N: Normal vector of the surface. H: Half direction vector.



(a) **(b)**

Figure 3.5: Comparison between Blinn-Phong shading (a) and toon shading (b). Blinn-Phong shading produces smooth transitions between color values, while toon shading abruptly changes colors.

This factor only affects the diffuse and specular term of equation 3.5:

$$i = p_a + dp_d l_f + sp_s l_f. \tag{3.7}$$

3.3 Toon Shading

Toon shading is a stylistic rendering approach used to emulate the look of cartoon art. The shading helps to simplify parts of a visualization while maintaining a cue for depth and still outlining the morphological structure [48]. Like Blinn-Phong shading, toon shading consists of three lighting terms. These are calculated with the same formulas as for Blinn-Phong shading, with the main difference that a fixed value t_f (toon factor) is set for specific ranges of p_d. This leads to abrupt transitions between colors in the final shading compared to Blinn-Phong, which can be seen in Fig. 3.5.

The diffuse term d is calculated using equation 3.1. Depending on the calculated dot product, t_f is set in the following way:

$$t_f = \begin{cases} 0.8, & \text{if } d \geq 0.95, \\ 0.5, & \text{if } d \geq 0.5, \\ 0.25, & \text{if } d \geq 0.25, \\ 0.1, & \text{otherwise.} \end{cases} \tag{3.8}$$

The specular term s is calculated using equation 3.4. If the angle between H and V exceeds a certain threshold, s is set to the maximum value:

$$s = \begin{cases} 1.0, & \text{if } \max(0, H \cdot V) > 0.99, \\ 0.0, & \text{otherwise.} \end{cases} \tag{3.9}$$

The final color is calculated by:

$$i = p_a + dp_d l_f t_f + sp_s l_f, \tag{3.10}$$

using equation 3.7 and adding the toon factor to the diffuse lighting calculation.

3.4 Oren-Nayar Shading

The diffuse term of the lighting calculation in section 3.2 is based on the Lambertian reflection model [21]. The brightness of Lambertian surfaces is independent of the viewing direction, which can be seen in the calculation of the diffuse term in equation 3.1. An emerging problem with this reflection model is that for several real-world objects, the calculated reflection is not sufficient [33], since the reflection of light from rough, non-reflective surfaces is more evenly distributed than from shiny or glossy surfaces. The reflected light in the Lambertian diffuse calculation falls off too quickly near the edges to imitate these rough surfaces. A solution to this problem is presented by Oren and Nayar [33].

They use the concept of micro facets to approximate reflected light on rough surfaces, which was earlier used by Torrance and Sparrow [46] to calculate specular reflections from rough surfaces. The roughness is modelled in form of V-cavities at the microscopic level, as shown in Fig. 3.6a. Each facet of each V-cavity has perfect Lambertian reflection when the facet corresponds to a pixel on screen, but when multiple facets of multiple cavities are combined, this is no longer true for the corresponding pixel. Oren and Nayar assume that each facet of a V-cavity da is small compared to the area dA corresponding to one pixel. Additionally, da is large compared to the wavelength of the incoming light λ. θ_a denotes the slope of the V-cavity facet depending on the global surface normal $\hat{n}$ in this point and the facet normal $\hat{a}$. This is shown in Fig. 3.6b and is summarized in the following equation:

$$\lambda^2 << da << dA. \tag{3.11}$$

Furthermore, the brightness of a pixel changes depending on the angle between the viewer and the light source. If the angle is small, the viewer sees the brighter of the

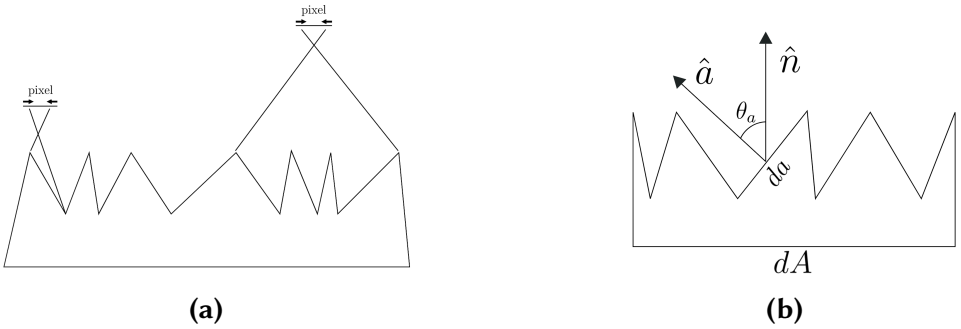


Figure 3.6: (a) shows the difference between one and multiple V-cavities covered by one pixel. If one facet of a V-cavity corresponds to one pixel on screen, the reflection can be assumed to be perfect lambertian. If factets of multiple V-cavities contribute to one pixel on screen, the contributions of each facet to the lighting have to be approximated. Therefore the contributions of facets on different magnification levels change. (b) shows the V-cavities model: da is the area of a facet of the cavity. dA is the area of the surface corresponding to one pixel. $\hat{a}$ is the normal of the facet. $\hat{n}$ is the global surface normal in this point. θ_a is the angle between $\hat{a}$ and $\hat{n}$. Adapted with permission from Springer Nature: Springer Science and Business Media LLC, M. Oren and S. Nayar, Generalization of the Lambertian Model and Implications for Machine Vision, *International Journal of Computer Vision*, 14(3):227-251, Copyright 1995 by Springer Nature.

two surfaces of the V-cavity, whereas if the angle is large, the viewer sees more of the darker surface. This is shown in Fig. 3.7. The distribution of surface facets and their respective slopes is assumed to be Gaussian with mean μ and standard deviation σ.

The original proposal calculated light rays bouncing between adjacent facets of the V-cavities, called interreflections, to calculate the total radiance of the surface. This is computationally very expensive, so they proposed a qualitative model that ignores these interreflections. The qualitative model calculates the amount of diffuse reflection d in the following way:

$$dp_d = \frac{p}{\pi} E_0 \cos\theta_i (A + B \max(0, \cos\phi_r - \phi_i) \sin\alpha \tan\beta), \tag{3.12}$$

where p describes the fraction of light reflected by the surface. E_0 is the irradiance of the facet if the direction of the incoming light vector equals the direction of the facet normal (i.e. at maximum incidence) and θ_i is the angle between the incoming light L and the surface normal $\hat{n}$. ϕ_r is the angle between the view direction V and $\hat{n}$, ϕ_i the angle between L and $\hat{n}$. α is the maximum of the two angles between V and $\hat{n}$ and L and $\hat{n}$. β is the minimum of the two angles between V and $\hat{n}$ and L and $\hat{n}$. A and B were numerical evaluated by varying surface roughness, the angles of incidence and reflection and set in the following way:

$$A = 1.0 - 0.5 \frac{\sigma^2}{\sigma^2 + 0.33}, \tag{3.13}$$

$$B = 0.45 \frac{\sigma^2}{\sigma^2 + 0.09}, \tag{3.14}$$

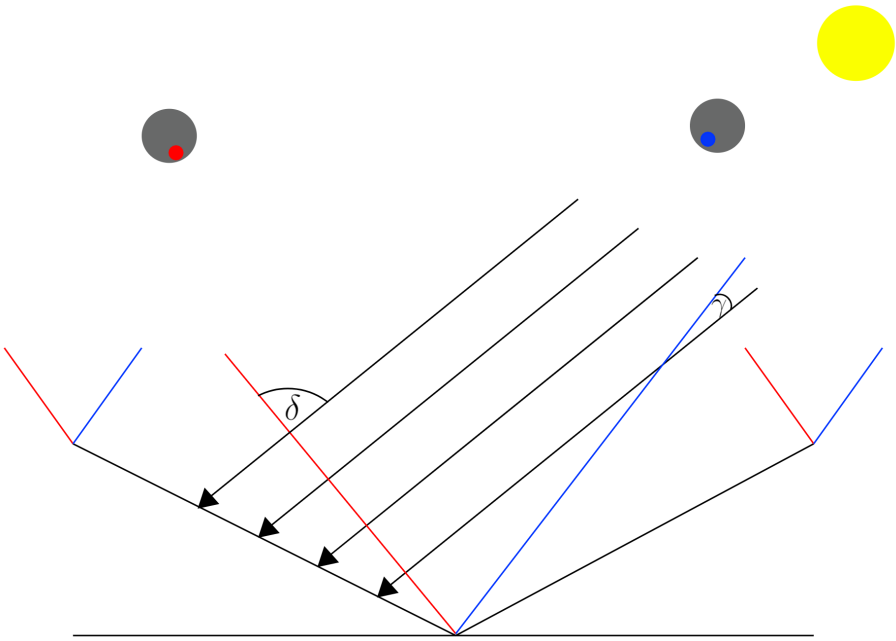


Figure 3.7: Explanation of the brightness perceived by a viewer when viewing a V-cavity from different angles. The angle δ between the light source and Viewer 1 (red) is large and he or she therefore sees more of the darker part of the cavity, while the angle γ between viewer 2 (blue) and the light source is small and he or she sees more of the bright part of the cavity.

where σ is the surface roughness given by the Gaussian distribution. In the final code $\frac{p}{\pi}E_0$ equates to p_d, which is given by the diffuse part of the material structure. $\cos\theta_i$ equates to $\max(N \cdot L)$. One can observe that if $\sigma = 0$, the calculation of the diffuse term in equation 3.12 resembles default Lambertian diffuse reflection, as seen in the implementation of the diffuse term in section 3.2 in equation 3.1. Results of different roughness values can be seen in Fig. 3.8.

3.5 Cook-Torrance Shading

Cook and Torrance [8] use a different approach to calculate the specular term of the lighting equation. They propose a model to approximate the behaviour of specular lighting depending on the roughness of a surface. They describe that the distribution of reflected light in the specular term is dependent on micro facets as described in section 3.4, using the approach of Torrance [46].

Cook and Torrance assume that only facets (see Fig. 3.6b) whose normal $\hat{a}$ points in the direction of H (see equation 3.3) contribute to the specular term of the reflection from L to V (see equation 3.4). Therefore the specular term is:

$$s = \frac{F}{\pi}\frac{DG}{(N \cdot L)(N \cdot V)}, \tag{3.15}$$

where F is the Fresnel term [43] describing how light is reflected from each facet. D describes the fraction of facets that are oriented in the direction of H, called the normal distribution function. G is the geometrical attenuation factor used to

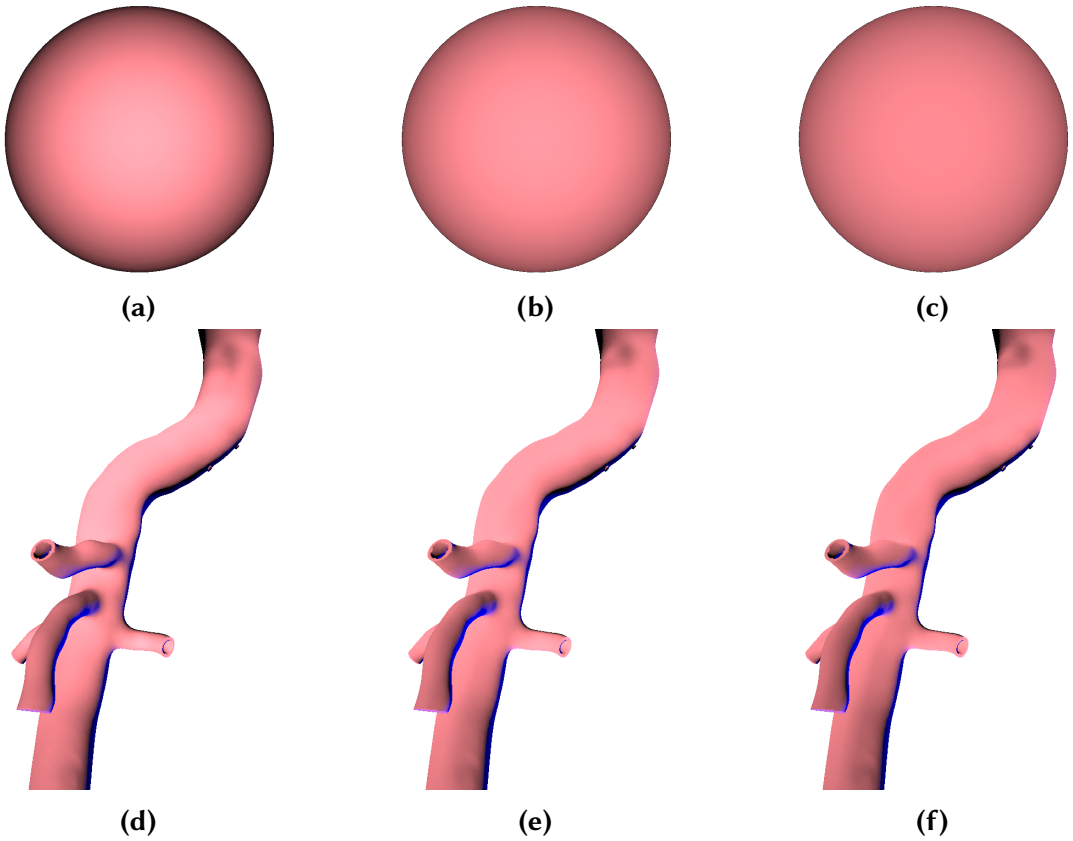


Figure 3.8: Sphere and vessel rendered with different roughness settings for the Oren-Nayar shading: (a), roughness = 0.0, (b), roughness = 0.5, (c), roughness = 1.0. The ambient and specular term have been set to zero.

describe the shadowing and masking of facets among each other. We approximate F using Schlick's [40] approach:

$$F(\theta) = f + (1 - f)(1 - \cos\theta)^5, \tag{3.16}$$

where θ is the angle between L and N and f is the reflection coefficient when $N \cdot V = 0$. f is calculated from the incident of refraction of the material the light is travelling through n_1 and the incident of refraction of the object n_2:

$$f = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2. \tag{3.17}$$

We calculate D using the normal distribution function of Trowbridge and Reitz [47]:

$$D = \frac{\sigma^2}{\pi((N \cdot H)^2(\sigma^2 - 1) + 1)^2}, \tag{3.18}$$

where σ is a factor describing the roughness of the surface. G is calculated using the approach of Smith [42] taking view direction and light direction into account:

$$G = G_1(L)G_1(V). \tag{3.19}$$

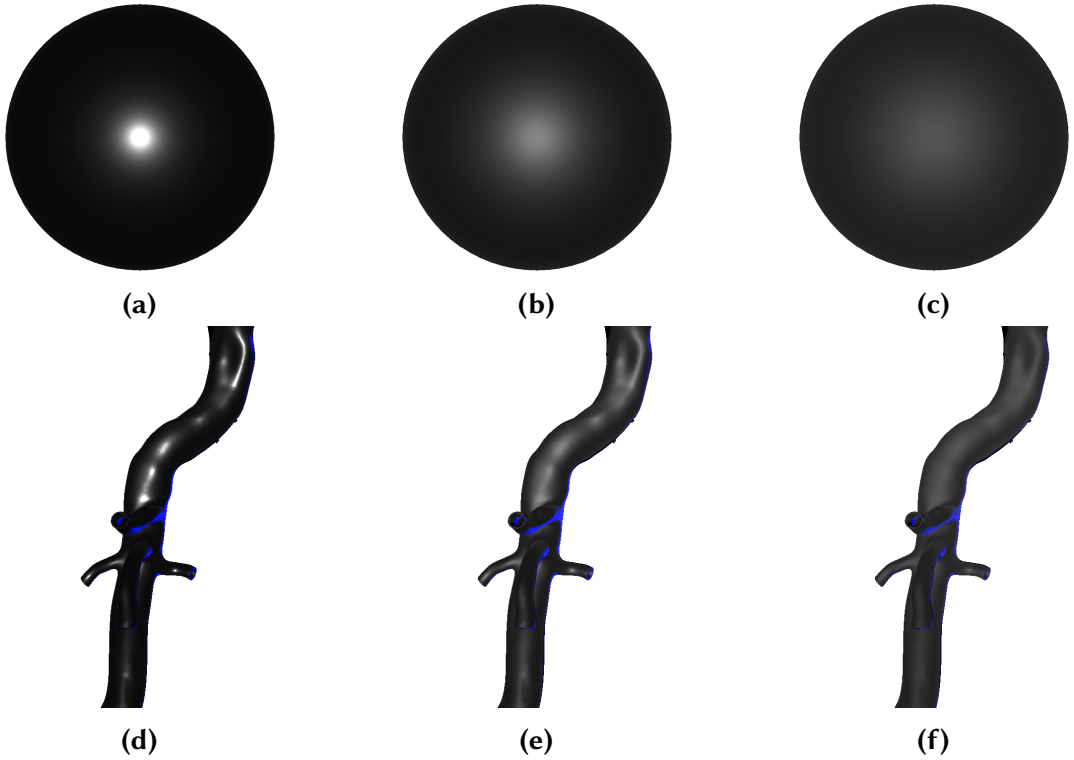


Figure 3.9: Sphere and vessel rendered with different roughness settings for the Cook-Torrance shading: (a), roughness = 0.1, (b), roughness = 0.3, (c), roughness = 0.5. The ambient and diffuse term have been set to zero.

Using Smith's approach G_1 is for the light direction calculated as:

$$G_1(L) = \frac{2(N \cdot L)}{(N \cdot L) + \sqrt{\sigma^2 + (1 - \sigma^2)(N \cdot L)^2}}, \tag{3.20}$$

and for the view direction respectively:

$$G_1(V) = \frac{2(N \cdot V)}{(N \cdot V) + \sqrt{\sigma^2 + (1 - \sigma^2)(N \cdot V)^2}}. \tag{3.21}$$

For different roughness values the results in Fig. 3.9 were obtained.

3.6 Lit-Sphere Shading

Lit-sphere shading is, like toon shading (see section 3.3), a stylistic rendering approach used to enable users to select a texture and apply the texture to shade the mesh depending on the camera view and surface normal. It was introduced by Sloan et al. [41] trying to utilize existing shading studies and extract shadings from pictures to create an image of a sphere which can be used to shade a model.

We use an improved version of this approach presented by Todo et al. [45] to calculate the ambient and diffuse term and provide users with different textures of spheres in a material gallery that can be selected and used for the shading. To calculate the final color values, the texture is used as a lookup and depending on the

view space normal N_V the coordinates of the according color are obtained. N_V is described by Sloan in the following way:

$$N_V = (N_{V_x}, N_{V_y}, N_{V_z}), \tag{3.22}$$

where:

$$N_{V_x} = N \cdot V_x, \tag{3.23}$$

$$N_{V_y} = N \cdot V_y, \tag{3.24}$$

$$N_{V_z} = N \cdot V_z. \tag{3.25}$$

V_x, V_y and V_z are the three dimensional view plane vectors and N is the normal vector. Assuming that the coordinate in the middle of the texture represents the reflection of light, when $N \cdot V = 1.0$, falling off to the edges to $N \cdot V = 0.0$, the texture coordinates (u, v) are obtained from a texture in OpenGL in the following way:

$$(u, v) = (0.5 \cdot N_{V_x} + 0.5, 0.5 \cdot N_{V_y} + 0.5). \tag{3.26}$$

If $N_{V_z} < 0$, the color is set to a fixed value. The calculation of this shading only depends on the view and normal vectors and can therefore not be used for shadings employing dynamic lighting.

We use a front and a back light, which can both be freely rotated around the mesh by the users. Because of this, the proposal of Sloan is not suitable, but Todo et al. [45] developed an extension for lit-sphere shading that uses dynamic lighting, introducing a new space normal representation in form of light space normals.

They define the light space for a given light direction L using three orthogonal three dimensional vectors L, L_x and L_y. A lightspace normal N_L is defined in the following way:

$$N_L = (N_{L_x}, N_{L_y}, N_{L_z}), \tag{3.27}$$

where:

$$N_{L_x} = N \cdot L_x, \tag{3.28}$$

$$N_{L_y} = N \cdot L_y, \tag{3.29}$$

$$N_{L_z} = N \cdot L_z. \tag{3.30}$$

The texture coordinates are obtained in the shader by:

$$(u, v) = (r \cos\theta + 0.5, r \sin\theta + 0.5), \tag{3.31}$$

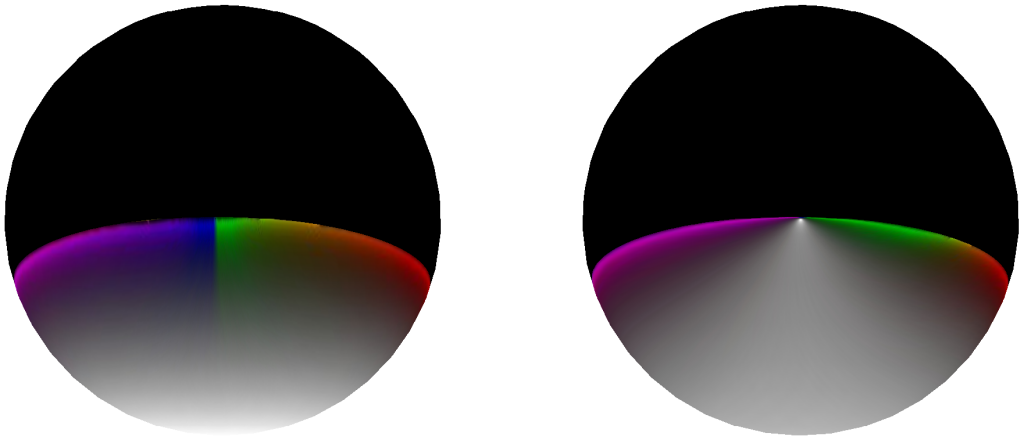
where r is calculated by:

$$r = \frac{\arccos(N_{L_z})}{\pi}, \tag{3.32}$$

and θ is given by:

$$\theta = \arctan\left(\frac{N_{L_Y}}{N_{L_x}}\right). \tag{3.33}$$

To create the light view, one can either manually define it using the static tangent space or use the provided method by Todo et al. [45]. They state that the manually created light view produces distorted highlights at the singular point, therefore we employ the recommended method to implement the light view. This is done by calculating the angle between the camera view V and the light L and then rotating the camera view to the light view. The difference can be seen in Fig. 3.10.



(a) (b)

Figure 3.10: Difference between tangent space and the transformation of camera view to light view. In (a) the sphere is rendered using the tangent space for the light view creating distortions at the singular point. In (b) the sphere is rendered using the transformation of the camera view to light view creating correct highlights.

3.7 Screen Space Ambient Occlusion

SSAO is used to approximate shadows that occur when the scattering of ambient light is blocked by surrounding geometry. These shadows are part of the ambient term, where p_a is multiplied by the ambient occlusion factor a_o that determines how much of the ambient light is prevented form hitting the fragment that is rendered. SSAO can be added to the aforementioned shading algorithms by modifying the ambient term with the calculated occlusion factor. Equation 3.7 is then modified in the following way:

$$i = p_a a_o + dp_d l_f + sp_s l_f. \tag{3.34}$$

Ambient occlusion helps with depth perception of the rendered scene and is therefore a useful addition to grasp the geometry of objects.

To calculate the ambient occlusion factor, the approach of Vries [9] was used. Based on Cryteks implementation in the CryEngine 2, he describes the method for the implementation in OpenGL. The base idea is to calculate the ambient occlusion factor for each fragment by sampling depth values around the according fragment. The sampling is done by creating a sampling kernel in form of a hemisphere around the fragment the a_o is supposed to be calculated for and sampling the depth of the surrounding geometry. Samples that have a higher depth (farther away from the camera) than the fragment the a_o is being calculated for are inside of the geometry and contribute towards the occlusion factor. This is shown in 2D in Fig. 3.11, where the grey circles show samples inside of the geometry and the white circles show samples outside of the geometry. To make geometry that is close to the fragment the a_o is being calculated for have a bigger impact than geometry that is farther away, the sample kernel is filled with the help of a linear interpolation function. This leads to a high density of sample points in close proximity of the fragment with the number of samples decreasing the farther away they get. A problem that can occur, when using the same sample kernel for every fragment, are banding artifacts.

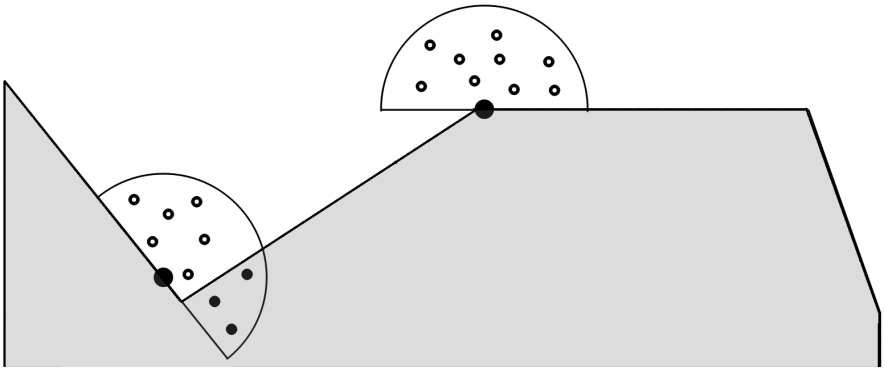


Figure 3.11: 2D presentation of the sample kernel in form of a hemisphere. The bigger black circle is the fragment the ambient occlusion factor is being calculated for. The white dots with black border are samples taken outside of the geometry not contributing to the occlusion factor and the black dots are inside of the geometry and therefore contribute to the occlusion factor.

These artifacts can be circumvented by using multiple randomly generated sample kernels for each individual fragment, but this approach would be to computationally intensive and therefore Vries suggests to only use one sample kernel and rotate it for every fragment. This introduces enough randomness to avoid banding effects and still keep a high performance. This improvement introduces another problem in form of noise, which can be avoided by blurring the noisy image.

In contrast to former shaders used in this thesis, SSAO is not implemented in a single vertex and fragment shader. First, a geometry pass is done, where the first vertex shader transforms the fragment positions and normals to view space. In the first fragment shader the sample kernel is used to calculate the a_o. The kernel is rotated with the help of a noise texture and transformed from tangent space to view space using a tangent, bitangent, normal matrix (TBN). With the sample points transformed to view space, the ambient occlusion factor is calculated by comparing the fragments depth to all of the sample points depth. If the sample depth is greater than the fragment depth, the sample contributes to the occlusion factor. Lastly, the occlusion factor is divided by the number of samples taken to normalize it.

To further tweak the approach, Vries suggests to use a bias which is added to the sample positions depth during the depth comparison. Testing showed that this can change the resulting effect tremendously and therefore the bias has been implemented as a user controllable parameter. Additional changeable variables the radius of the sample kernel, which determines how much of the surrounding geometry affects the occlusion factor and the strength of the occlusion by raising the a_o to a user specified power. In the next step, the occlusion factor texture is passed on to the second fragment shader that blurs the texture and outputs a texture containing the blurred a_o. For SSAO, we had to update shaders with the ability to pass a texture to them containing the occlusion value for every fragment. The p_a then only has to be multiplied by the a_o as seen in equation 3.34. The entire process of combining the different textures for the final visualization can be seen in Fig. 3.12

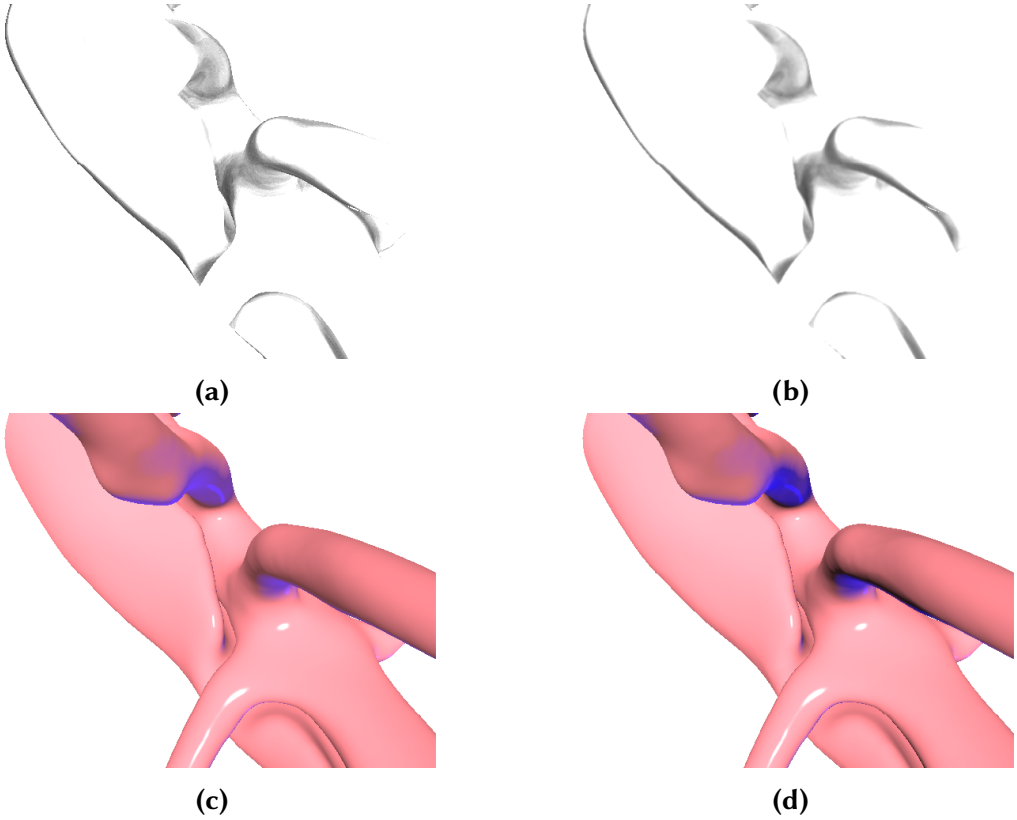
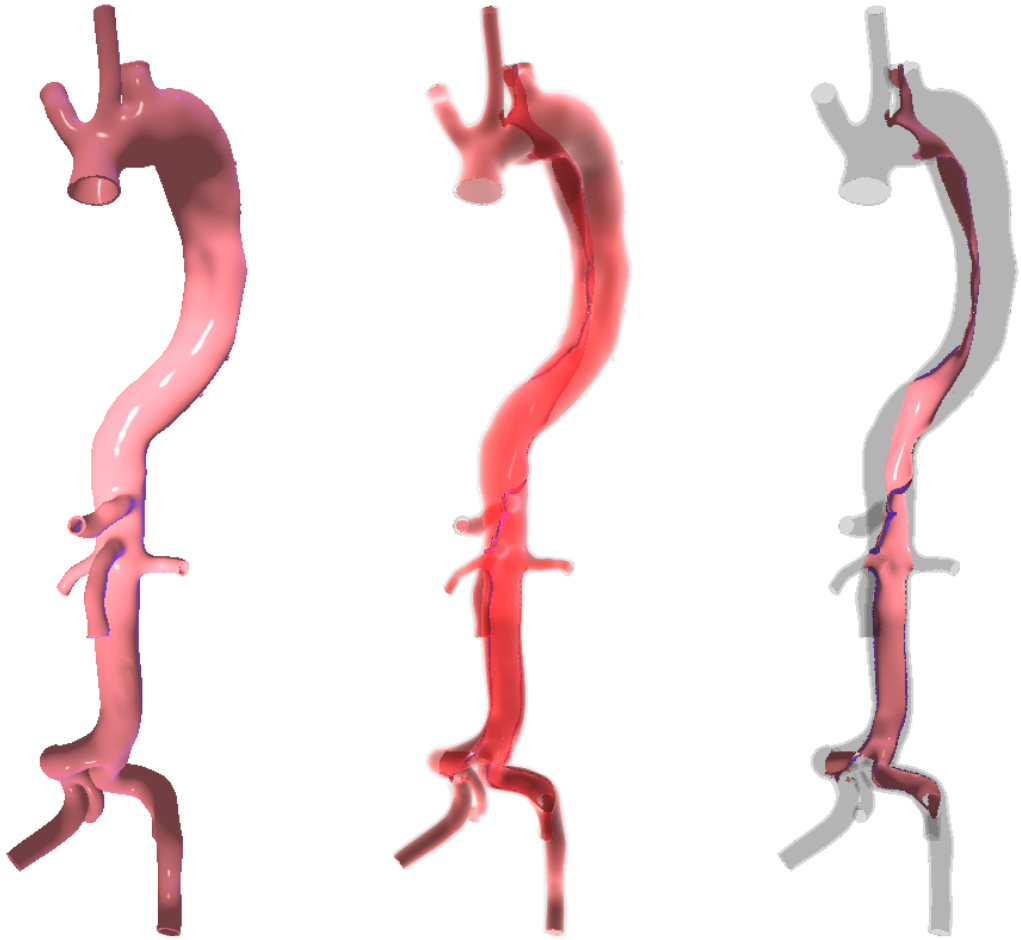


Figure 3.12: Process of applying screen space ambient occlusion. First the occlusion factor is calculated using the sample kernel to sample depth values around the regarding fragment. The random rotation of the kernel leads to noise (a) which is blurred in the next step (b). (c) shows the shading of the mesh to which the occlusion factor of (b) will get added using 3.34 resulting in the final shading (d).

3.8 Order-Independent Transparency

Rendering certain objects transparent inside of a visualization is a key requirement for visualizing vascular meshes and to achieve deeper understanding of the geometry in the scene. Important information can be partly or completely obstructed by opaque structures making it impossible to obtain the information of the occluded geometry without removing it entirely. This could for example be fluid inside of a vessel or information visualizing the flow or other properties. To keep the context of the surrounding vessel and simultaneously obtain the information inside of the vessel, the outer layer has to be rendered transparently. This is shown in Fig. 3.13. If the vessel was not rendered transparently the flap would be obstructed entirely. Although OpenGL provides the possibility of blending, depending on alpha values that determine how transparent an object is, it is not as simple as drawing the fragments with their respective alpha values, because the correct color is dependent on the order of the fragments. To solve this problem, fragments can be sorted according to their depth and then drawn and blended in the correct order using per-pixel linked lists [2].

Another approach is to weigh the color of the fragments according to the fragments depth in the scene as presented by McGuire and Bavoil [30]. The final color C_f of a



(a) (b) (c)

Figure 3.13: An opaque rendering of the outer vessel wall (a) would obstruct the view of the fluid and the flap. In (b), the vessel wall and fluid are rendered transparently so that the user can examine the flap. In (c), the fluid is rendered completely transparent, with only the outer vessel wall as context for examining the flap.

background surface C_0 is based on the surfaces in front of C_0 and the scalar coverage α of those surfaces:

$$C_f = \frac{\sum_{i=1}^{n} C_i}{\sum_{i=1}^{n} \alpha_i} \left(1 - \prod_{i=1}^{n}(1-\alpha_i)\right) + C_0 \prod_{i=1}^{n}(1-\alpha_i). \tag{3.35}$$

This operator now exactly computes coverage of the background and therefore improves the color term. McGuire and Bavoil explain that in equation 3.35, the surfaces with the highest coverage influence the combined color of the partial coverage surfaces the most, regardless of depth ordering. To fix this problem, they improved their original operator from equation 3.35 in the following way:

$$C_f = \frac{\sum_{i=1}^{n} C_i \cdot w(z_i, \alpha_i)}{\sum_{i=1}^{n} \alpha_i \cdot w(z_i, \alpha_i)} \left(1 - \prod_{i=1}^{n}(1-\alpha_i)\right) + C_0 \prod_{i=1}^{n}(1-\alpha_i). \tag{3.36}$$

This operator adds weights to the colors decreasing with higher depth of the blended fragment, where z is zero at the center of projection that decreases with higher

distance from the camera to $-\infty$. The weight function $w(z_i, \alpha_i)$ has to be chosen so that it gives surfaces that are close to the camera higher weights. McGuire and Bavoil present different weight functions. Testing showed that for our use case the depth complexity and the number of blended layers were too low to detect differences between the provided weight functions. In addition, we could not find any differences in execution speed when using different weighting functions, therefore we decided to use:

$$w(z, \alpha) = \alpha \cdot \max\left[10^{-2}, \min\left[3 \times 10^3, \frac{10}{10^{-5} + (|z|/5)^2 + (|z|/200)^6}\right]\right]. \quad (3.37)$$

We implement OIT as follows: First, opaque objects are rendered in an opaque pass with a chosen lighting shader creating a texture containing the respective color values C_0. Next, the transparent objects are rendered with the same shader in a transparency pass that creates a texture containing the color and transparency values. To determine the transparency of a fragment, the shaders were updated with the ability to calculate view dependent transparency based on the Fresnel effect where:

$$\alpha = 1 - \max(0, N \cdot V). \quad (3.38)$$

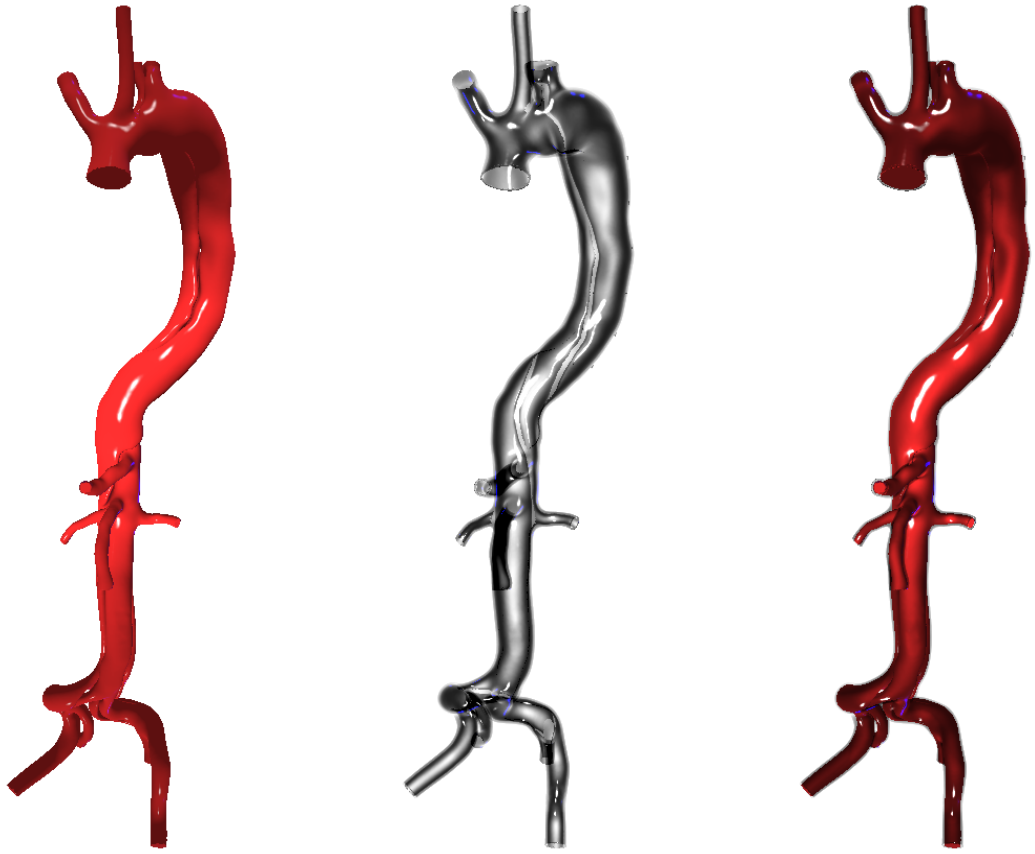
This leads to high transparency of the fragment, when the viewer is looking directly at it falling off, the lower the angle the between the view and normal vector becomes. After the additional transparency values were calculated, the shader calculates the impact each fragment has on the final color C_f using equation 3.37. In the last step, the opaque and transparent textures are blended depending on the alpha values of the transparent texture using equation 3.36. The different steps are shown in Fig. 3.14.

3.9 Shader Structure

To combine the different shading techniques, especially the combination of different lighting terms, multiple shaders are used using the output of the previous shaders as inputs. The shaders are split to avoid redundancy in the code, otherwise every lighting shader would have to include the code for every other lighting shader to calculate the respective diffuse and specular terms.

At first the geometry is processed and saved into a fragment buffer object containing textures for every vertex attribute needed for the calculation of the various techniques. Those textures are passed to the different shaders in the pipeline to calculate the different lighting terms. Furthermore, each shader adds its lighting calculations to the final color texture which is drawn over a screen filling quad in the last step.

In case of SSAO, the respective shaders explained in section 3.7 are used to calculate the occlusion factor, which is simply multiplied to the ambient shader output. OIT effectively doubles the number of shaders, because each lighting term has to be calculated for the occlusion and transparency pass separately. The blending of both textures results in another texture containing the final shading which is drawn on screen. The different modifiable parameters each step contains can be seen in Fig. 3.15.



(a) (b) (c)

Figure 3.14: Different visualizations of a human aorta with its branches using transparency: First the opaque objects are rendered (a) using one of the lighting shaders in this case the fluid inside of a vessel. Next the transparent objects are rendered (b) using a lighting shader extended with equation 3.38 and equation 3.37. Here, the outer layer of the vessel is rendered. In the final step both textures are blended using equation 3.36, resulting in the final shading (c).

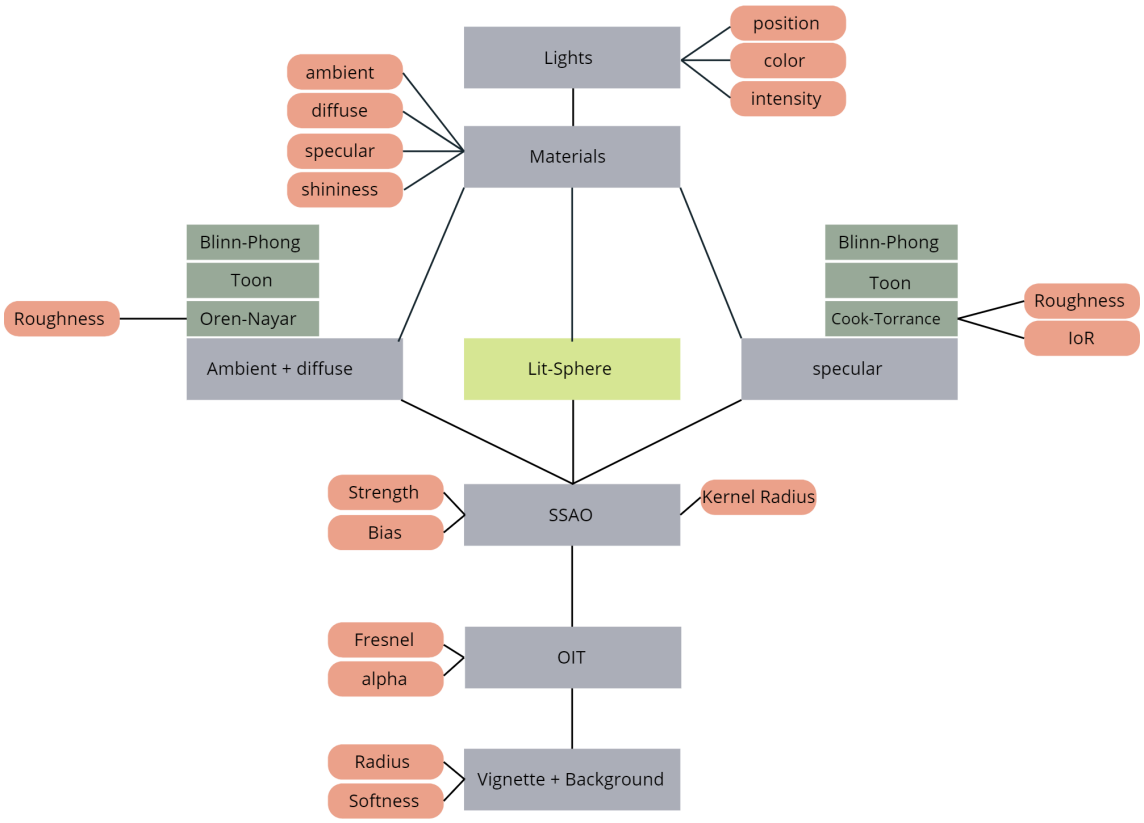


Figure 3.15: Overview of the different parameters that can be modified in each pipeline step. Grey boxes mark the main steps in the pipeline. Green boxes mark the different lighting shaders that can be selected for the specific step, whereby the light sphere shader is in its own category, as depending on the texture used, ambient, diffuse and specular terms can be combined as desired. Red boxes with rounded edges mark parameters that can be modified.

4. Results and Discussion

In this chapter different combinations of techniques are shown and the user customizable values are explained. Additionally, arising problems and limitations are shown and discussed.

The material colors for the ambient and diffuse term can be specified by the user. That means using equation 3.5 p_a and p_d can be set as a three dimensional vector containing the red, green and blue color values (RGB) making up the color for the specified material. The default values set by the tool are: 220, 56 and 63.5, resulting in the pink color used throughout the renderings in this thesis. Additionally, the specular color p_s and shininess factor c (see equation 3.4) of the specular term can be set. The higher the shininess factor gets, the smaller the angle between H and N has to be for a specular highlight to be seen. Furthermore, the color l_c and intensity of the lights l_i can be controlled by the user impacting the diffuse and specular term (see equation 3.6 and equation 3.7). The distance of the lights l_d to the mesh can not be controlled at this time, but the lights can be rotated freely and independently of each other around the mesh using quaternions to calculate the rotations to avoid gimbal lock. The possibility to activate an additional visual aid in the form of a view cube is provided. The view cube can be toggled and is rendered in another viewport in the bottom left of the screen. The cube represents the mesh to show its current orientation and the activated lights are represented by colored spheres. When the light position is changed, the spheres change their position respectively so that the user can always check where the lights are positioned in regard to the mesh (see Fig. 4.1).

Oren-Nayar and Cook-Torrance shading are both based on the idea of micro facets. Therefore, the roughness σ of the surface can be set for both models. The chosen default value for σ is set to 0.2, providing the subjectively best result for the rendered mesh. For Cook-Torrance the incident of refraction for the medium the light is travelling through n_1 and the incident of refraction of the object n_2 can be changed. n_1 is per default set to 1.0 resembling the index of refraction of air and n_2 is set to 2.0 representing the standard value usually used for Cook-Torrance shading.

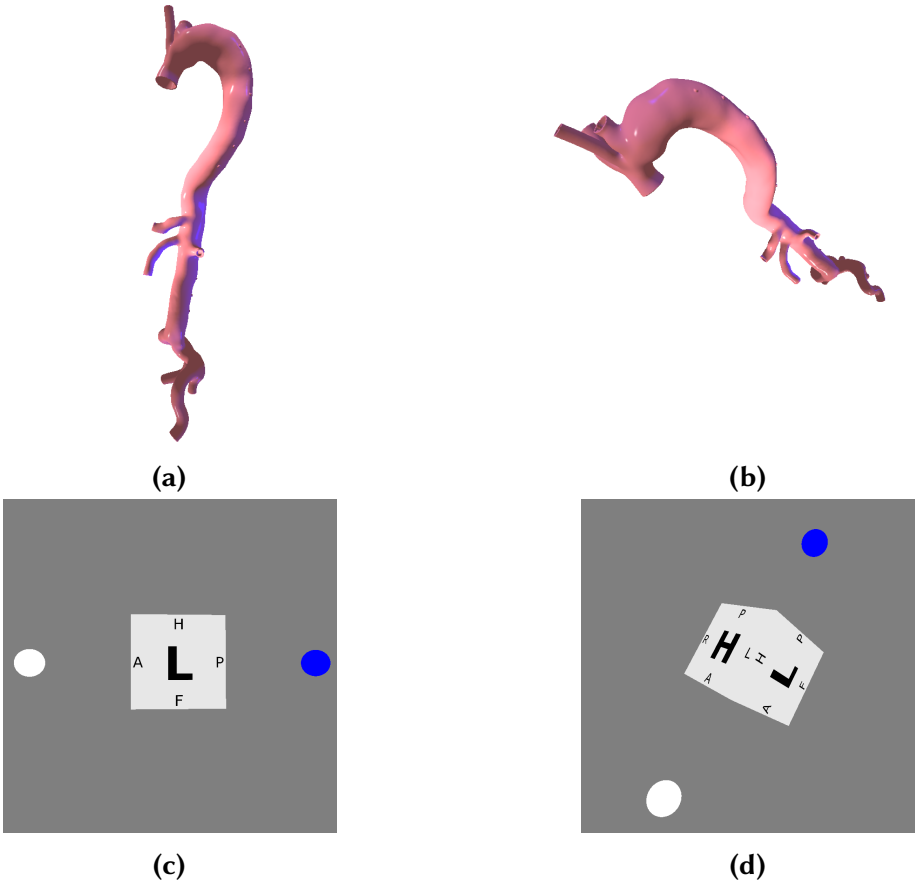


Figure 4.1: View cube used to show the orientation of the mesh and lights. The spheres are colored in the same way as the lights and show the corresponding light position.

For lit-sphere shading, a material gallery (see Fig. 4.2) is added providing textures which are used for the computation of the final color. Here the material gallery provides the ability to choose a different texture for the front and back lighting of the mesh instead of the material color and light color, which are usually used for the calculation of the final shading. It can also be extended with textures for specular highlights, which are added to the lighting calculation. Results are shown in Fig. 4.3 and Fig. 4.4.

SSAO is a useful tool to enhance the depth perception of the scene, when it is tuned correctly. Finding the right tuning can be a difficult task for inexperienced users which might result in a lot of testing to achieve a satisfactory outcome. Although the visualization tool provides pre-set values for the number of samples in the sample kernel, the radius of the sample kernel, the bias and the impact the effect has on the ambient lighting, these values might not be sufficient for every used mesh.

If the radius of the sample kernel is chosen too small, the ambient occlusion is not noticeable, but if it is chosen too large, the entire mesh gets darkened and the ambient occlusion is not noticeable either. The number of samples in the kernel has a huge impact on the final shading quality. If it is chosen too small the, quality of the ambient occlusion could look rough, whereas a too large sample size could impact

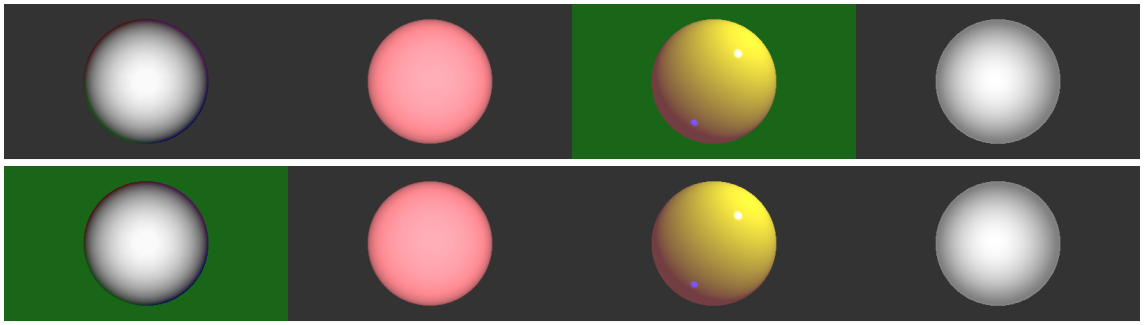


Figure 4.2: Screenshot of the material gallery showing the selectable materials for the lit-sphere shading. The top row sets the material for the front light, whereas the bottom row sets the material for the back light. The green background indicates which material is selected. Materials are sampled from spheres, rendered with different shading approaches from the toolkit.

performance. The bias adds a set value to the depth of the sample the fragment's depth is being compared to. Testing showed that with this bias the visually best results could be achieved. If the bias was set to zero the effect looked very rough and unfinished. The best results were achieved when the bias was set to be twice as high as the radius of the sample kernel. The strength of the ambient occlusion factor is the easiest to tweak, because it only affects the brightness of the effect. Figure 4.5 shows the obtained results for different parameter settings using SSAO with just the ambient term of Blinn-Phong shading. As shown in Fig. 4.5, SSAO improves the depth perception tremendously. Nevertheless, the correct settings for SSAO are of the utmost importance to achieve correct results.

OIT was one of the more complex techniques to get to work properly, causing problems with the different weight functions and parameters the weighted blended order-independent transparency approach of McGuire and Bavoil [30] provided. The results that were achieved with the technique are satisfactory, but further testing with more complex scenes should be conducted to see if the technique holds up when more than two meshes are blended together. Testing of different weight functions $w(z_i, \alpha_i)$ showed no big differences, leading to the assumption that the complexity of the scene was too low to discern benefits of one over the others. The Fresnel effect can be controlled through the edge falloff determining how strong the effect is (i.e. how fast does the alpha of the fragment being rendered fall of depending on the angle between V and N). Renderings resulting from different settings for the edge falloff of the Fresnel effect can be seen in Fig. 4.6.

The diffuse and specular terms of the implemented shading techniques can be combined without limitations and the parameters specific for the term and selected shading technique can be changed by the users: First, users can set the color of the ambient, diffuse and specular light that is reflected by the materials. For the specular term the shininess factor can be set additionally. Next, the users specify which shading algorithm to use for the diffuse term and which shading algorithm to use for the specular term. Then the shader specific parameters have to be set. For example: the user selected lit-sphere shading for the diffuse term and Cook-Torrance for the specular term. For lit-sphere shading, the user can select a texture for the front light

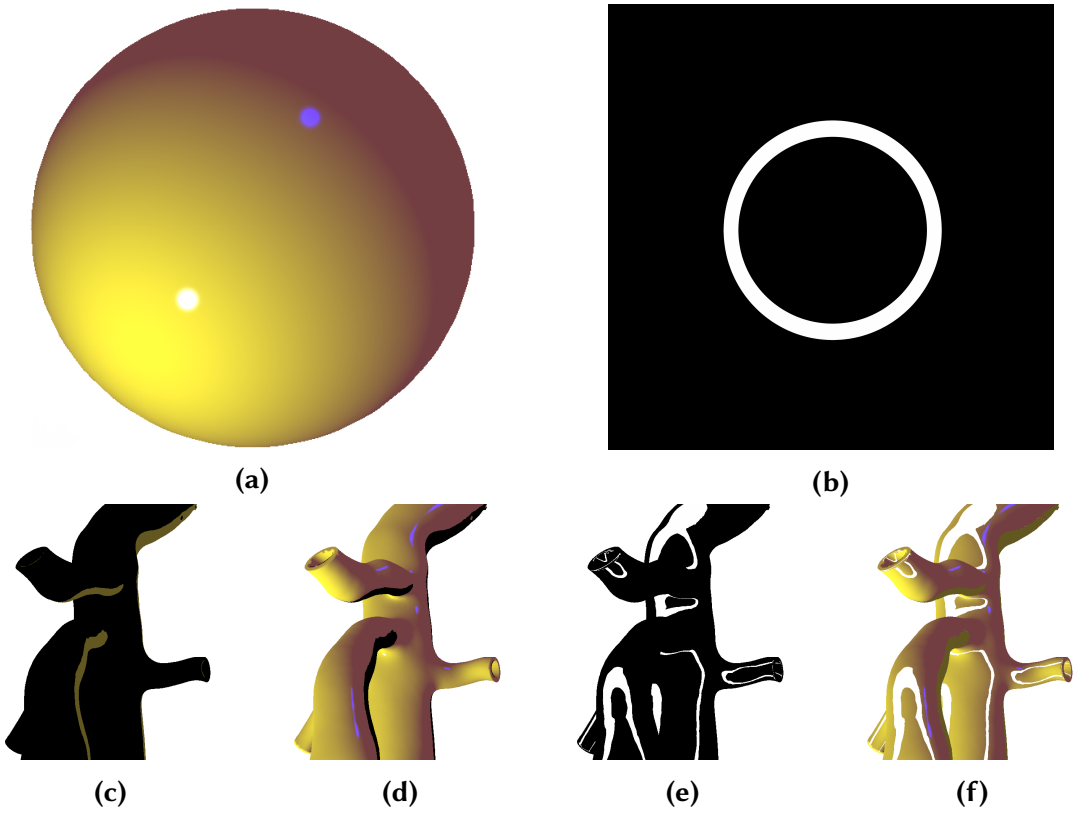
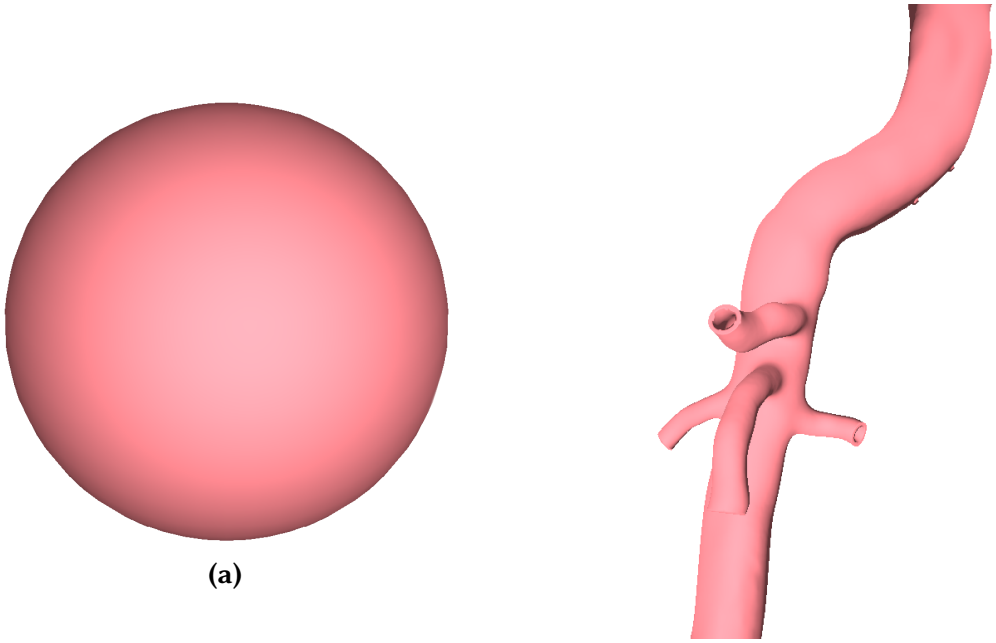


Figure 4.3: Textures used in the lit-sphere shading approach. (a) is used as an ambient and diffuse texture at the same time resulting in the shading of (c). Texture (b) can be used to add an additional highlight with a special form to the shading, which is either controlled by the back light or by the front light. In this shading it was dependent on the front light. The shading just using (b) can be seen in (e). (c) contains the ambient term of the lighting. The ambient light is applied everywhere, where the diffuse part is not used (i.e. where the dot product of the normal and light space normal is negative). The color is determined by using the darkest color of (a). Combining everything (f) is the result.

and the back light and for Cook-Torrance he or she can specify the roughness of the surface σ and the refraction indices n_1 and n_2 (see equation 3.17).

Next, one can choose to activate SSAO and/or OIT. For these techniques he or she can adjust the values previously explained in this section. To finalize the visualization and to create individualized screenshots, an additional effect shader can be activated that surrounds the rendering with a vignette and displays a background texture loaded by the user. The vignette effect can be controlled through parameters for softness of the transition between the fore- and background and the radius (see Fig. 4.7). The background texture can be any portable pixmap (.ppm) image file loaded by the user through the ”load custom background” button.

A problem that arises when combining different techniques with each other is the impact each technique has on the final shading. For the default shading algorithms this can always be controlled through the light and material factors, but lit-sphere shading sets a different challenge: The materials provided for lit-sphere shading have



(a) (b)

Figure 4.4: The texture in (a) was generated from a sphere rendered with ambient and diffuse Phong shading. Using this texture, the lit sphere shading produces the results seen in (b) that are very similar to Phong shading.

to be carefully selected to not diminish the effect of the other shading approaches. All textures to be used with lit-sphere shading have been created using the visualization tool with default shading parameters set. Therefore, the problem should not occur while using the provided textures, but if the tool should be extended to support user generated textures, this should be considered.

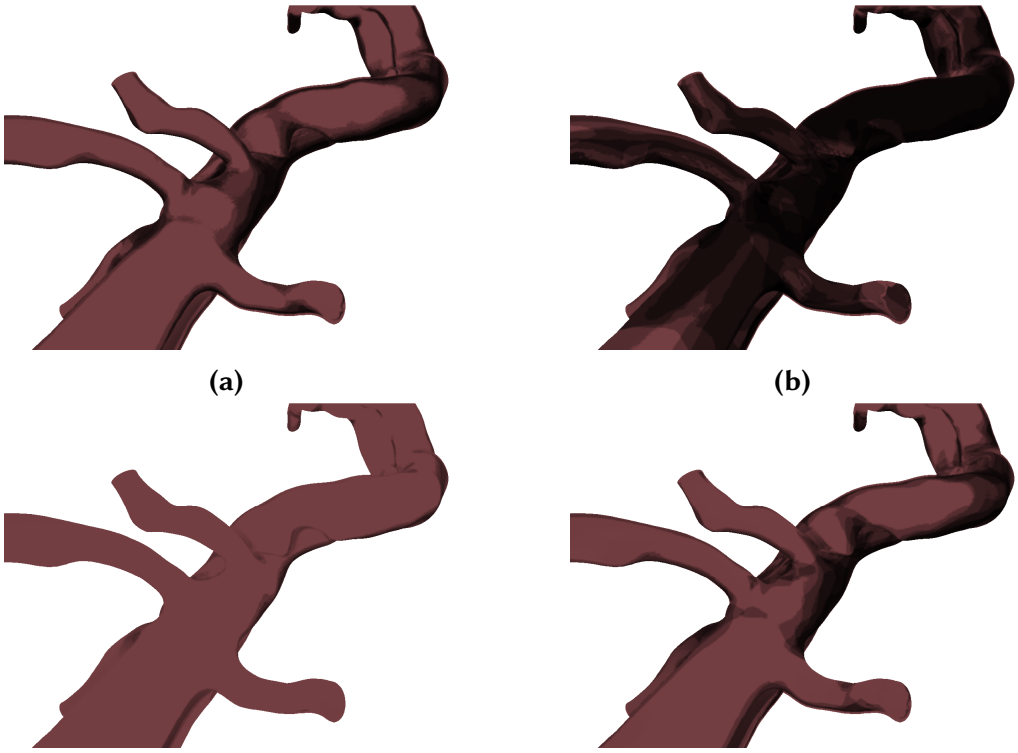


Figure 4.5: Mesh of fluid inside of a vessel, rendered with different settings for SSAO altering the bias and sample kernel radius (b, r) with the strength set to 5.0. The visualizations are rendered with the following settings: (a): (3.0, 3.0). (b): (3.0, 6.0). (c): (6.0, 3.0). (d): (6.0, 6.0).

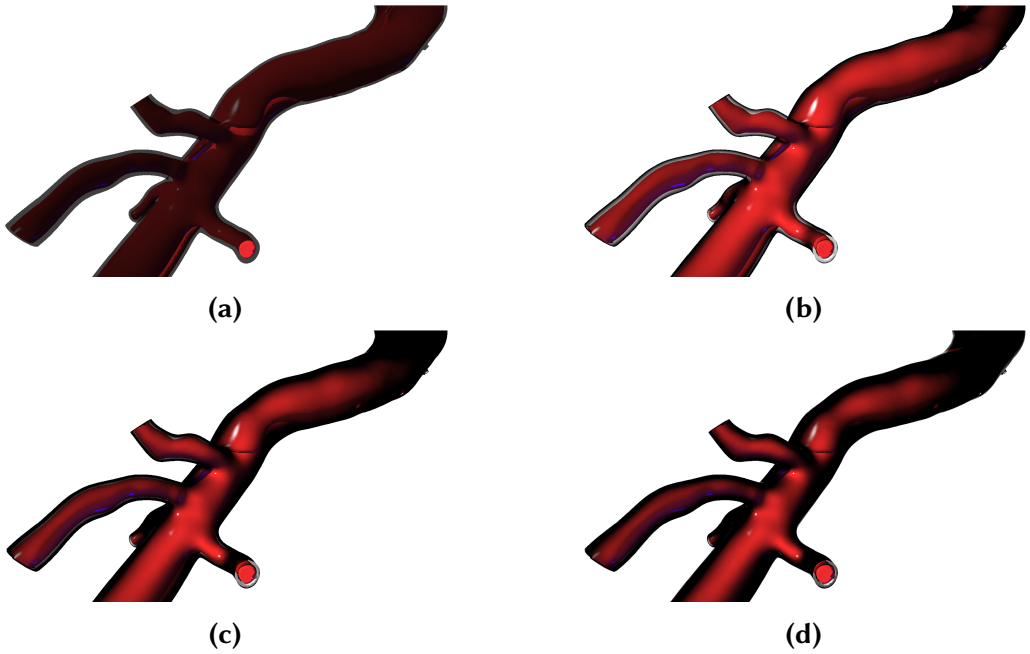


Figure 4.6: Figure displaying the outer vessel wall, rendered transparently using weighted blended order-independent transparency and the inner fluid rendered with Blinn-Phong shading. (a) using no Fresnel effect with alpha set to 0.5. In (b) the edge falloff is set to 0.25. In (c) the edge falloff is set to 0.5. In (d) the edge falloff is set to 0.75.

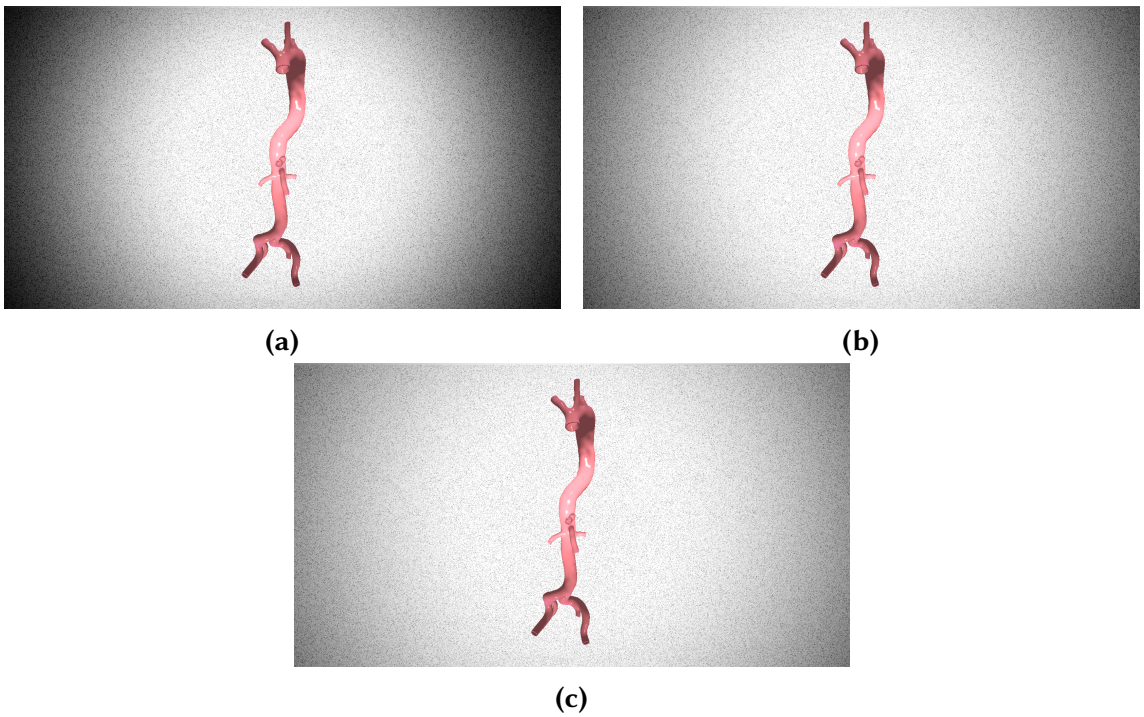


Figure 4.7: Examples for different radius and softness settings (r, s) for the vignette. (a): (1.2, 0.9). (b): (1.5, 1.2). (c): (1.8, 1.5). A custom background in form of a noise texture was set.

5. Conclusion and Future Work

This thesis presented a tool for the visualization of three dimensional surface meshes focusing on vascular structures. Implemented techniques were designed in a way that gives users a high degree of freedom and wide range of customizable parameters to design their own renderings. A high focus was placed on the interactivity through the multiple ways users can change the properties of every important aspect of the different shading steps.

A technique to increase depth perception was implemented in the form of SSAO. Furthermore, OIT is a useful addition if users want to render certain objects transparent and through this obtain more information about occluded geometry or the interior of the object. Making it possible to combine every implemented technique with one another and making almost every parameter of the lighting equation modifiable without using a shader for every combination has been a challenge, but the resulting tool is a useful addition to the framework. Future improvements should aim especially at being compatible and interchangeable with already implemented techniques.

Additions to the visualization tool should further increase the impact users can have on their renderings. These could be more sophisticated approaches to SSAO or the implementation of horizon-based ambient occlusion [4] or voxel ambient occlusion [35].

Since the implemented approach of weighted blended OIT may not be the best one to render complex scenes with many different transparent objects, other techniques based on depth peeling or linked lists could be made available for selection so that users can decide which approach best suits their needs.

The lit-sphere rendering could be improved by allowing users to directly create their own materials from their created shadings by projecting the shading of their chosen render settings onto a sphere and saving the created texture directly to the material gallery. Additionally, the users should be able to load their own materials from files.

BRDF's containing approximations for sub-surface scattering would be a useful addition, especially to render vessel materials more realistically. The implementation

of a depth-of-field technique would add another stylistic layer to the visualization tool, that could be toggled and adjusted by users, to further increase the attractiveness of composited scenes. Furthermore, models using approaches to calculate the impact of refraction could be implemented to enable the realistic modeling of fluids.

Apart from the implementation of new techniques, the user interface could be revised to improve the workflow of users. It should support context based selections of parameters, highlighting which parameter can be used currently and hide parameters that currently have no effect. In addition, each parameter should have an explanation and examples of its effect in a window easily accessible by users (i.e. double click, right click, mouse hover).

Bibliography

[1] P. Barla, J. Thollot, and L. Markosian. X-Toon. In *International Symposium on Non-photorealistic Animation and Rendering*, volume 2006, pages 127–132. ACM Press, 2006. doi: 10.1145/1124728.1124749. (cited on Page 4)

[2] P. Barta and B. Kovács. Order Independent Transparency with Per-pixel Linked Lists. In *Central European Seminar on Computer Graphics*, 2011. (cited on Page 5 and 20)

[3] L. Bavoil and K. Meyers. Order Independent Transparency with Dual Depth Peeling. Technical report, Nvidia, 2008. (cited on Page 5)

[4] L. Bavoil, M. Sainz, and R. Dimitrov. Image-space Horizon-based Ambient Occlusion. In *ACM SIGGRAPH*, page 1. ACM Press, 2008. doi: 10.1145/1401032.1401061. (cited on Page 5 and 33)

[5] B. Behrendt, P. Berg, O. Beuing, B. Preim, and S. Saalfeld. Explorative Blood Flow Visualization Using Dynamic Line Filtering Based on Surface Features. *Computer Graphics Forum*, 37(3):183–194, 2018. doi: 10.1111/cgf.13411. (cited on Page 2 and 6)

[6] K. Berbaum, T. Bever, and C. S. Chung. Light Source Position in the Perception of Object Shape. *Perception*, 12(4):411–416, 1983. doi: 10.1068/p120411. (cited on Page 3)

[7] J. F. Blinn. Models of Light Reflection for Computer Synthesized Pictures. *ACM SIGGRAPH Computer Graphics*, 11(2):192–198, 1977. ISSN 0097-8930. doi: 10.1145/965141.563893. (cited on Page 10)

[8] R. L. Cook and K. E. Torrance. A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics*, 1(1):7–24, 1982. ISSN 0730-0301. doi: 10.1145/357290.357293. (cited on Page 1 and 14)

[9] J. de Vries. *Learn OpenGL*. Kendall & Welling, 2020. ISBN 9090332561. (cited on Page 18)

[10] D. DeCarlo and S. Rusinkiewicz. Highlight Lines for Conveying Shape. In *Symposium On Non-photorealistic Animation And Rendering*, NPAR '07, page 63–70, New York, NY, USA, 2007. ACM Press. ISBN 9781595936240. doi: 10.1145/1274871.1274881. (cited on Page 4)

[11] J. Díaz, P. Vázquez, I. Navazo, and F. Duguet. Real-time Ambient Occlusion and Halos with Summed Area Tables. *Computers & Graphics*, 34(4):337–350, 2010. doi: 10.1016/j.cag.2010.03.005. (cited on Page 4)

[12] J. Díaz-García and P. Vázquez. Fast Illustrative Visualization of Fiber Tracts. In *Advances in Visual Computing*, pages 698–707. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-33179-4_66. (cited on Page 6)

[13] W. Engel. *ShaderX7 : Advanced Rendering Techniques*. Course Technology/-Cengage Learning, Australia, 2009. ISBN 9781584505983. (cited on Page 2)

[14] C. Everitt. Interactive Order-independent Transparency. Technical report, Nvidia, 2001. (cited on Page 5)

[15] R. Gasteiger, M. Neugebauer, C. Kubisch, and B. Preim. Adapted Surface Visualization of Cerebral Aneurysms with Embedded Blood Flow Information. In *Eurographics Workshop on Visual Computing for Biology and Medicine*. The Eurographics Association, 2010. ISBN 978-3-905674-28-6. doi: 10.2312/VCBM/VCBM10/025-032. (cited on Page 5 and 6)

[16] B. Gooch and A. Gooch. *Non-photorealistic Rendering*. A K PETERS LTD (MA), 2001. ISBN 1568811330. (cited on Page 4)

[17] D. Gross and S. Gumhold. Advanced Rendering of Line Data with Ambient Occlusion and Transparency. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):614–624, 2021. ISSN 1941-0506. doi: 10.1109/tvcg.2020.3028954. (cited on Page 4)

[18] W. Hao, W. Che, X. Zhang, and Y. Wang. 3d Model Feature Line Stylization Using Mesh Sharpening. In *Conference on Virtual-Reality Continuum and its Applications in Industry*, pages 249–256. ACM Press, 2010. doi: 10.1145/1900179.1900233. (cited on Page 4)

[19] D. C. J. Kahrs, S. Calahan and S. Poster. Pixel Cinematography: A Lighting Approach for Computer Graphics. In *ACM SIGGRAPH Course Notes*, 1996. (cited on Page 3)

[20] A. Khan, I. Mordatch, G. Fitzmaurice, J. Matejka, and G. Kurtenbach. ViewCube. In *Symposium on Interactive 3D graphics and games - SI3D*, pages 17–25. ACM Press, 2008. doi: 10.1145/1342250.1342253. (cited on Page 2)

[21] S. J. Koppal. Lambertian Reflectance. In *Computer Vision*, pages 441–443. Springer US, 2014. doi: 10.1007/978-0-387-31439-6_534. (cited on Page 12)

[22] M. S. Langer and H. H. Bülthoff. Depth Discrimination from Shading under Diffuse Lighting. *Perception*, 29(6):649–660, 2000. doi: 10.1068/p3060. (cited on Page 4)

[23] K. Lawonn, R. Gasteiger, and B. Preim. Adaptive Surface Visualization of Vessels with Animated Blood Flow. *Computer Graphics Forum*, 33(8):16–27, may 2014. doi: 10.1111/cgf.12355. (cited on Page 6)

[24] K. Lawonn, I. Viola, T. Isenberg, and B. Preim. A Survey of Surface-based Illustrative Rendering for Visualization. *Computer Graphics Forum*, 37(6): 205–234, 2018. doi: 10.1111/cgf.13322. (cited on Page 4)

[25] C. H. Lee, X. Hao, and A. Varshney. Geometry-dependent Lighting. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):197–207, 2006. doi: 10.1109/tvcg.2006.30. (cited on Page 3)

[26] G. Liu, D. Ceylan, E. Yumer, J. Yang, and J. Lien. Material Editing Using a Physically Based Rendering Network. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2280–2288. IEEE, 2017. doi: 10.1109/iccv.2017.248. (cited on Page 4)

[27] T. Luft, C. Colditz, and O. Deussen. Image Enhancement by Unsharp Masking the Depth Buffer. *ACM Transactions on Graphics*, 25(3):1206–1213, 2006. doi: 10.1145/1141911.1142016. (cited on Page 6)

[28] R. Mantiuk. Gaze-dependent Screen Space Ambient Occlusion. In *Computer Vision and Graphics*, pages 16–27. Springer International Publishing, 2018. doi: 10.1007/978-3-030-00692-1_2. (cited on Page 4)

[29] M. Maule, J. L. D. Comba, R. Torchelsen, and R. Bastos. Memory-Efficient Order-Independent Transparency with Dynamic Fragment Buffer. In *25th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 134–141. IEEE, 2012. doi: 10.1109/sibgrapi.2012.27. (cited on Page 5)

[30] M. McGuire and L. Bavoil. Weighted Blended Order-independent Transparency. *Journal of Computer Graphics Techniques (JCGT)*, 2(2):122–141, 2013. ISSN 2331-7418. (cited on Page 2, 5, 20, and 27)

[31] M. McGuire and J. F. Hughes. Hardware-determined Feature Edges. In A. Hertzmann, editor, *Non-photorealistic animation and rendering*, page 35, New York, NY, 2004. ACM Press. ISBN 1581138873. doi: 10.1145/987657.987663. (cited on Page 6)

[32] C. Münstermann, S. Krumpen, R. Klein, and C. Peters. Moment-Based Order-Independent Transparency. *ACM on Computer Graphics and Interactive Techniques*, 1(1):1–20, 2018. doi: 10.1145/3203206. (cited on Page 5)

[33] M. Oren and S. Nayar. Generalization of Lambert's reflectance model. In D. Schweitzer, editor, *Computer graphics and interactive techniques*, pages 239–246, New York, NY, 1994. ACM Press. ISBN 0897916670. doi: https://doi.org/10.1007/BF01679684. (cited on Page 1 and 12)

[34] S. Park and N. Baek. Outline-dependent Screen-space Ambient Occlusion. In *Lecture Notes in Electrical Engineering*, pages 193–200. Springer Singapore, 2019. doi: 10.1007/978-981-15-1465-4_20. (cited on Page 4)

[35] R. Penmatsa, G. Nichols, and C. Wyman. Voxel-space Ambient Occlusion. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, page 1. ACM Press, 2010. doi: 10.1145/1730804.1730989. (cited on Page 33)

[36] B. T. Phong. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18(6):311–317, 1975. doi: https://doi.org/10.1145/360825.360839. (cited on Page 1, 6, and 9)

[37] B. Preim, A. Baer, D. Cunningham, T. Isenberg, and T. Ropinski. A Survey of Perceptually Motivated 3d Visualization of Medical Image Data. *Computer Graphics Forum*, 35(3):501–525, 2016. doi: 10.1111/cgf.12927. (cited on Page 4)

[38] T. Ropinski, J. Meyer-Spradow, S. Diepenbrock, J. Mensmann, and K. Hinrichs. Interactive Volume Rendering with Dynamic Ambient Occlusion and Color Bleeding. *Computer Graphics Forum*, 27(2):567–576, 2008. doi: 10.1111/j.1467-8659.2008.01154.x. (cited on Page 4)

[39] C. Schlick. A Customizable Reflectance Model for Everyday Rendering. In *Fourth Eurographics Workshop on Rendering*, pages 73–83, 1993. (cited on Page 6)

[40] C. Schlick. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum*, 13(3):233–246, 1994. doi: 10.1111/1467-8659.1330233. (cited on Page 15)

[41] P. Sloan, W. Martin, and A. Gooch. The Lit Sphere: A Model for Capturing Npr Shading from Art. *Graphics Interface 2001*, Ottawa:143–150, 2001. doi: 10.20380/GI2001.17. (cited on Page 1 and 16)

[42] B. Smith. Geometrical shadowing of a random rough surface. *IEEE Transactions on Antennas and Propagation*, 15(5):668–671, 1967. doi: 10.1109/tap.1967.1138991. (cited on Page 15)

[43] E. M. Sparrow and R. D. Cess. *Radiation heat transfer*. Hemisphere Pub. Corp, Washington, 1978. ISBN 9780891169239. (cited on Page 14)

[44] N. Thibieroz. *Order-independent Transparency Using Per-pixel Linked Lists*, chapter Part VII. Chapter 2. Order-independent transparency using per-pixel linked lists, pages 409–431. A K PETERS LTD (MA), 2011. ISBN 1568817185. (cited on Page 6)

[45] H. Todo, K. Anjyo, and S. Yokoyama. Lit-sphere Extension for Artistic Rendering. *The Visual Computer*, 29(6-8):473–480, 2013. ISSN 1432-2315. doi: 10.1007/s00371-013-0811-7. (cited on Page 2, 16, and 17)

[46] K. E. Torrance and E. M. Sparrow. Theory for Off-Specular Reflection From Roughened Surfaces. *Journal of the Optical Society of America*, 57(9):1105–1114, 1967. doi: 10.1364/josa.57.001105. (cited on Page 12 and 14)

[47] T. S. Trowbridge and K. P. Reitz. Average Irregularity Representation of a Rough Surface for Ray Reflection. *Journal of the Optical Society of America*, 65(5):531–536, 1975. doi: 10.1364/josa.65.000531. (cited on Page 15)

[48] R. van Pelt, J. O. Bescós, M. Breeuwer, R. E. Clough, M. E. Gröller, B. ter Haar Romenij, and A. Vilanova. Exploration of 4d MRI Blood Flow Using Stylistic Visualization. *IEEE Transactions on Visualization and Computer*

Graphics, 16(6):1339–1347, 2010. doi: 10.1109/tvcg.2010.153. (cited on Page 1, 4, and 11)

[49] J. Vermeer, L. Scandolo, and E. Eisemann. Stochastic-depth Ambient Occlusion. Master's thesis, Delft University of Technology, 2021. (cited on Page 5)

[50] X. Wu and L. Zhao. Study on Iris Segmentation Algorithm Based on Dense U-Net. *IEEE Access*, 7:123959–123968, 2019. doi: 10.1109/access.2019.2938809. (cited on Page 5)

[51] D. Zhang, C. Xian, G. Luo, Y. Xiong, and C. Han. DeepAO: Efficient Screen Space Ambient Occlusion Generation Via Deep Network. *IEEE Access*, 8: 64434–64441, 2020. doi: 10.1109/access.2020.2984771. (cited on Page 4)