České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Eliška Kuběnová

Studijní program: Otevřená informatika
Obor: Počítačová grafika a interakce

Název tématu: Návrh a implementace software pro test percepce průhlednosti

Pokyny pro vypracování:

Seznamte se s metodami pro nefotorealistické vykreslování 3D objektů, které se snaží zlepšit percepci faktorů: tvar, vzdálenost, uspořádání dle vzdálenosti od kamery. Dále analyzujte možnosti, jak existující metody kombinují více faktorů dohromady (např. se snaží současně zlepšit percepci tvaru a uspořádání dle vzdálenosti od kamery). Na základě analýzy pro každý faktor vyberte a implementujte alespoň 2 metody.
Dále navrhněte a implementujte alespoň dva způsoby kombinace alespoň dvou faktorů. Implementace by měla umožňovat 2.5D výstup ve formě několika vrstev poloprůhledných obrázků. Výsledek své práce demonstrujte alespoň na pěti 3D modelech různé složitosti. Vyberte dvě metody ovlivňující stejný faktor a proveďte jejich porovnání pomocí uživatelských testů.

Seznam odborné literatury:

1) Preim, B., Baer, A., Cunningham, D., et al. (2016). A Survey of Perceptually Motivated 3D Visualization of Medical Image Data. In Computer Graphics Forum (Vol. 35, No. 3, pp. 501-525). Blackwell Publishing Ltd.
2) Baer, A., Gasteiger, R., Cunningham, D., Preim, B. (2011). Perceptual evaluation of ghosted view techniques for the exploration of vascular structures and embedded flow. In Computer Graphics Forum (Vol. 30, No. 3, pp. 811-820). Blackwell Publishing Ltd.

Vedoucí: Ing. Ladislav Čmolík, Ph.D.

Platnost zadání: do konce zimního semestru 2018/2019

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
4
L.S.

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 3.4.2017

Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Computer Graphics
and Interaction

Master's thesis

Design and Implementation of Software for Transparency Perception Tests

Bc. Eliška Kuběnová

Supervisor: Ing. Ladislav Čmolík, Ph.D.

Study Programme: Open Informatics
Field of Study: Computer Graphics and Interaction

24th May 2018

Acknowledgements

First of all, I would like to thank Ing. Ladislav Čmolík, Ph.D. for his kind and professional guidance, for valuable consultations and patience during supervising this thesis. I extend my gratitude to my parents, my loved ones, and my friends for their love and support that guided me not only when I was working on my thesis but also during my life.

Declaration

In Prague on 24th May 2018 . . . .

Citation of this thesis

Kuběnová, Eliška. *Design and Implementation of Software for Transparency Perception Tests*. Master's thesis. Czech Technical University in Prague, Faculty of Electrical Engineering, 2018.

Abstrakt

Tato diplomová práce se zabývá analýzou nefotorealistických metod pro vykreslování 3D objektů, které se snaží zlepšit percepci faktorů: tvar, vzdálenost a uspořádání dle vzdálenosti od kamery, a jejich možnými kombinacemi. Cílem práce je implementovat metody, které umožňují zprůhledňování na základě těchto faktorů, a vyzkoušet jejich kombinace. Na základě analýzy a výsledků dále navrhnout software, který umožní testování percepce. Byla navržena a provedena uživatelská studie, ve které byla testována percepce čtyř různých metod pro vizualuzaci poloprůhledných 3D objektů včetně jedné metody nefetorealistické průhlednosti. Uživatelská studie byla provedena k ověření celého procesu. V průběhu studie se nevyskytly, žádné problémy.

Klíčová slova nefotorealistické vykreslování, percepce tvaru, percepce hloubky, percepce uspořádání objektů dle vzálenosti od kamery, vizuální podněty, metody pro vylepšení percepce, kombinace faktorů

Abstract

This thesis deals with an analysis of non-photorealistic methods for visualization of 3D objects. These methods should improve the perception of three factors: shape, depth and depth-order, and their possible combinations. The goal of this work is to implement methods that allow us to modulate opacity of the objects based on these factors, and try their combinations. Based on the analysis and results then design and implement a software for perception tests. The user perception study has been designed. In this study, the perception of four visualization methods of 3D objects was performed including one method of non-photorealistic opacity modulation. The user study was conducted to verify the entire process. There were no problems during the study.

Keywords non-photorealistic visualization, shape perception, depth perception, depth-order perception, visual cues, enhancement methods, combination of visual cues

Contents

List of Figures

Introduction

In this work, I will deal with the problem of non-photorealistic visualization of semi-transparent 3D meshes.

In the recent years, the bewildering amount of approaches to non-photorealistic visualization of semi-transparent surfaces appeared. However, most of the approaches were not perceptually evaluated with humans. Therefore, it is not entirely clear in which situation one should use which approach.

There are three main goals of this thesis:

1. To analyze the existing approaches to non-photorealistic visualization of semi-transparent 3D meshes and determine the visual cues used in the majority of the approaches. In the whole thesis, I will focus especially on visual cues that could improve the perception of the shape of the 3D mesh, perception of distance between semi-transparent surfaces of the 3D mesh, and perception of depth order of the semi-transparent surfaces.

2. To design and implement an application that will allow the user to easily combine desired visual cues together, visualize semi-transparent 3D meshes using the visual cues, and export the scene in a form that will ease the creation of data for perceptual user studies.

3. To design, implement and perform perceptual user study utilizing the implemented application from the previous step and web service for collecting data from empirical studies developed as diploma thesis by Lebedova [15].

There is a potential that such application could increase the number of performed perceptual studies and consequently increase the knowledge about the perception of semi-transparent 3D meshes visualized with non-photorealistic approaches.

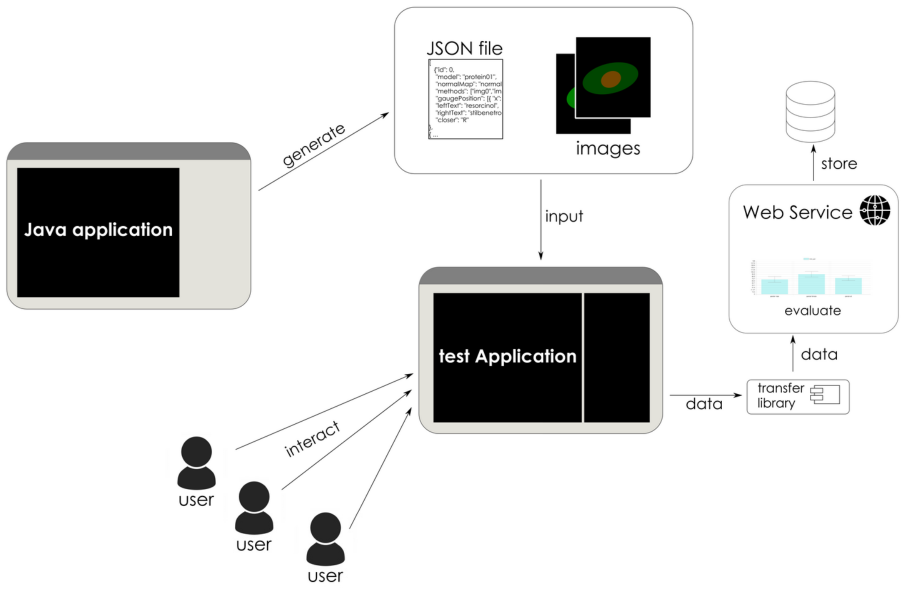


Figure 0.1: Scheme of interconnection of individual parts.

For a better idea how individual parts are connected in this concept, I attach a simple scheme in figure 0.1. The generator in visualization application will generate JSON file with simple scene definitions along with a set of images. It creates input that is possible to connect with the test application. The users than participate in user perception study by interacting with the test application. From the test application, data are sent to web service for evaluation.

Structure of the Following Text

The following text is organized as follows: Chapter 1 discusses theoretical background that is needed for problem understanding. Then the analysis of possible visual cues and existing enhancement methods for each factor is presented. These methods were categorized based on three factors.

In Chapter 2 I describe in more details visual cues and enhancement methods that I chose for the implementation. I also describe a design for an application that is able to load 3D model with several meshes and allows the user to combine more visual cues together. Moreover, a design of test application is presented.

In Chapter 3 I describe more implementation details and the structure of application parts. Chapter 4 presents final results with possible visual cues combinations and test application walkthrough.

And finally, in Chapter 5, the perceptual user study is presented. It elaborates the experiment setup and design and provides the user test results.

Chapter 1

Analysis

In the following chapter, I will discuss theoretical background that is needed for problem description and understanding. The first section focuses on the principles of the rendering of semi-transparent objects. The algorithms as ray casting and depth peeling will be presented along with the duality of this problem. A simple modification of depth peeling algorithm that allows us to modify the opacity will be also mentioned.

In the second section, the analysis of possible visual cues and existing enhancement methods for each factor will be described. Furthermore, methods suitable for combining multiple factors will be mentioned.

1.1 Theoretical Background

To describe some of the advanced techniques a related theory should be introduced first. In this section I will introduce problems associated with this work such as the problem of rendering semi-transparent objects.

1.1.1 Rendering semitransparent objects

1.1.1.1 Depth Buffer

Natural approach to rendering opaque objects in correct order is using a z-buffer. The z-buffer or depth buffer contains the depth of the closest pixel to the camera. Using depth test we can find out which fragments become pixels. Only these pixels are rendered. Without depth buffer the pixels are rewritten by any later fragments, regardless to their depth.

Since rendering of semitransparent geometry needs to take all of semitransparent fragments into consideration, the depth buffer cannot be used. Without depth buffer we have to solve the order of rendered objects to get correct result.

We have to take semi-transparent objects in back-to-front or front-to-back order, so we can render the objects properly with correct alpha blending. For

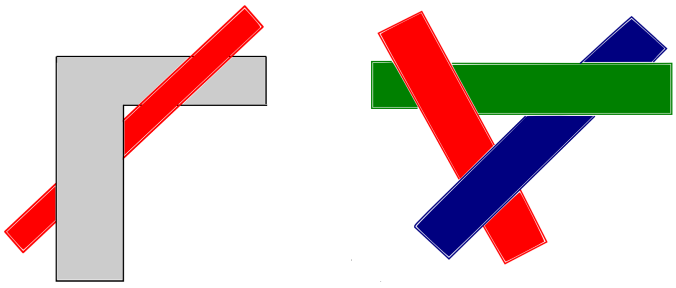


Figure 1.1: Problem cases for the Painter's algorithm [1].

example, the Painter's algorithm [1] is based on sorting objects by depth and rendering them in back-to-front order. It partially solves the problem of the wrong order. However it fails to solve a depth cycle. Problem cases are shown in figure 1.1.

1.1.2 Order Independent Transparency (IOT)

Order Independent Transparency (IOT) is common term for techniques that do not require sorting objects before rendering. The order of fragments is the most important thing for those algorithms. In this section I will describe two possible representations of the problem.

1.1.2.1 Ray Casting

Even though ray casting is not directly used in existing implementation, I will present a description of algorithms that use it, so that the whole topic can be understood thoroughly. The ray casting is commonly used in computer graphics and it brings us a way to sort fragments of objects along rays. In volume rendering we only deal with primary rays. The solution is straightforward.

Algorithm 1 Ray Casting

```
For each pixel of final image
  1. Shoot primary ray.
  2. Find intersections of all 3D objects with the ray. Denote them as
     samples.
  3. Sort the samples according to their depth.
  4. Compose the samples from back to front using the over operator or
     from front to back using the under operator.
```

1.1.2.2 Depth Peeling

Depth peeling is an iterative algorithm for rendering of 3D objects. It is operating on a per-fragment basis. The essence of this technique is that with n passes over the scene, we can get n layers deeper into the scene. With this approach we get both depth and color information of each layer. This method is an alternative to previously introduced ray casting. The depth peeling algoritm is suitable for GPU implementation and it will be used further in this work.

The first pass of depth peeling algoritm is standard rendering pass with depth test. In the second pass, we use the depth buffer computed in the first pass to “peel away” depths that are not farther than nearest depths from the first pass. We compute depth buffer for second pass, which can be later used to “peel away” first and second layer. For each fragment we need to perform two depth tests [16].

In existing implementation, each layer is stored in two floating point textures. The color of the samples is stored in the first texture. The second texture contains normal vectors and depths. The depth peeling algorithm is modified by bitmask texture to consider only the first sample of each group of 3D objects. Another modification is based on importance filtering. Only samples with higher importance than the previous ones are taking into account. Modified depth peeling can be found in [17]. General algorithm (algorithm 2) for depth peeling on GPU follows. We need depth buffer D, shadow map S [16], texture for current layer L and result would be stored to texture R.

Algorithm 2 Depth Peeling on GPU

```
Depth test enabled.
Render geometry.
Store layer $L$ (consisting of the nearest fragment) to $R$.
While there are still fragments to be processed
    1. Enable depth test.
    2. Render geometry.
    3. Peel away fragments from previous layer using $S$.
    4. Store result to $L$.
    5. Create shadow map $S$ from the camera view.
    6. Disable depth test.
    7. Render quad.
    8. Blend $L$ to $R$.
In final pass, render texture R to framebuffer.
```

1.1.2.3 Dual Representation of the Problem

As ray casting and depth peeling algorithms were introduced it is appropriate to point out the dual representation of the problem.

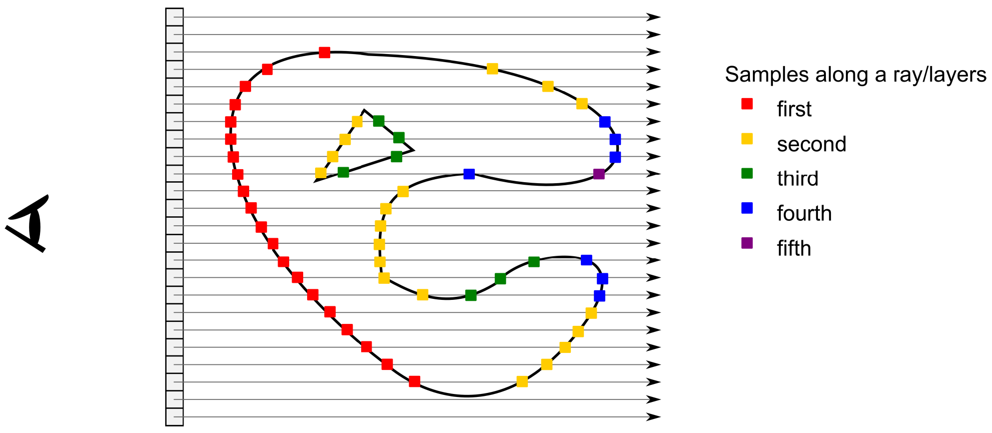


Figure 1.2: Dual representation of the problem. Samples are ordered along a ray. Moreover, samples marked with the same color form a layer.

Figure 1.2 shows us the dual representation. If we consider the ray casting terminology, i.e. sorted samples along the ray, we can consider them as sorted layers. The number of layers is then equal to maximal number of samples along the rays.

When the ray tracing is used, the distance between fragments along the ray is known. In case of depth peeling, we do not have this information, but in contrary to ray casting, with depth peeling, we know which fragment is in the same layer.

Since we only need to know the distance to the first following fragment, we are able to modify the depth peeling algorithm as presented in section 1.1.2.2 so that we have two layers of fragments in each subsequent iteration. Therefore we are able to modulate the opacity of the layer based on the shape (thanks to neighbouring fragments in the layer). Moreover, we can modulate the opacity based on the depth since we know the relative distance between two layers.

The composition of the layers into the final image is similar to composition of samples along one ray and it is described below [17].

1.1.2.4 Composing the Final Image

If we consider the final image composition, the fragments need to be sorted along the ray. Each direction of sorting (front-to-back, back-to-front) requires different blending approach using over and under operator.

Over Operator An over operator is the key to the back to front composition. The over operator is, in effect, a normal painting operation. The Painter's algorithm which was mentioned in section 1.1.1.1 works on the same principle.

Let us consider denote that we have samples s_1 ...s_m sorted by their distance from the viewpoint. The sample s_i is a point on a surface of the 3D object and it has unique identifier of the object it belongs to associated. Each sample has its color $\vec{c_i}$, opacity α_i, normal vector $\vec{n_i}$, and position $\vec{p_i}$. The samples along each ray are composed using equation 1.1

$$\vec{C} = \vec{C_1} \; over \; (\vec{C_2} \; over \; ... \; (\vec{C_{m-1}} \; over \; \vec{C_m}) \; ... \;) \tag{1.1}$$

where $\vec{C_i} = \vec{c_i} * \alpha_i$ of sample s_i, $i \in 1...m$, premultiplied with opacity of the sample and the over operator is

$$\vec{C_{out}} = \vec{C_i} \; over \; \vec{C_{in}} = \vec{C_i} + (1 - \alpha_i) * \vec{C_{in}} \tag{1.2}$$

By using the equation 1.2 we get

$$\vec{C} = \vec{C_1} + \vec{C_2} * (1 - \alpha_i) + ... + \vec{C_m} * (1 - \alpha_i) * ...(1 - \alpha_m - 1)) \tag{1.3}$$

Under Operator Other approach is to use the front to back composition with under operator. With this approach we need to accumulate opacities through the composition. The samples are composed using equation

$$\vec{C} = \vec{C_m} \; under \; (\vec{C_{m-1}} \; under \; ... \; (\vec{C_2} \; under \; \vec{C_1}) \; ... \;) \tag{1.4}$$

As in previous case we get

$$\vec{C_{out}} = \vec{C_i} \; under \; \vec{C_{in}} = \vec{C_{in}} + T_{in} * \vec{C_i} \tag{1.5}$$

$$T_{out} = T_{in} * (1 - \alpha_i); \tag{1.6}$$

where T_{out} is the transparency accumulated along the ray. Its initial value is $T_1 = (1 - \alpha_1)$. In existing implementation the front to back composition with under operator is used [17].

1.1.3 Opacity Modulation

Opacity modulation is a desired feature of the OIT algorithms. In general, the opacity could be modulated based on lighting intensity, density, surface curvature, importance, distance between samples along the view ray, cut motivated approach that computes distance from defined plane or it could be based on any other feature.

Various importance, curvature and distance measures exist. In next section I will present methods that emphasize shape, depth or depth ordering in semitransparent objects, where different opacity modulations are applied.

1.2 Analysis of Visual Cues in Existing Methods

The problem with transparency perception has been studied in psychology for decades. Unfortunately, there is no comprehensive overview or studies that would be dealing with comparison of different approaches. Various factors like shadow, lighting, contrast have been considered as critical visual cues to reveal transparency. As was mentioned before, methods suitable for both volumetric data and polygonal representation of 3D models will be analyzed since some of the enhancement principles used in volume rendering can be modified for 3D meshes.

Polygonal representation of 3D objects is a collection of vertices, edges, and faces. The vertices define the shape of the objects and other information gives us the topology of the surface.

Volumetric data are represented similarly like a 2D image. The image consists of pixels (picture elements). The volumetric data are organized in a 3D regular grid and it consists of voxels (volume elements). Volumetric data can be obtained for example by CT (Computed Tomography).

It is generally difficult to simultaneously visualize interior and exterior structures of rendered objects. To reveal interior structures in volumetric rendering, we can use steep transfer function, volume clipping, ghosting, etc. With steep transfer function, i.e. those with rapidly increasing opacity at a particular range of interest, we may increase depth cues by the appearance of some surface, but it is done by hiding all information of a part of the volume.

Thus, volume clipping allows us to cut selected parts away of the volume. Sometimes it is the only way to reveal important details inside the volume. However, this method is generally not data-aware and it does not take any volume features into account. Since we want to focus on transparency perception, this method is not suitable for the purpose of this thesis.

Ghosting approach is rather commonly used in illustrative methods. With ghosting, the less important parts are, the more transparent they become. The goal of ghosting technique is to provide enough cues to enable mental completion of the partly removed structures [18]

As was mentioned before, the non-photorealistic (NPR) approaches allow us to modulate opacity based on object features for better visual perception. In this section I want to discuss a variety of feature enhancement techniques that tries to improve perception, relates to transparency and focuses on shape, depth and depth ordering perception.

1.2.1 Shape Perception

The shape perception is a quite complex problem. When the 3D object is projected to a 2D image, we lose a considerable amount of information [8]. Psychological studies revealed that silhouette and hatching lines might improve understanding of images content. Outline or silhouettes are probably the sparsest meaningful visualization that can roughly convey the object shape. In figure 1.3 the simple overview is presented.

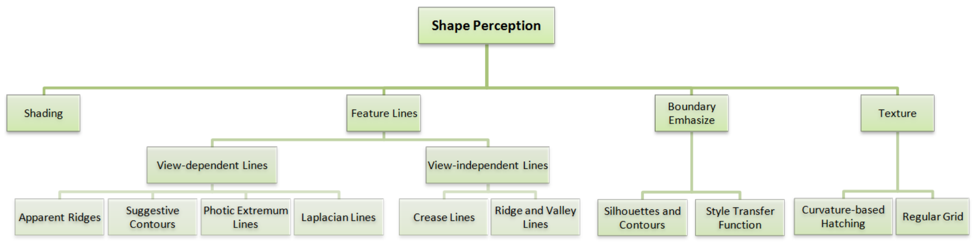


Figure 1.3: Shape perception techniques are based on shading, lines, textures and boundary emphasis [8].

1.2.1.1 Shading, Colors and Illumination

The natural approach to shape perception is through the colors. Generally you can create transfer function that will enhance shape features for specific models. However, definition of such transfer function is time consuming and usually also extremely difficult.

Shape information can be obtained by the changes in brightness along the surface. In case of shading methods, we assume that there is only one light source illuminating the image. However, the interaction between the illumination and the geometry, orientation etc. could cause the same sensation for different surfaces.

Shape cues from diffuse shading are minimal. On transparent surface the specular highlights can provide some cues to surface shape. But since they are view-dependent, it cannot provide e.g. relative depth cues. Specular highlights move slowly over highly curved regions, but on nearly planar areas they will move with moving viewer.

The illumination is usually supported by some motion mechanism or other interactivity. Also the occluding edges that delineate an object from background can have powerful influence on the perception of 3D shape.

Gooch's Shading

The special lighting model was proposed by Amy Gooch et. al [2]. This model uses a cool-to-warm color scale to allow shading to occur only in mid-tones so that edge lines and highlights would remain visible.

In this work, the blue and yellow tones are chosen. Some illumination variations are also added so that subtleties of shape could be visible. The proposed results could be seen in figure 1.4.

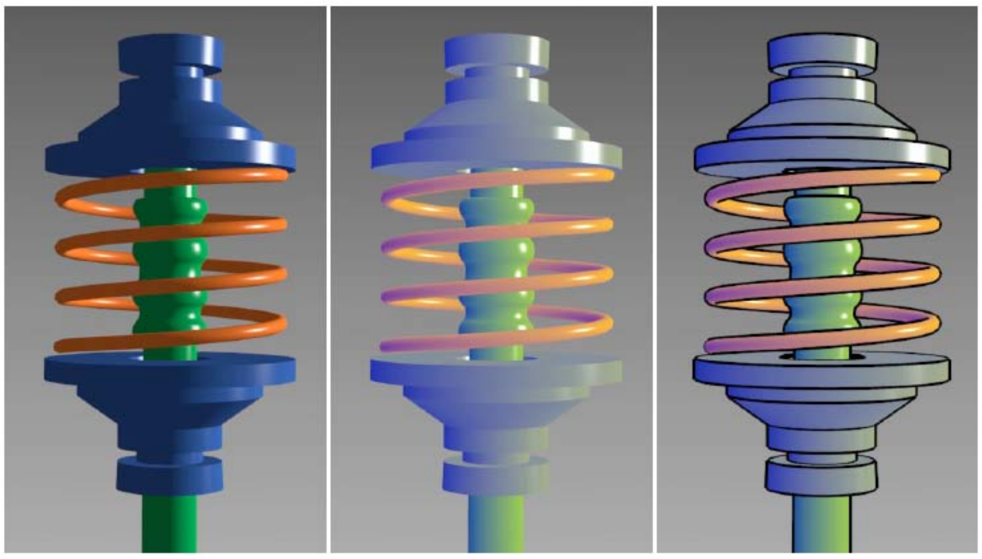
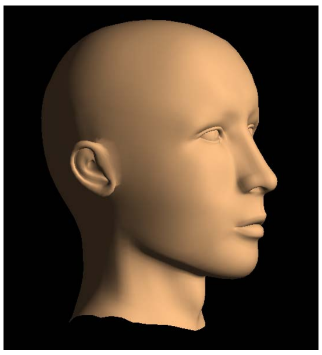


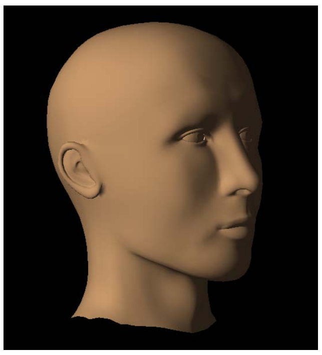
Figure 1.4: Left to right: a) Phong model for colored object. b) Gooch's shading model with added highlights. c) Gooch's shading model with highlights and edge lines. [2].

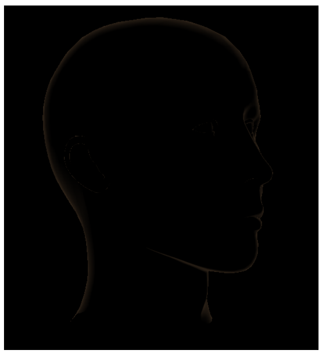
Note that Gooch's shading approach could be combined simply with silhouette/edges enhancement.

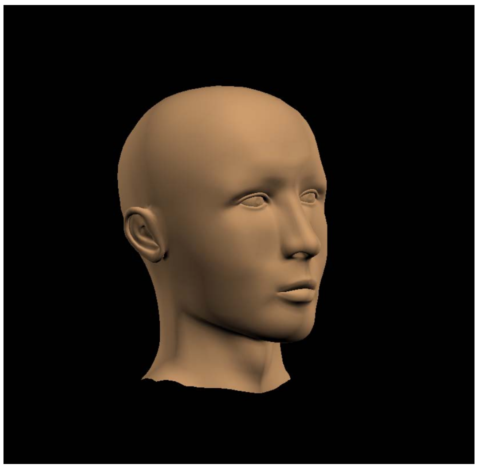
NPR Shading

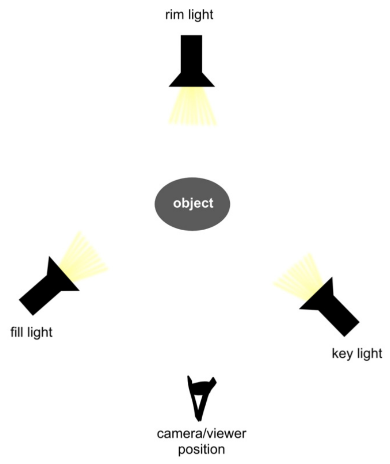
Another NPR lighting model is based on three-point lighting. The primary or key light creates dominant illumination and define the angle of it (figure 1.5a). The fill light softens the illumination. It simulates the secondary light or reflected light in the scene (figure 1.5b). Thanks to fill light, the parts away from light (in shadows) are also visible. The last light is the rim light. It comes from back of the object and it adds a defining edge to show where the object ends and background begins figure 1.5c.


(a) Primary/key light


(b) Fill light


(c) Rim light


(d) Adding all lights together



(e) Example of light positions.

Figure 1.5: Three-point lighting model.

1.2.1.2 Lines

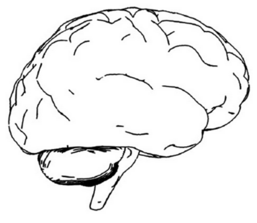
As was mentioned before, it is possible to enhance shape perception by lines. There are boundary lines that include silhouettes, contours, and feature lines like apparent ridges or crease lines. Generally, boundaries can get us info about sharp edges on objects. Feature lines works with higher order of curvature measures and they could give us better understanding of the shape. However feature line techniques could be less effective than shading in case we work with more complex objects.

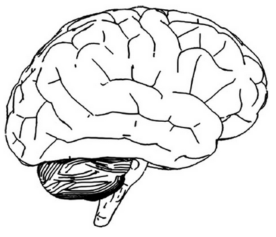
Boundaries

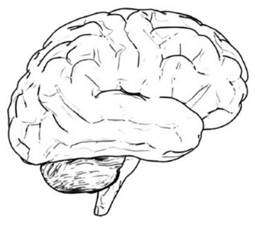
Silhouette and contour curves [3] are loci of points where the normal vector and the view vector are perpendicular

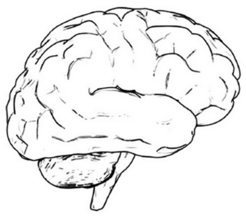
$$dot(\vec{n}, \vec{v}) = 0 \tag{1.7}$$

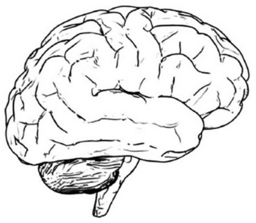
where $\vec{n}$ is the normal vector, $\vec{v}$ is the view vector (points to the camera). Silhouette curves form a closed outline around the object projection. Contour curves (in figure 1.6a) may be disjoint, but unfortunately they cannot depict salient regions. Thus, they are not appropriate to gain a spatial impression of the object.


(a) Contours


(b) Crease lines


(c) Ridge and valley lines


(d) PEL


(e) Apparent Ridges

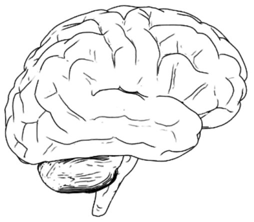

(f) Suggestive contours

Figure 1.6: Different line types on simple brain model for perception enhancement [3].

For the discrete case, edge is highlighted as contour if the sign of the dot product changes. However, this technique does not allow us to control width of the contour. By analyzing the surface curvature, using second-order derivatives, the contour thickness could be regulated [19].

Despite lesser accuracy, we could use better and more computationally effective approach. The concept of the style transfer function was proposed in [20]. The curvature of object surface is only approximated by changes of normal directions. The angle between two subsequent normals along the rate taken at sufficiently small steps gives us needed measure. Then the contour thickness could be controlled by the presented measures.

Feature Lines

A variety of lines exists to represent discontinuities in surface normal, curvature etc. Generally, two classes of feature lines exist [8].

View-independent feature lines These features are influenced by the shape of an object. From different views, they are the same. This class includes crease lines (in figure 1.6b). Crease lines are a set of edges, where an angle between the normals of the corresponding incident triangles exceeds a user-defined threshold [3]. As mentioned above, normal changes can measure magnitude of the curvature.

It also includes ridge and valley lines (in figure 1.6c). As was previously described, this method is also based on curvature computations. For ridge and valley lines we need to compute principle curvatures and associated principle curvature directions. Formally, ridges and valleys are defined as loci of points at which the principle curvatures assume an extremum in the principle direction [3]. With ridge and valley lines we are able to detect even small shape features. However, due to calculations which tend to propagate errors, this method is susceptible to noise.

View-dependent feature lines As class title suggests, these lines takes view direction into account. Suggestive contours (in figure 1.6f) are probably the most common lines that belong to this class. This method extends definition of contours. Suggestive contours may be seen as inflection points on the surface. Unfortunately, when using these contours, some sharp edges are not depicted.

Photic extremum lines (PEL, in figure 1.6d) are inspired by edge detection in image processing. This method is based on the magnitude of the light gradient. It uses variations of illumination. Similar to apparent ridge, PEL is also a third-order feature line, and thus computationally expensive.

Another type of lines was inspired by image processing. The introduction of Laplacian lines is based on the Laplacian-of-Gaussian edge detector. Laplacian lines calculate the Laplacian of illumination function and determine the zero crossing as feature lines. The lines are only drawn if the magnitude of the illumination gradient exceeds user-defined threshold [3]. Usage of Laplacian lines in interactive applications is possible [21], because the most expensive computations can be pre-processed.

Blueprints

I would also like to mention blueprints [4], an NPR technique that outlines visible and non-visible perceptually important edges. It also applies depth peeling to decompose 3D geometry. The blueprint method works in image space and is hardware-accelerated. The example of final blueprint is in figure 1.7.

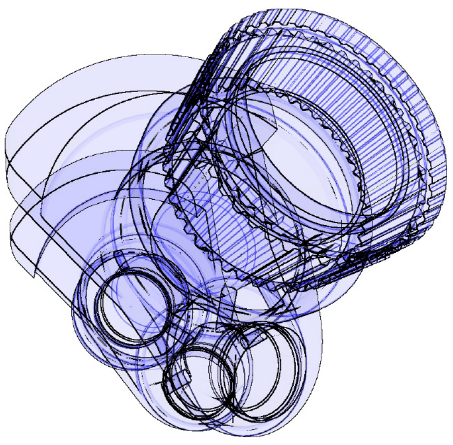


Figure 1.7: The blueprint of a crank [4].

Perceptually important edges include silhouette, border and crease edges. Blueprint technique extracts these edges by determining discontinuities in both the normal and z-buffer. For extracting discontinuities the sampling of neighbouring texels is used. These edges form a single texture referred to as edge map.

For each edge map a screen-aligned quad (SAQ) is textured with the edge map and the corresponding depth map. Then, the fragment program replaces fragment z-values by texture values derived from accessing of the depth map. The fragment then proceeds to the ordinary depth test. Note that the back clipping plane fragments are rejected. For depth cues, the intensity values derived from edge map are used as color blending factor. In order to improve rendering speed a desired minimal number of fragments to pass the depth test is specified, it decreases the number of rendering passes.

Possibly, some areas that could be important to understand the overall structure may be occluded. To ensure better visibility, the depth masking was presented. Depth masking provides a termination condition for blueprint rendering. An additional rendering pass is applied to generate a depth texture of designated components. If a specified fraction of fragments passes the ordinary depth test, the technique terminates. Otherwise, another depth layer must be peeled away.

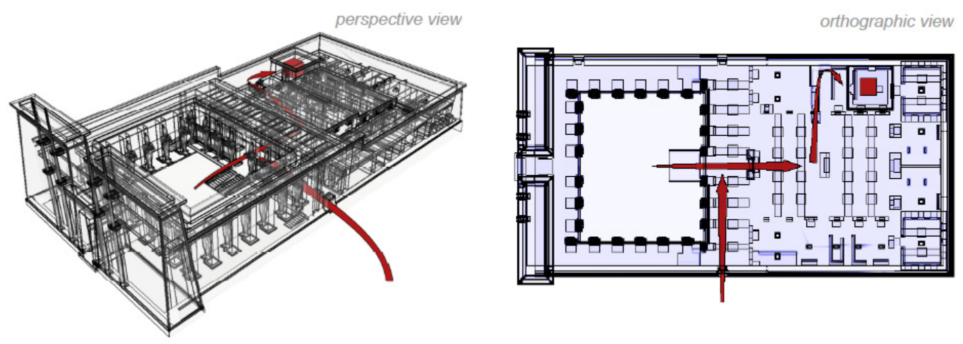


Figure 1.8: Perspective and orthographic views for the Temple to an upper chamber. The perspective view provides more a spatial outline while the orthographic view provedes more systematic insight [4].

It appears that orthographic views are typically more comprehensive than perspective views with increasing structural complexity. The comparation could be seen in figure 1.8 However, the perspective view still provides better spatial orientation and conceptual insight.

Phantom Lines

Phantom lines method is based on drawing only the outlines of transparent objects. These outlines are rendered with different style of line. This technique allows only two states – fully opaque or fully transparent objects – and it ignores all of the material and surface properties. Since we lose details of transparent objects, this method is mostly used for some types of technical illustration. The method is shown in figure 1.9.

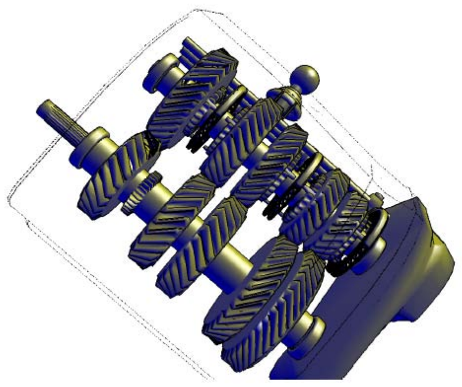


Figure 1.9: Phantom lines showing transparent object [5].

1.2.1.3 Illustrative Context-preserving Volume Rendering

One of the methods available in existing implementation uses shading intensity for opacity modulation function. Illustrative Context-Preserving Volume Rendering [6] is extension of both classical direct volume rendering and gradient-magnitude opacity modulation.

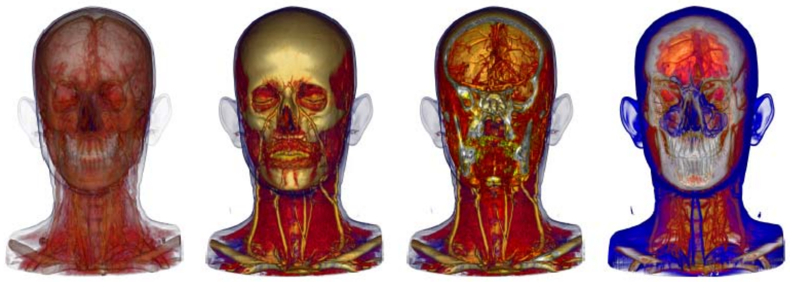


Figure 1.10: Left to right: a) Gradient-magnitude opacity-modulation. b) Direct volume rendering (DVR). c) DVR with cutting plane. d) Context-preserving volume rendering [6]

The function of shading intensity, gradient magnitude, distance to eye point, and previously accumulated opacity are used to selectively reducing the opacity in less important data region. This method was proposed as an alternative to traditional clipping methods. Since we want to focus mainly on transparency, I will present this method briefly.

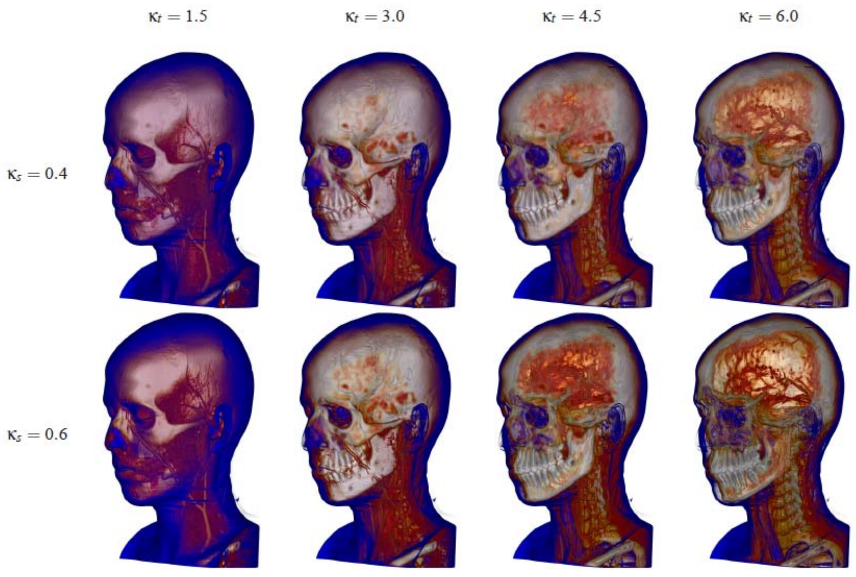


Figure 1.11: Context-preserving Volume Rendering with different setting of user-specified parameters [6]

The illumination usually has no influence on the opacity. The idea was to reduce the opacity in the regions where large area is highly illuminated. These regions normally correspond to rather flat surface. On the other hand, the contours would remain visible, since they received less lighting. The influence of this method is regulated by two user-specified parameters. One parameter corresponds to the depth of a clipping plane. The other parameter allows controlling the sharpness of the cut. The proposed results could be seen in figure 1.10. The influence of user-specified parameters are visible in figure 1.11

1.2.1.4 Conveying the 3D Shape via Texture

It is possible to use textures to convey the shape of smoothly curving transparent object. In [7], authors try to deal with rendering the transparent surfaces in such a way that their three-dimensional shape can be readily understood and their depth distance from underlying structures could be still clearly perceived.

To do so, they defined a principal direction texture that brings three important considerations - stroke direction, stroke placement and stroke length and width. They suggested a method for texturing transparent surfaces with a set of uniformly distributed short and opaque strokes. The strokes are locally oriented in the direction of greatest normal curvature and their length is proportional to the magnitude of the curvature in the stroke direction (in figure 1.12.

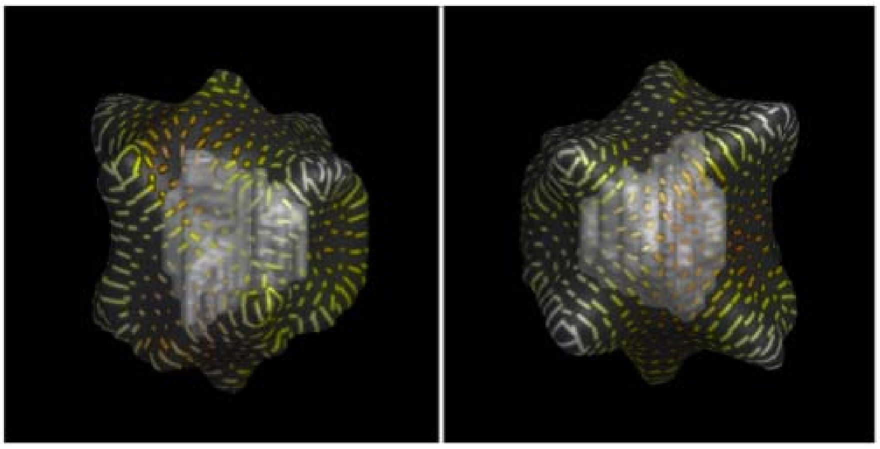


Figure 1.12: An example of using a texture to convey the 3D shape of the transparent object. In the left picture, the stroke length is proportional to the magnitude of the normal curvature in the stroke direction. In the right picture, the stroke length is constant [7].

1.2.2 Depth Perception

There are many ways to enhance depth perception. Depth cues could be categorized into two classes – monoscopic depth cues and stereoscopic depth cues [8].

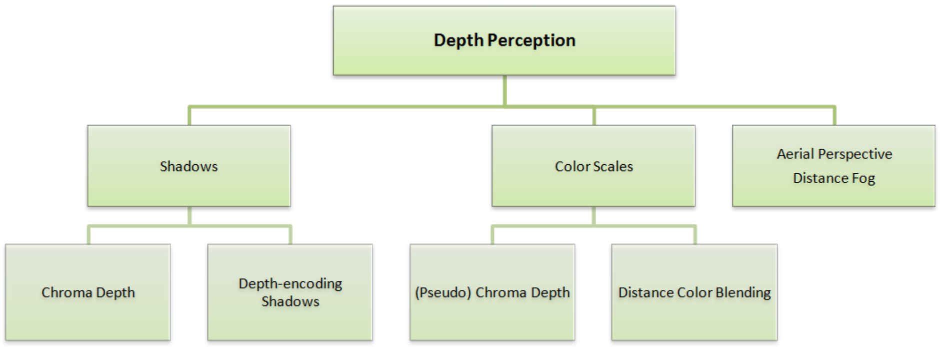


Figure 1.13: Depth perception techniques use shadows, depth-dependent color scales, or aerial perspective [8].

Monoscopic depth cues can be seen by a single eye. Many of them relate to motion of the object. The motion parallax exploits the image changes that occur when object moves relatively to the observer. Another motion-based approach is called kinetic depth effect. This effect is strongly connected to observing of the shape from motion.

The distance fog or aerial perspective is another cue that belongs to this class. It is a term for situation when the foreground has higher contrast and the background has lower contrast. The color of distant objects is usually also shifted to the blue end of color spectrum. Also the lighting and shading has more influence on depth perception. Since we use a monoscopic display, the monoscopic visual cues will be the cues that we will deal with.

Binocular depth cues imply the fact that two eyes have slightly different views. For example binocular disparity or convergence belong to this class of cues. The binocular disparity is based on two images of the same scene obtained from slightly different angles.

Shadows have naturally an important role in spatial perception. Also techniques like silhouette enhancement or tone shading can be useful to support depth perception. These techniques were described in section 1.2.1. Note that depth perception is usually closely connected with the perception of depth ordering.

1.2.2.1 Chroma Depth, Pseudo Chroma Depth, Distance Color Blending

Since the refraction of blue wavelength light in the eye can result in illusion that a blue object seems to be further away than red objects, we could use chroma depth to enhance depth perception without glasses. For shadows, based on chroma depth, the blue color is added to simulate the shadow.

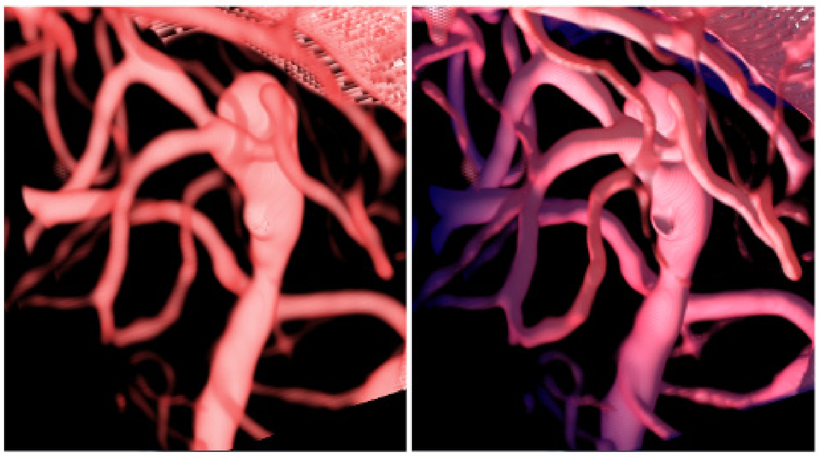


Figure 1.14: The left image shows a standard volume rendering of the vessels. The right image shows enhanced visualization with distance color blending where the vessels that are farther away are colored in a shade of blue (could be referred as pseudo chroma depth) [9].

Pseudo chroma depth refers to using only one transition color for blending between proximal and distal parts. Chroma depth uses more hues than pseudo chroma depth and should therefore allow to measure differences in depth better. In spite of this fact, the pseudo chroma depth produces slightly better results in general. In figure 1.14 you can see the example where blue color is used to visualize the vessels that are farther away.

The chromadepth-based technique can be simply combined with contours, halos, shading etc. It could be also used for depth enhancing for MIP rendering approaches.

1.2.2.2 Distance Fog or Aerial Perspective

As was mentioned before, distance fog or aerial perspective is also a possibility for enhancing the depth perception. Generally, aerial perspective is the key element that significantly affects the depth perception. The greater the distance between object and viewer, the greater amount of atmosphere overlaps the object. Accordingly, the object takes over atmosphere's features like color, hue etc. The example of such behavior can be found in figure 1.15. Therefore if we consider more than one non-opaque overlapping layers, the objects behind these layers seem to be less detailed and partially take the color of layers in front of them.

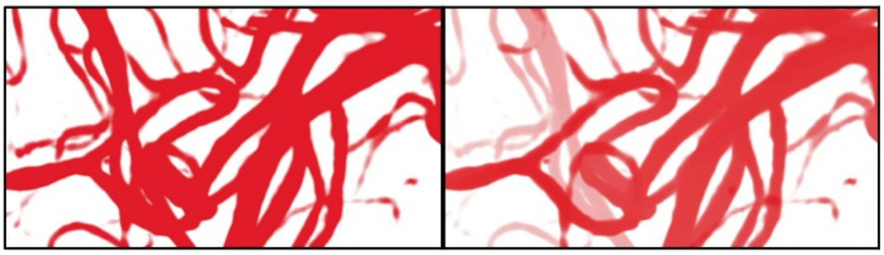


Figure 1.15: The left image shows vascular volume visualization with no special cues. In the right image, the aerial perspective/fog is used for better perception of depth [10].

The intensity of this effect can be related to the overall depth of the scene or to the relative depth between samples along each ray and it could be increasing linearly or exponentially. Exponential fog is more physical based model. Light intensity is reduced over the distance due to absorption and scattering from air particles and it decreases exponentially with the distance from the object.

1.2.2.3 Depth-encoding Shadows

For depth-encoding shadows technique, the stripe texture was used in [11] to illustrate distances between branches of vascular structures. The distance to the observer is encoding to stroke width of a procedurally generated stripe

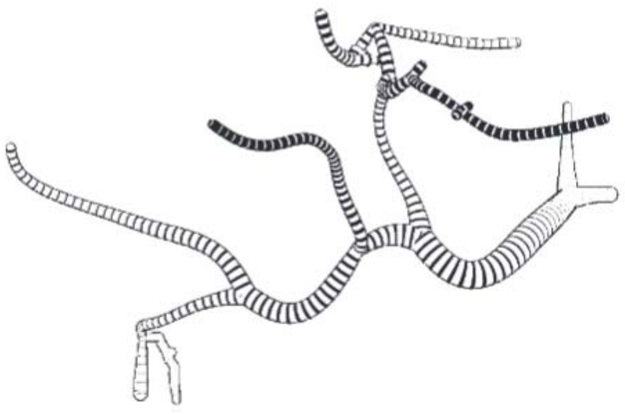


Figure 1.16: Generated stripe texture for depth encoding [11].

texture. Thicker black stroke indicate a shorter distance to the viewer. The result of this method is shown in figure 1.16.

1.2.3 Perception of Depth Ordering

Generally, any occlusion is the best indicator of depth ordering. With transparency, in case an object is occluded by another object, the X and T junctions can be beneficial to the perception of depth ordering. X junctions, where the borders of two objects cross in an 'X' fashion, are most informative regarding the perception of transparency. The best depth-ordering cue is for fully opaque surfaces as in the case of a T-junction. Thus, the strength of depth-ordering cue is directly proportional to the degree of contrast reduction.

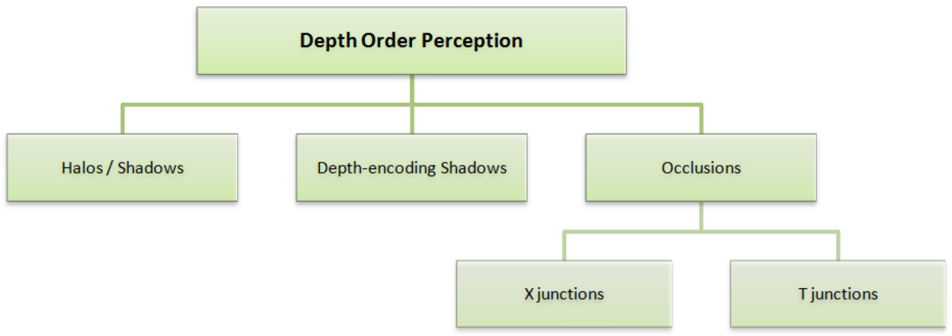


Figure 1.17: Depth order perception techniques use mainly occlusions and halos.

To overcome the limitation of X-junctions, the researchers proposed to use Transmittance Anchoring Principle (TAP). According to TAP, regions with highest contrast are perceived to be in front.

Shadows, luminance, contrast and colors are also key elements for good perception of depth ordering. The same applies to halos and silhouette enhancement. The overview of main depth order enhancement techniques is shown in figure 1.17.

1.2.3.1 Volumetric Halos

Shadows are produced when occluding structures decrease the amount of illumination on adjacent objects. On the other hand, the halos are structures that shine on adjacent object. The foreground features could be emphasized with bright halos. When the halo color is dark, hallos are slightly like shadows.

Volumetric halos method was introduced in [22]. It presents a flexible method for enhancing and highlighting regions of interest. The halo contribution is determined during volume rendering. The algorithm operates on view-aligned slices in front-to-back order. The algorithm pipeline is executed for every slice and it consists of 3 phases and additional step for blending.

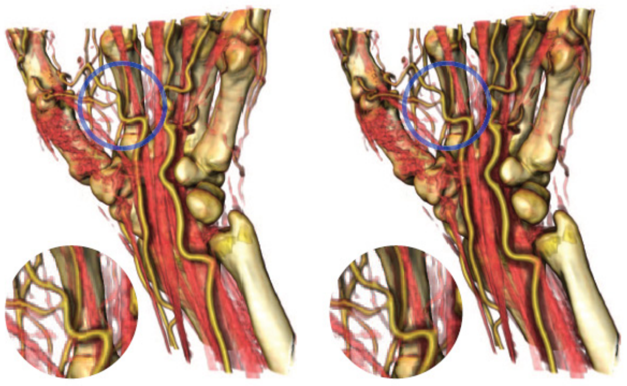


Figure 1.18: On the left there are the directional occlusive halos that are visible only when they occludes other structures. On the right the unidirectional occlusive halos are shown [22].

First phase is called halo seeding. During this phase, seed intensity value is generated for all samples on slice. Every point with nonzero intensity is a seed point. To limit seeds to regions around the contour, the angle between the view vector and gradient vector is computed. If these vectors are nearly orthogonal, the sample point is on contour. The next phase is halo generation. For a smooth halo we need to spread halo effect from seed to its neighbouring points. This leads to iterative process where for each step the neighbouring voxels are set as a seed point with intensity values decreased respectively to original seed point.

After generating the halo field image it has to be mapped to visual contribution in the image by halo profile function. It maps halo intensity values to colors and opacities. Based on how the combination is performed, two different kinds of halos can be specified. Emissive halos behave as if they were part of the volume. The halo intensity value is firstly mapped using the halo profile function and then blended after the actual volume contributions. These halos therefore partially occlude everything located behind them.

Occlusive halos (in figure 1.18), on the other hand, only contribute to the image. Although this halo type is similar to shadows, halos are usually drawn in the same style irrespective of the distance between two objects.

1.2.3.2 Smart transparency based on XT junctions

The transparency enhancement method that enables good recognition of depth ordering is proposed in [23]. For this technique three transparency fields on the surface should be defined – one for the base transparency, one for the silhouette enhancement and one for the halos. The values for the two latter fields are fixed on the silhouette. A diffusion process is used to spread effects along the silhouette. Finally, all three fields are combined and the final image is created with alpha blending.

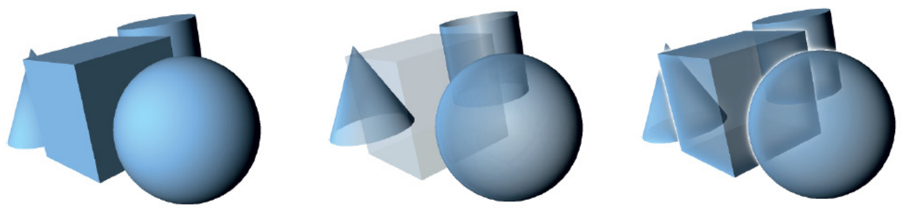


Figure 1.19: Comparation of different transpareny – opaque rendering (left), normal variation transparency (middle) and smart transparency (right)[23].

The 2.5D space representation is used and the solution is based on illustration buffer. The illustration buffer stores linked list of all surface layers for each pixel. Each fragment stores an arbitrary number of surface properties for deferred computation, as well as explicit links to neighbouring fragments. The illustration buffer is implemented on GPU and it combines the ideas of the A-buffer and G-buffers. In this approach, each pixel stores a linked list of fragments. After the surface polygons are rasterized, each fragment is added into the respective list. Finally, each list is sorted by distance to the viewer and processed to obtain the final pixel color.

As you can see in figure 1.19, this method properly highlights all silhouettes and maintains high contrast while the transparency for all layers is used.

1.2.3.3 Combination of X-junctions and TAP

Firstly, the X-junction model is used to estimate the perceived depth ordering. If this model fails, i.e. depth ambiguity occurs, the TAP model is used to deal with ambiguity situation. Since TAP depends on contrast perception, it can help us to decide depth ordering [12]. Suggested enhancement is shown in 1.20.

1.2.3.4 Depth-encoding Shadows at Intersections

This technique, described in [11], is version of distance-encoded shadows mentioned above. The size of the drawn shadow at an intersection in figure 1.21

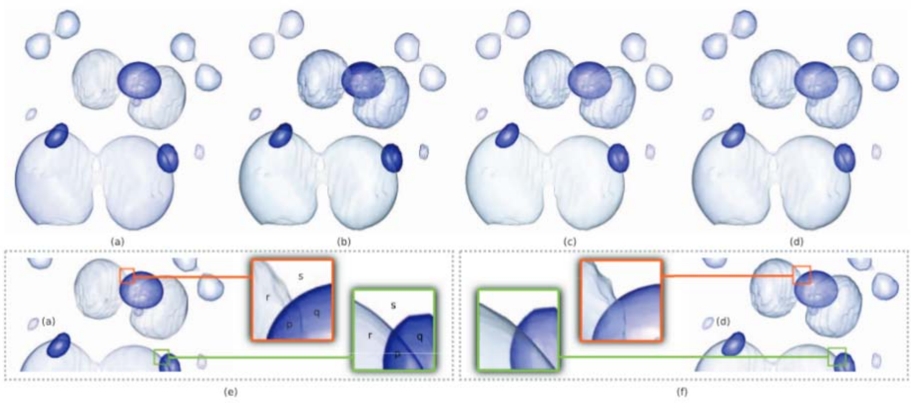


Figure 1.20: (a) An original image in which the depth relations between structures are misleading. An image enhanced by: (b) the depth ordering energy (c) both the depth ordering and transparency energy (d) the depth ordering, transparency,and image faithfulness energy. (e) Illustration of two X-Junctions in (a). (f) Illustration of two selected X-Junctions in (d). [12]

corresponds directly to the depth distance betwee the two overlapping segments.

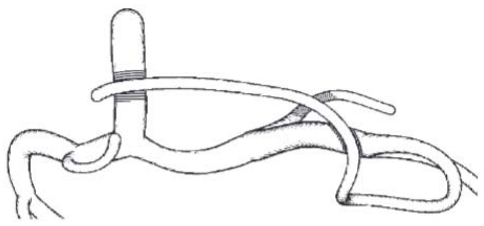


Figure 1.21: Example of adaptive line width hatched shadow based on depth-encoding shadows [11].

1.2.3.5 View-dependent transparency model

Authors of [5] extracted set of rules from traditional methods in manual drawings. Firstly, faces of transparent objects never shine through. Secondly, opaque objects which are occluded by two transparent objects do not shine through. Finally, transparency falls off close to the edges of transparent objects and increases with the distance to edges.

Based on these rules, the objects which are blended, have to be determined following the first two rules. This task may be solved by depth sorting proposed in [5]. Transparency values have to be computed for correct blending between objects according to the third rule. For this task concept of view-dependent transparency was introduced.

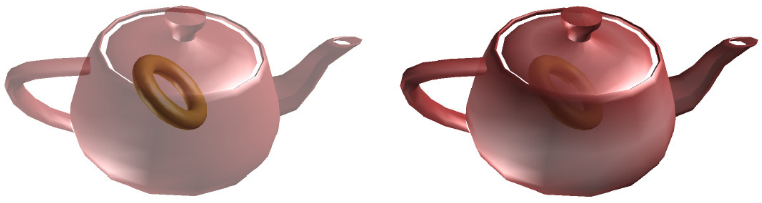


Figure 1.22: Difference between standard transparency blending (left) and view-dependent transparency blending (right) [5].

In view-dependent transparency, silhouettes are determined as edges connecting two faces, where one is back facing and the other one is front facing. Detected silhouettes are stored in a list. Then the distance of each vertex to closest silhouette is computed and used to calculate the alpha value.

This approach does not need complete depth sorting. Only the closest transparent objects and the opaque objects directly behind these transparent surfaces need to be determined.

Note that this method can also be combined with other techniques described above, such as tone shading or silhouette enhancing. The result of this combination is shown in figure 1.23.

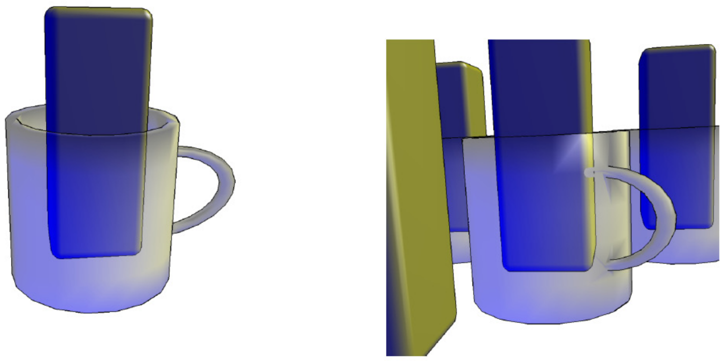


Figure 1.23: Proposed view-dependent transparency model combined with Gooch's shading and silhouettes enhancement in order to add better shape perception [5].

1.2.4 Methods Suitable for Combination of Different Visual Cues

There are multiple criteria that are possible to use for the opacity modulation. Some of the visual cues are easy to combine like for example a shading with specular highlights. Others are more difficult since they are usually contradicting. We want to find methods that combine those visual cues to a single alpha value which will correspond to both. Firstly, I will mention some methods based on a combination of different styles or techniques. Secondly, I will present the possible logic operators that could be used to aggregate opacity modulation of different visual cues.

Two-level Volume Rendering

For combining different styles or techniques some approaches do exist. First I would like to mention Two-level Volume Rendering [13], which allow us to use different rendering techniques. Calculations are decomposed to two levels – local and global rendering. For every pixel, they theoretically investigate a viewing ray through the data and detect what objects are intersected. For every intersected object, a single value is computed (local rendering) and composed to a pixel value with others along the ray (global rendering).

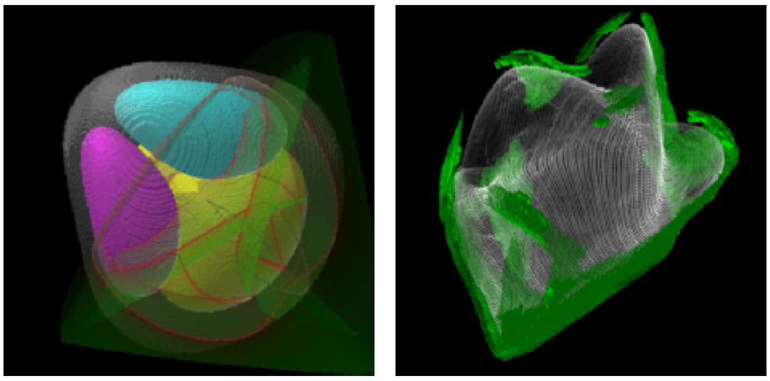


Figure 1.24: Dynamic systems visualized by the use of two-level volume rendering [13].

Semantic Layers

Another approach that allows us to combine styles is using of semantic layers [14]. This approach uses fuzzy logic and takes multiple volume attributes as an input and evaluates a set of rules to determine which style would be applied to the current sample. Both attributes and styles are described by semantic values and by its membership functions. Description of the desired mapping is referred to as rule base. The premise is a logical combination of semantic values of the volume attributes and the conclusion is a list of styles that are affected by the rule. The influence on each style is aggregated and then defuzzyfied. In order to style application, the style needs to be parameterized (color scale, line width etc.). It was proposed to use orthogonal styles, i.e. these that do not influence each other. Styles can be applied incrementally or selectively. The incremental application resembles the work of an illustrator. First a basic style is applied, on top of it a shading, contours and specular highlights are drawn. In the selective application, each semantic layer defines one style according to a set of rules. Then, each layer is applied to specific regions according to its style volume. Note that a completely opaque style overdraws all styles that are lower in hierarchy.

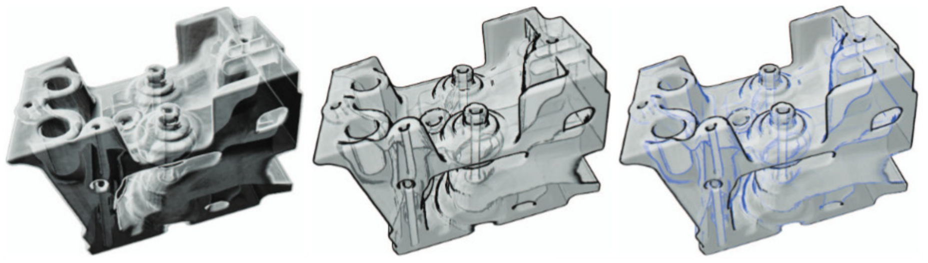


Figure 1.25: The curvature based approach with semantic layers. On the left the cartoonish shading effect is presented. In the middle the cartoonish shading is reduced and countours are added. In the right image a rule is used to selectively apply the contours. Convex regions are shown in black, whereas concave regions are shown with blueish contours [14].

Logic Operators

If we want to aggregate the effect of properties on transparency, we need to somehow combine these parameters. For example, we could evaluate each sample according to its relative distance from previous layer (in the direction from the viewer) and according to its normal vector (curvature of the surface). In order to combine these parameters, we could use some logic operators.

Firstly, I would mention the probabilistic sum. If we consider property A and property B, we could combine them by probabilistic sum as shown in equation 1.8.

$$R = A + B - A * B \tag{1.8}$$

If we assume that both A and B are from interval [0,1], the result R will also be in interval [0,1]. The equation 1.8 refers to fuzzy logic operator OR. Consider the following situation. The result parameter is sample opacity in the interval [0,1]. Parameter A is depth measurement, parameter B is curvature measurement, both in the interval [0,1]. If we use the probabilistic sum to combine parameters A and B, it only needs one parameter to be 1 to get a fully opaque sample in the result.

Secondly, the maximum of two values could be used as a result (equation 1.9). If we refer to the previous case, we only get a fully opaque sample if one of these parameters is 1.

$$R = max(A, B) \tag{1.9}$$

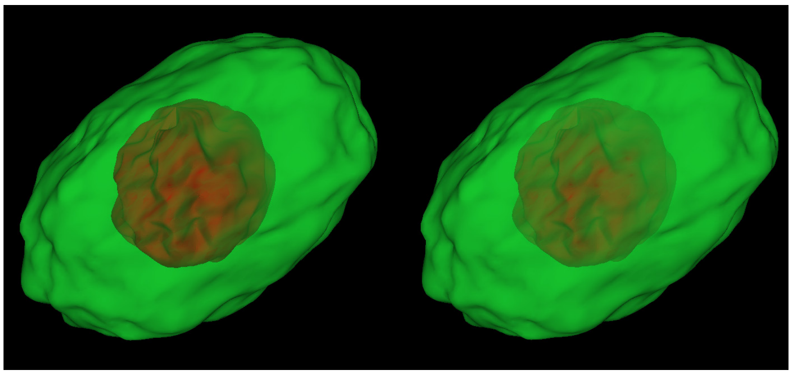


Figure 1.26: Example of alpha value using shape curvature and depth opacity modulation. In the left figure, the maximum operator is used. In this case, the shape factor dominates over depth. In the right figure, the probabilistic sum is used.

Finally, the weighted sum could be used to combine the influence of specific measurements. If we map the result to opacity, we need to scale values into the interval [0,1] to get proper visual results. The weighted sum is shown in following equation (1.10)

$$R = \sum_{i=1}^{n} w_i * x_i \tag{1.10}$$

where R is result value (not scaled), n is number of combined parameters, w_i is weight of parameter i and x_i is value of parameter i.

There are many logic parameters that could be used to combine the parameters. It only depends on what we want to achieve.

Chapter 2

Design and Discussion for Experiments

Firstly, I would like to describe an overview of applications that should be implemented in the scope of this thesis. First of all, we will need an application that will be able to display 3D model with polygonal representation. The simple GUI should be provided in this application. We need to implement several visual cue enhancement methods that were described in chapter 1. Visual cues should be possible to turn on/off from the GUI. In addition, it should be possible to achieve a combination of desired visual cues.

Secondly, the user should be able to export the images for perceptual user study from the application. The scene definition and test setup should be also exported using the application.

Thirdly, we need to create a test application, that should be able to work with exported data and collect data from participants.

At the beginning of my work, my thesis supervisor provided me the application that contained the modified depth peeling algorithm that is using two layers in each iteration. As was mentioned before, it enables us to modulate opacity based on the relative distance between two layers. I will refer to this application as to the existing implementation. The technologies used in implementation will be described later in Chapter 3.

In this chapter, the necessary changes in the existing implementation will be listed. Then, visual cues that I implemented will be described and discussed.

I will implement several visualization enhancement methods, test them on prototype models. Thus I will be able to evaluate which visual cues truly works in case of semi-transparent objects. Then I will use a combination of those visual cues. Finally the experiment with users will be designed. The experiment will use several levels of opacity/transparency. The experiment is thoroughly described in Chapter 5.

2.1 Solving existing issues

Firstly, we needed to solve some issues in existing implementation. There were two implementation tasks that should be resolved:

- solving the problem with opacity modulation based on distance without using selective transparency
- changing last render pass to blend final image with background

Both of these tasks should be created as an extension of existing implementation based on multi-pass rendering. Detailed information of the implementation could be found in [17].

2.1.1 Opacity modulation based on distance

There are two ways to consider distance for opacity modulation in existing implementation. Cmolik [17] used the approach that modulates opacity based on distance to the next sample along the ray. Note that if we take into consideration only distance between samples, it is not necessary to modulate opacity of last sample along the ray. On the other hand, Bruckner [18] used the distance from viewer.

If the selective transparency is used, we set the last layer, i.e. the layer that has no other layer behind it, as fully opaque. Since we want to keep the idea used by Cmolik [17], we won't modulate the last sample opacity. We compute following difference for every sample s

$$s_{next}.depth - s.depth \tag{2.1}$$

where s is current sample and s_{next} is sample from next layer. Result of this difference falls within the range [0,1].

In case the selective transparency is not used, negative values of the difference in formula 2.1 can be obtained by existing implementation. The solution to this problem is clamping the result to the required range or setting it to 0 for other computations, if $s_{next}.depth$ is equal to zero.

The comparison of results is shown in figure 2.1. Each row shows the same situation with (left) and without (right) selective transparency. In first row, smaller factor for opacity modulation based on sample distance is used.

2.1.2 Blending with background

The existing solution for placing the background color was based on finding areas where no geometry is rendered. These areas are determined by bitmasks. These operations are executed as last render pass to compose the final image (in case of labeling [17] there are still several passes to be processed).

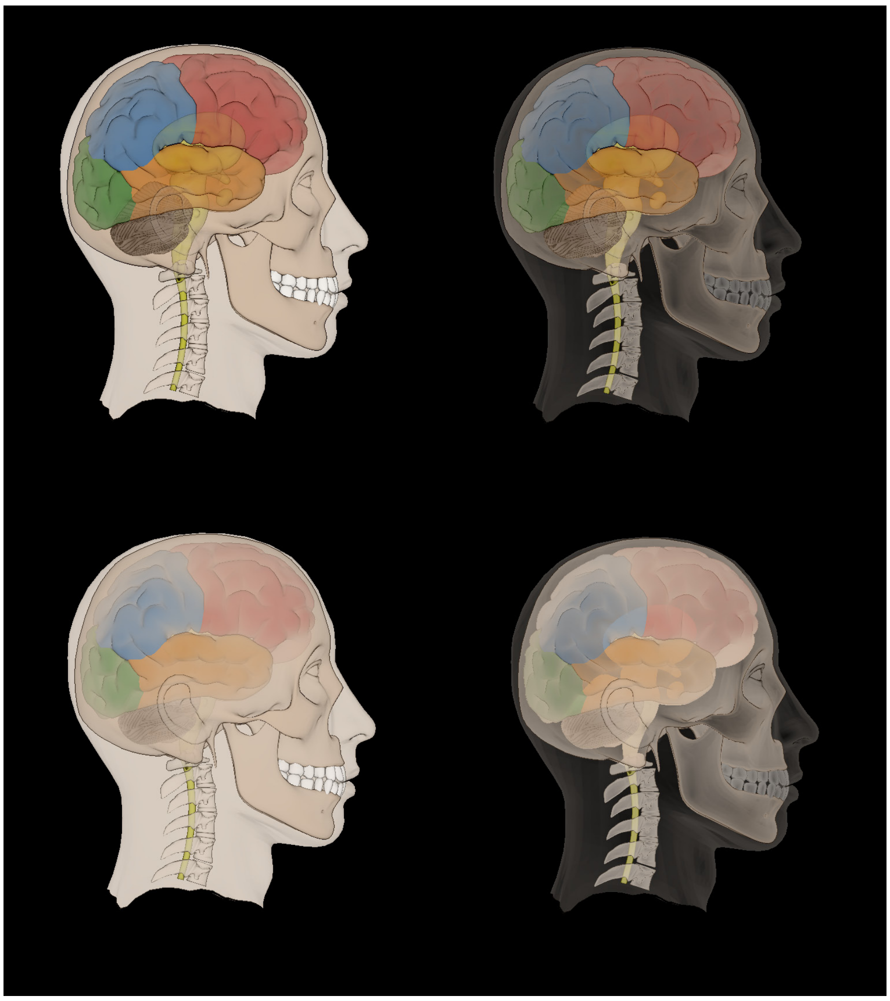


Figure 2.1: On the left is opaque modulation by distance between samples with using selective transparency. On the right side is the same situation without selective transparency. Second row has increased distance factor for modulation.

I do not need to preserve exactly the same approach to blend final image with background. In existing pass I only add another composition using under operator defined in section 1.1.2.4. When no selective transparency is used, the blending is applied. Thus, for each fragment on position (x, y) the color is established as

$$\vec{c}_{(x,y)} = \vec{c}_{t(x,y)} \; + \; \alpha_{t(x,y)} \; * \; \vec{c}_{background} \tag{2.2}$$

where $\vec{c}_{(x,y)}$ is final fragment color, $\vec{c}_{t(x,y)}$ is color from texture, i.e. color computed from all previously blended layers, and $\alpha_{t(x,y)}$ is alpha from texture.

The alpha value is equivalent to accumulated fragment transparency from previous passes. Finally $\vec{c}_{background}$ represents color of our background.

2.2 Depth Enhancement Methods and Depth Ordering

If we consider depth and depth ordering enhancement methods, these factors goes hand in hand with each other. Although these are different tasks, the methods that can be tested are more or less the same.

2.2.1 Aerial Attenuation/Distance Fog

In the existing implementation, there was possible to modulate opacity based on depth. It was already mentioned in section 2.1.1. During the process, I have changed the model to an exponential one which is described in equation 2.3

$$dist = max(1.0 - e^{(-distancePower*space)}, 0.0) \tag{2.3}$$

This approach seems to be more suitable for optical properties of the material. The result of the exponential model is presented in figure 2.2

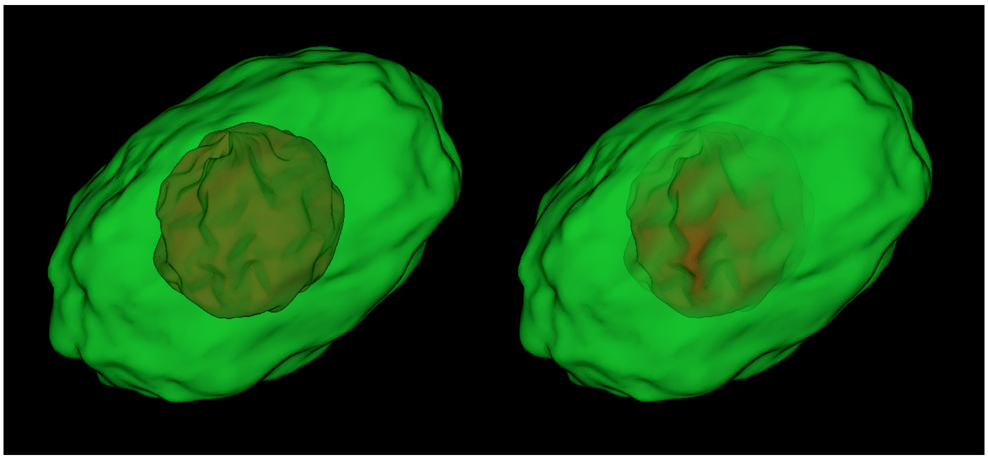


Figure 2.2: Influence of distance fog on semitransparent object. On the left model no distance fog is used. On the right model the opacity modulation based on depth and equation 2.3 is used. As you can see, it is clearly visible which parts of inner (red) object is closer to the outer surface (green object).

Since we modulate only the surface opacity, this method seems to be perspective for depth perception experiments.

2.2.2 Encoding Depth to Color

Since I wanted to use the existing approach of depth peeling and relative distance between the samples, I decided to try chromadepth-like coloring of the models. I am using warm-to-cold color scale, where yellow represents warm tones and blue cold tones. Firstly, the existing approach based on curvature and distance between samples is used to modulate opacity. Secondly, fragment

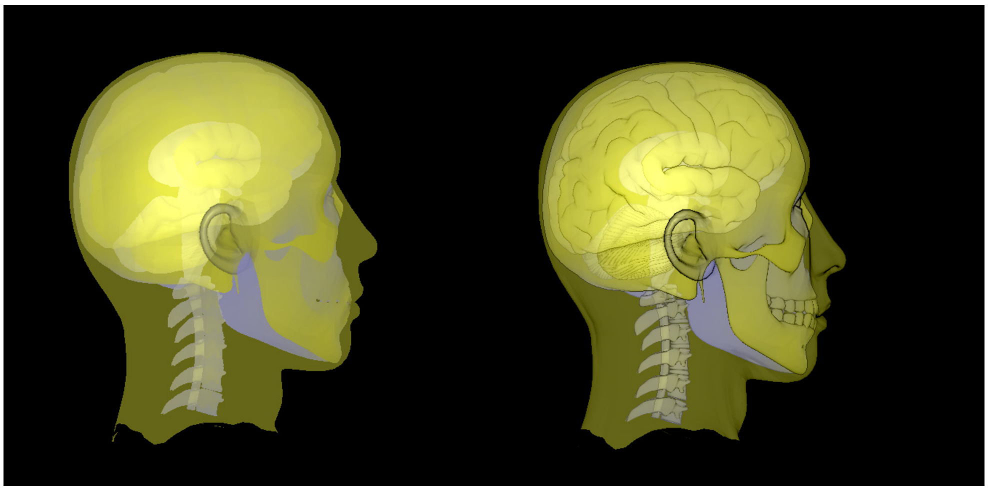


Figure 2.3: Simple chromadepth-like coloring (left) and enhancing the perception of shape with edges (right).

color is composed by equation 2.4.

$$c_{\vec{frag}} = c_{\vec{front}} * z_{frag} + dist * c_{\vec{back}} - \vec{curvature} \tag{2.4}$$

where $c_{\vec{frag}}$ is final fragment color, z_{frag} is the normalized z-coordinate of fragment, *veccurvature* represents curvature measurement. Finally, $c_{\vec{front}}$ and $c_{\vec{back}}$ are color vectors for distant and nearby regions. You can increase or decrease the blue effect by changing the distance influence parameter. The *curvature* was used mainly for shape perception enhancement. You can see the difference in figure 2.3.

The color-depth encoding method does not seem to be suitable for this task. There is a severe problem with semi-transparency of layers. When colors of layers are mixed according to the alpha of closer layers, it loses the conceptual benefits of this method.

2.3 Shape Enhancement Methods

To reveal the surface of objects in rendering we use shading that could help us distinguish possible curvature and roundings of surfaces. Our visual system tries to interpret the brightness pattern as shading due to spatial changes on the surface and its variations in reflecting properties.

The most common shading method is Phong shading. However, the Phong-shaded surface does not provide the same geometric information than for example human-drawn illustrations. But shading properties could be more or less tuned. We can try some methods to do that.

2.3.1 Gooch's Shading

Gooch's shading is the special lighting model that was proposed by Amy Gootch et. al and it was presented in section 1.2.1.1. The implementation is shown in listing 2.1.

```
void setGoochShading(float al)
{
    gl_FragColor.xyz = vec3 (0.0, 0.0, 0.0);

    vec3 N = vec3(real(texture2D(depth, texCoord).xyz));
    float NdotL = dot(L,N);
    float interpolationFactor = (1 + NdotL) * 0.5;

    // set colors
    vec3 warmColor = vec3(y, y, 0.0);
    vec3 coolColor = vec3(0.0, 0.0, b);
    vec3 objectColor = texture2D(color, texCoord).xyz;

    //combine colors
    vec3 coolColorMod = coolColor + alpha * vec3(objectColor);
    vec3 warmColorMod = warmColor + beta * vec3(objectColor);

    vec3 I = mix(coolColorMod, warmColorMod, interpolationFactor);
    gl_FragColor.xyz = I * (1 - al);
}
```

Listing 2.1: Gooch's Shading

Method for Gooch's shading computation gets computed alpha as an input parameter. The alpha is computed as a combination of shape based opacity and depth based opacity. It allows us to combine Gooch's shading with previously presented opacity modulation techniques. The `real` function changes normal vector from texture to [-1, 1] range.

Vector L that is pointing to light is set as static. The cosine term (the dot product of L and N) is used to blend between two colors - yellow and blue. Interpolation factor is computed based on equation 2.5 described in [2].

$$I = \left(\frac{1 + dot(\vec{l}, \vec{n})}{2} \right) * k_{cool} + \left(1 \quad \frac{1 + dot(\vec{l}, \vec{n})}{2} \right) * k_{warm} \tag{2.5}$$

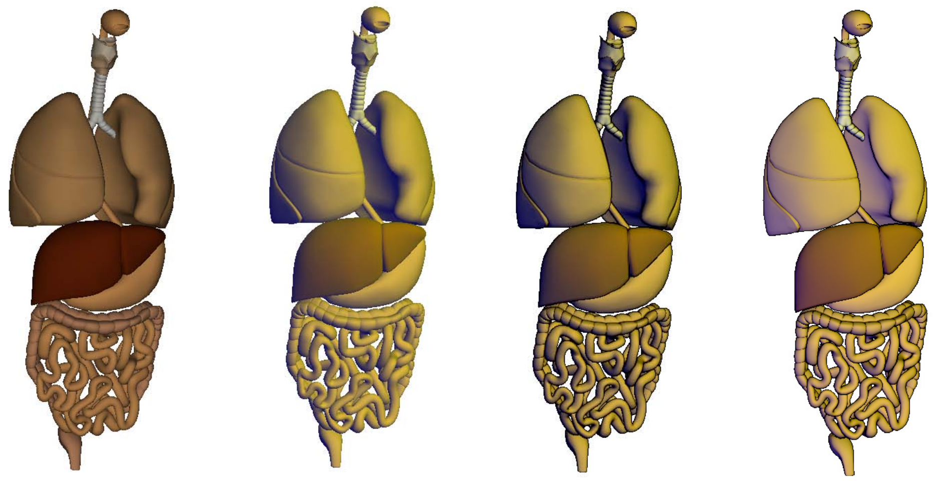


Figure 2.4: At first model from the left, you can see the object without hue shift effect. The second and third ones using cool-to-warm hue shift. The third one is also using edges. The last one represents much smaller hue shift. You can see that it does not cover any possible detail in darker parts.

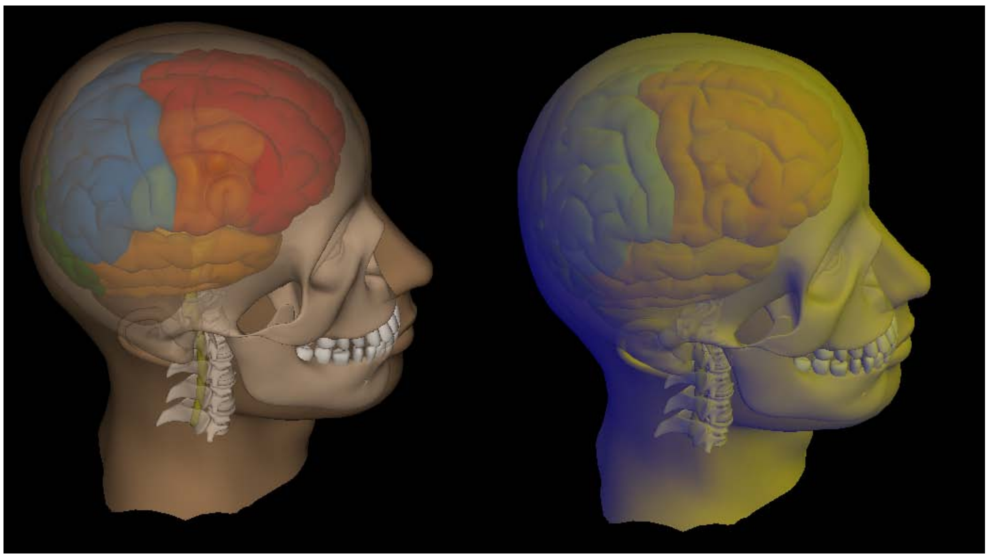


Figure 2.5: Gooch's shading with transparency has no added value. Even if some parts could reveal better shape clues, others are not even visible.

The intensity of the warm color (`y` parameter) and the cool color (`b` parameter) can be set from GUI. Also, the `alpha` and `beta` parameters are input from the GUI. They define how much blending will be used between cold/warm color and color of the object.

Despite the fact that Gootch's shading is truly enhancing the perception of shape, as you can see in figure 2.4, it turns out that it only works for the opaque object. When transparency is used, the blending of colors of overlapping layers are mixed together and the enhancement by colors lost its importance (figure 2.5).

2.3.2 NPR Shading

Another method that should help to better shape perception is NPR Shading which I mentioned in section 1.2.1.1. It based on three-point light composing to reveal details in parts that could be unlit if we use only single light. The implementation is shown in listing 2.2.

Similarly as in previously described lighting method, also NPR lighting method gets a parameter that represents the opacity calculated from shape based opacity and depth based opacity. The opacity is then modulated by the curvature of the object, as is shown in figure 2.6 and it could be combined with aerial attenuation based on the distance between two layers.

```
void setNPRShading(float al)
{
    gl_FragColor.xyz = vec3(0.0, 0.0, 0.0);
    vec3 objectColor = texture2D(color, texCoord).xyz;
    vec3 N = vec3(real(texture2D(depth, texCoord).xyz));
    float NdotL1 = dot(normalize(L1),N);
    float NdotL2 = dot(normalize(L2),N);
    float NdotL3 = dot(normalize(L3),N);
    // ambient component
    gl_FragColor.xyz += objectColor * vec3(0.3, 0.3, 0.3);
    // diffuse components
    gl_FragColor.xyz += objectColor * (1 - al) * max(NdotL1, 0.0);
    gl_FragColor.xyz += objectColor * (1 - al) * max(NdotL2, 0.0);
    gl_FragColor.xyz += objectColor * (1 - al) * NdotL3;

    // add specular highligths from primary light
    if (specularHighlights == 1) {
        vec3 R = 2 * dot(normalize(L1),N) * N - normalize(L1);
        if(NdotL1 > 0.0) {
            gl_FragColor.xyz += lightColor * 0.5
                * pow(max(0.0, dot(N, R)), specularPower);
        }
    }
}
```

Listing 2.2: NPR Shading

Adding lights to specific position helps us to control or eliminate the shading and shadows produced by direct lighting. This method is suitable for further investigation in our experiments since it produced desired visual cues.

Furthermore, I tried to modify the shape opacity modulation not to be based on the curvature of the surface but on the intensity of light. If we use single light pointing from the camera position, we can get dark parts at the outer edge of the object. The curvature based on light intensity is calculated by equation 2.6:

$$curvature = (1 - abs(dot(\vec{N}, \vec{L}))) \tag{2.6}$$

where $\vec{N}$ is normal vector in specified point and $\vec{L}$ is light direction. The single light intensity modulation is presented in figure 2.7. The opacity modulation based on the single light intensity has already been implemented in the existing implementation.

The three-light model can be also used to opacity modulation based on the light intensity. I combined the contributions of the intensity of individual lights according to listing 2.3 where I prefer primary light over the rim and fill lights (figure 2.7).

```
float intensityCurvature(vec3 N)
{
        // rim light intensity
        float a = (1.0 - abs(dot(N, normalize(L3))));
        // fill light intensity
        float b = (1.0 - abs(dot(N, normalize(L2))));
        // primary light intensity
        float c = (1.0 - abs(dot(N, normalize(L1))));

        return (max(c, 0) * 0.55 + max(b, 0) * 0.2 + a * 0.15);
}
```

Listing 2.3: Opacity modulation based on intensity of three-light model

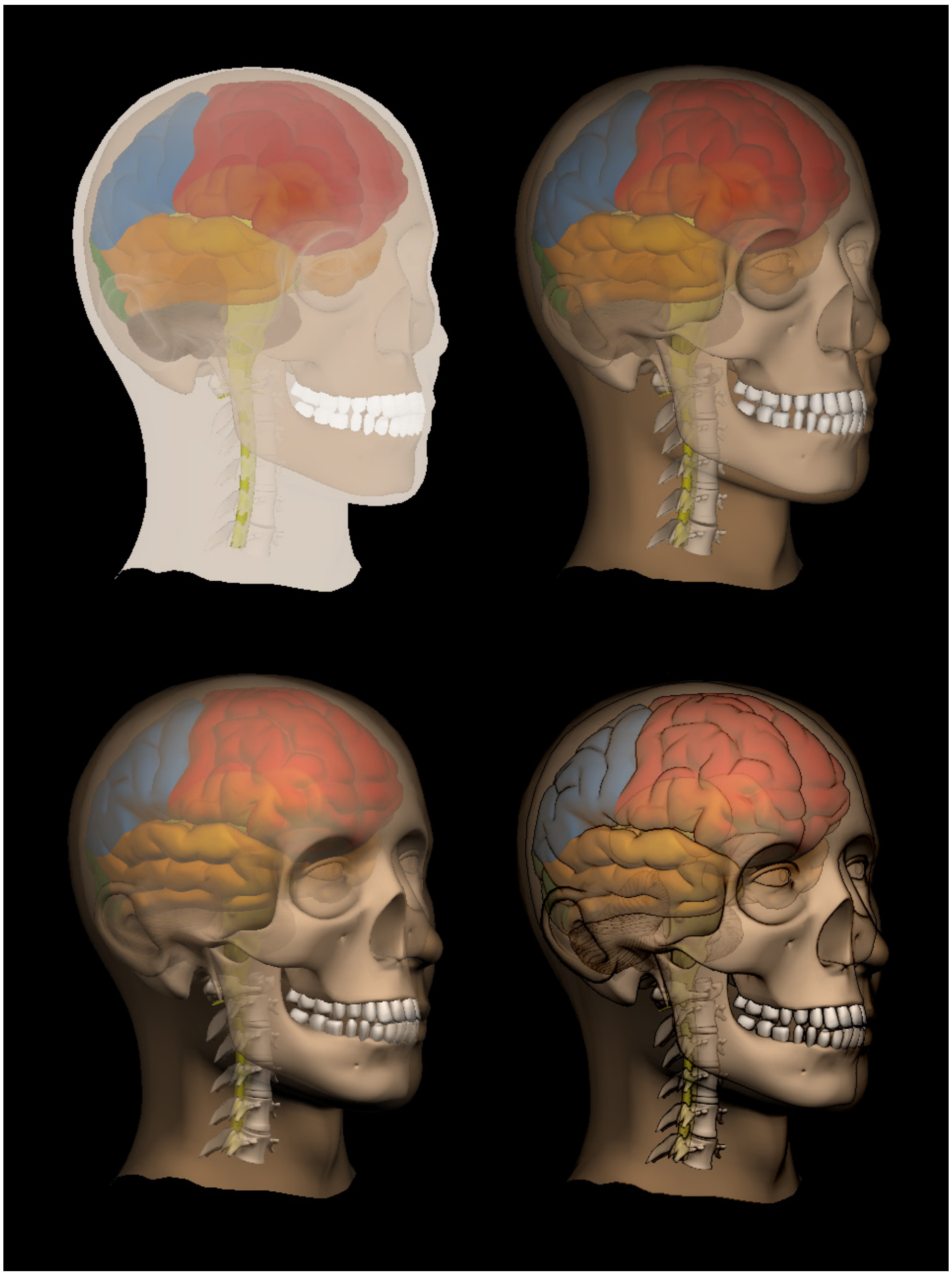


Figure 2.6: In the left figure in the first row, the basic model with attention (color modulation based on importance) is shown. The second model shows the same setting without attention. The NPR shading (three-point lighting) approach where opacity is based on the curvature of the object is located in the left figure of the second row. We can see that for example the curvature of the skull or of the ear is improved. The last picture shows edges as the additional cues.

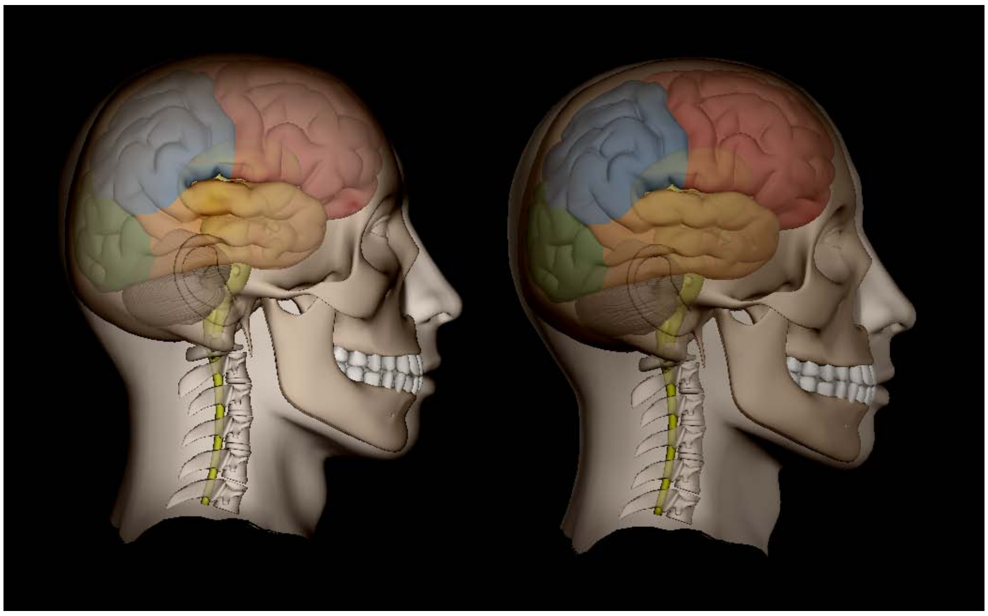


Figure 2.7: In the left figure, the opacity of the object is modulated by single light intensity. The light comes from camera position. In the right figure, three lights are combined for intensity calculation. As you can see, the boundary of the whole model is clearly visible thanks to the light combination, also the dark parts are lightened.

2.3.2.1 Opacity Modulation Based on Surface Curvature, Edge Detection

The implementation of the surface curvature calculation is based on my supervisor work [17]. It uses umbrella operator and it is calculated as the sum of distances between the normal vector of the vertex and normal vectors of its neighbor vertices.

The curvature is than calculated as in following equation 2.7:

$$curvature = min(\left(\frac{d(\vec{n_x}, \vec{n_a}) + d(\vec{n_x}, \vec{n_b}) + d(\vec{n_x}, \vec{n_c}) + d(\vec{n_x}, \vec{n_d})}{2} \right), 1.0); \tag{2.7}$$

where $d(\vec{u}, \vec{v})$ is distance between vectors $\vec{u}$ and $\vec{v}$, x is our fragment and a, b, c, d are neighbor fragments. The $\vec{n_t}$ then represents the normal vector of fragment t. It follows the rule that the larger the distance, the greater the curvature.

Also the depth (distance from viewer) of the neighbour vertices is taken into account. The curvature is combined with equation 2.8 using probabilistic sum presented in previous chapter. The curvature calculated from normal buffer is able to detect also silhouettes of the objects based on the depth in equation 2.8.

$$depth = min((10 * abs(4 * x_d - a_d - b_d - c_d - d_d), 1.0); \qquad (2.8)$$

where t_d represents the depth of fragment t, x is our fragment and a, b, c, d are neighbor fragments. The influence of curvature factor is shown in figure 2.8.

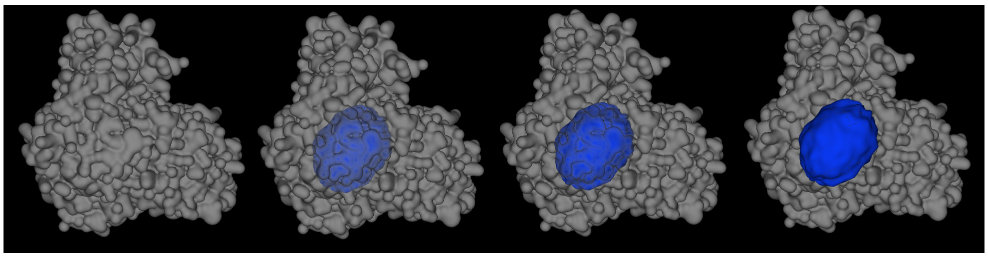


Figure 2.8: The opacity of protein model is modulated based on the curvature factor.

Edge detection method uses the surface curvature calculation. Where the curvature is high, the edge should be present. For these fragments, the dark color is used to contour the edge. For example, the edges are added as additional cues in figure 2.6.

2.4 Methods for Combination of Factors

As I mentioned in chapter 1, there are several possibilities how to combine more factors to opacity modulation. One of the factors should always be a shape. The second one any of depth options. Firstly, I decided to implement multiple fuzzy operators that could be used. The one use select depends what result you desire to obtain. Above that, you could combine the fuzzy operators and create a tree structure for different factors. You can for example combine distance with shape by probabilistic sum, then the result with importance by the maximum. Secondly, the Bruckner's composition will be presented as other possibility that combine shape and depth factor together.

2.4.1 Logic Operators

The method for combination of fuzzy logic operators is possible to set in `float combine(float A, float B){...}` method. A and B are floating point values that represent our factors. Currently, the shape is based on the curvature of the surface/intensity of single or more lights and depth on the relative distance between the layers.

In fuzzy logic, it is not possible to multiply the values. It needs to be a power of some value. The method return single value that is used as alpha for current fragment.

The default option of fuzzy operator combination is a probabilistic sum that corresponds to the OR logic operator for fuzzy logic. The alpha is computed as shown in equation 1.8. The comparison of the maximum and probabilistic sum was already presented in figure 1.26. Another methods like Einstein sum, Bounded sum or Dubois prade is provided in `combine` function.

2.4.2 Bruckner's opacity modulation based on Illustrative Context-preserving Volume Rendering

Besides the fuzzy logic operators, there is possible to combine shape and depth based on [18]. The method was described in section 1.2.1.3.

The opacity is modulated on following matter (equation 2.9 and 2.10):

$$exponent = (dP * abs(dot(\vec{n_x}, \vec{L})) * (1.0 - x_d) * (1 - x_a))^{sP} \tag{2.9}$$

$$alpha = importance^{exponent} \tag{2.10}$$

where dP and dS are parameters set from the GUI. These values correspond to how much the depth (dP as $depthPower$) and shape (sP as $shapePower$) parameters are taken into consideration. The dP param roughly corresponds to the depth of clipping plane. The dS param allows us to determine the sharpness of the cut [18]. The influence of these parameters are demonstrated in figures 2.9 and 2.10.

The light intensity (shading intensity) parameter is calculated by dot product of normal vector of fragment x and vector directing to the light ($\vec{L}$). The x_d parameter refers to depth of fragment x. It corresponds to position dependent term from [18]. Due to the term $(1 - x_a)$, where x_a is alpha of fragment x, structures located behind semi-transparent regions will appear more opaque.

This method was implemented in Tiger library. I reused the implementation so it could be combined in our application.

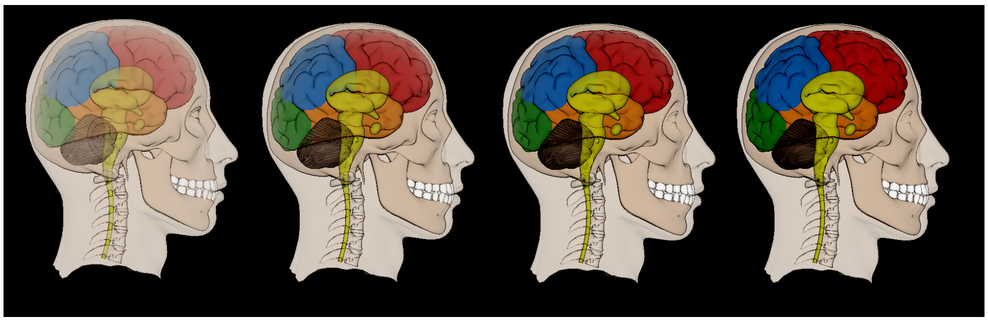


Figure 2.9: The Bruckner's opacity modulation technique based on the Illustrative Context-Preserving method. From left to right the depth power parameter increases. It corresponds to the depth of clipping plane.

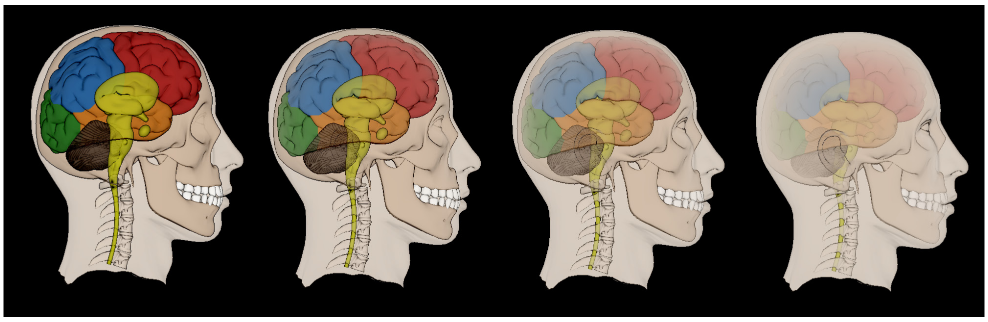


Figure 2.10: The Bruckner's opacity modulation technique based on the Illustrative Context-Preserving method. From left to right the shape power parameter increases. It determine how sharp the cut will look like.

2.5 Design of Application for Visualization of Visual Cues

The design of application for visualization and combination of visual cues is based on the existing implementation of `Tiger` library.

The layout of this application window with GUI is presented in figure 2.11. The panel with parameters is located on the right side of the panel. It is possible to hide it from the `View > Show parameters` menu. On the panel, there are several sliders, that sets the opacity of the whole model, the influence of shape (curvature) factor and influence of depth (distance) factor.

The second part of the panel allows the user to define a combination of factors. The fuzzy logic operators and Bruckner's composition is available. Both methods are described in section 2.4. The last part focuses on the parameters of Gooch's shading (section 2.3.1).

Variety of visual cues is possible to enable or disable in the `View` menu. It

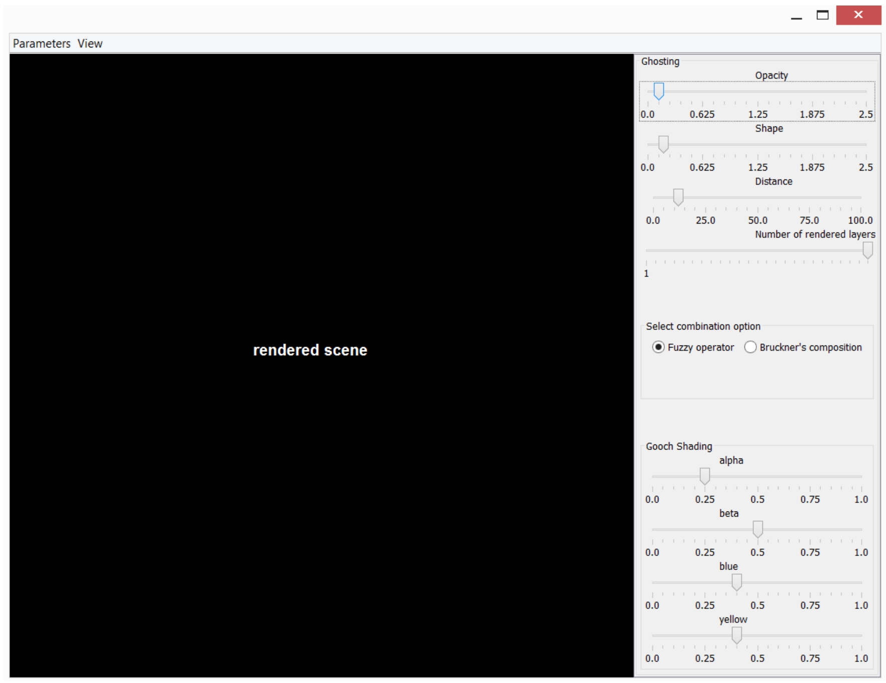


Figure 2.11: Window layout with GUI for visualization and combination of visual cues.

allows us to combine multiple cues. In addition, it is possible to show model parameters, that allows us to set importance/transparency of each mesh in complex 3D model. The rendering area is interactive. It allows the user to rotate model by pressing the left mouse button and dragging.

2.6 Test Application Design

The testing application should be able to handle several tasks. The application layout should be divided into two parts. The first part displays the current scene image, the second part is a control panel. The simple layout could be seen in figure 2.12.

From the user's point of view, the control panel is very simple. For the shape perception, it should provide short task description, buttons that allow him to perform the finish touch to gauge figure orientation and button to load next task that should be handled. For depth perception task, there should be buttons to answer the questions. For text readablity task, the input field should be provided. When user solves all requested tasks, it should automatically finish the test.

The rendering area should display chosen scene image. Moreover, in gauge figure tasks there should be also displayed gauge figure with default orienta-

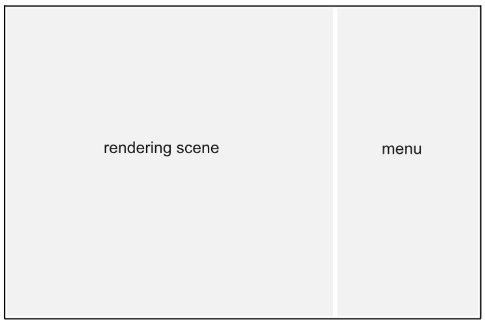


Figure 2.12: Simple layout of application for semi-transparency perception.

tion. The user should be able to rotate with the gauge figure.

2.6.1 Test Scenes

The sequence of images that will be generated from visualization application is the test scenes. These images for shape perception and depth perception with their description in JSON format should be provided to the test application. Moreover, the user should be able to practice the task before the test starts.

Scene definitions should create single JSON array. If I use phrase 'single scene' in context of scene definition, I mean the unique view on the displayed object. If we rotate the object, it would be another scene. The method then refers to the certain level of transparency (opaque, semi-transparent, transparent). For all methods, the gauge figure positions are shared within the scene. The single scene definition follows:

```
{
   "id": id,
   "model": modelName,
   "normalMap": normalMap,
   "methods":[imgName_1, ... ,imgName_n],
   "gaugePosition":[{"x":x1,"y":y1},{"x":x2,"y":y2},...],
   "leftText": leftLabel,
   "rightText": rightLabel,
   "closer": correctAnswer
}
```

The `modelName` corresponds to folder name with images of this scene, `normalMap` contains name of generated normal map. In array `methods`, names of generated images are listed. The `gaugePosition` is an array that contains several objects representing points with x and y coordinate. To define correct answers for left and right labels, the `leftText`, `rightText`, and `closer` is defined. The perception user study is explained in more detail in Chapter 5.

2.6.2 Gauge Figure Task for Shape Perception

I will display two gauge figure on the screen. One gauge figure will be placed on the object surface. This gauge figure should consist of two parts - cylinder and torus. The shaded material should be used for both objects. The gauge figure will be placed based on points predefined from test setup generator in visualization application. Selected points will be chosen in random order.

The second gauge figure will be located in top left corner of the screen. It will be enlarged and it should consist of two cylinders. The radius of the disk is three-quarters of the length of the needle for both of gauge figures. I will consider needle pointing in the positive y-axis direction as default orientation.

The user will then be allowed to interact with gauge figures with mouse dragging. It will rotate the gauge figure around the point on the surface of the rendered object.

For the evaluation of gauge figure error, I use angle between the real normal vector in specified point and the normal vector that user set by gauge figure. The error is computed by equation 2.11:

$$error(\vec{n_r}, \vec{n_u}) = arccos(dot(\vec{n_r}, \vec{n_u})) * \left(\frac{180}{\pi}\right) \tag{2.11}$$

where $\vec{n_r}$ represents the real normal vector and $\vec{n_u}$ the normal vector set by user.

I will also use polar coordinates tilt and slant. A tilt is calculated from the x and y coordinates of the normalized surface normal as shown in the equation 2.12

$$t(\vec{n}) = arctg2(\vec{n}.y, \vec{n}.x) * \left(\frac{180}{\pi}\right) \tag{2.12}$$

where $\vec{n}$ is the surface normal and $arctg2(x, y)$ is an inverse tangent function that returns the arc tangent of y/x using the signs of arguments to determine the correct quadrant. The tilt represents the rotation around the viewing vector. From the experiment I will collect both tilt and its absolute value.

The slant is calculated from z coordinate of surface normal as follows (equation 2.13):

$$s(\vec{n}) = arccos(\vec{n}.z) * \left(\frac{180}{\pi}\right) \tag{2.13}$$

where $\vec{n}$ is the surface normal and $arccos(x)$ is an inverse cosine function. The slant represents the depth of a surface's slope. When the slope is 0 degrees, the surface is perpendicular to the view vector [24]. From the experiment I will collect both slant and its absolute value.

2.6.3 Depth Perception

For the depth perception task, we consider simple layout of the scene. For two objects, we will compare their distance from the surface of the displayed object. In our case it will be two labels intersecting the 3D model of protein. For each scene, it must always be clear which of these two objects is on the left side, and which object is on the right side. Then we can ask the user to choose between those objects.

2.6.4 Collecting data

Web application made by Antonina Lebedeva [15] will be used to collect data from the experiment. It contains administration application to manage experiments and evaluate collected data. The evaluation is done using confidence intervals.

The test application will use JavaScript transfer library [15] for communication with web application. It uses the AJAX calls to web application endpoint.

Chapter 3

Implementation

In the scope of this chapter, I am describing how the testing application should handle the specified task and the technology I have been working with. I also present the application structure description and other implementation details.

3.1 Application for Visualization and Combination of Visual Cues

3.1.1 Technologies

I have chosen Java programming language as the Tiger library that I used in this work is implemented in this language. In following sections, I will briefly introduce technologies that I used in the implementation.

OpenGL

OpenGL is multiplatform application programming language for rendering graphics in 2D or 3D. It offers API for managing buffers, textures and allows us to communicate with the GPU. OpenGL is regularly updated by Khronos Group.

JOGL

Java Binding for the OpenGL API is designed to provide hardware-supported 3D graphics to applications written in Java. JOGL provides full access to OpenGL APIs and it is able to intergate with other libraries and widget sets.

Tiger

As I mentioned before, Tiger or Toolkit for Interactive Graphics rendering is a library written in Java language. It allows us to easily implement multipass rendering process. The author of this library is the supervisor of my thesis, Ing. Ladislav Čmolík, Ph. D.

GLSL

GLSL or OpenGL Shading Language with syntax based on C programming language offers us to write shaders. Shaders allow us to calculate effects on graphics hardware with a high degree of flexibility. GLSL provides us many features to work with textures, matrices, geometry etc.

3.1.2 Structure of the Application

Major parts of my work with NPR (nonphotorealistic) methods are situated within the `perception` package.

Description of tiger.perception

In `perception` package there are several enhancement methods I implemented. The backbone of the algorithm is taken from existing implementation of ghosting. The origin multi-pass rendering process is simplified for purpose of this thesis, unnecessary parts were removed, and it is implemented in `tiger.perception.SimpleGhosting` class. The `SimpleGhosting` is an abstract class which implements `GLEventListener`. `GLEventListener` is an interface that declares events to manage OpenGL rendering into a `GLAutoDrawable`. In `GLEventListener`, there are four event handlers that we know from OpenGL:

- `init` - is called immediately after the OpenGL context is initialized
- `dispose` - is called before the OpenGL context is destroyed
- `display` - is called to render OpenGL graphics
- `reshape` - repaint after resizing drawable

The `display` method is crucial for re-rendering and it is called periodically.
The implementation of main enhancement methods is carried out during the `composeLayers` pass. The `composeLayers` pass is not the composition of separate layers itself, but it handles the opacity modulation based on shape, the relative distance of layers etc. The `composeLayers` pass is defined in `tiger.perception.Ghosting_K` class which is a subclass of the `tiger.perception.SimpleGhosting` class.

The `createCompositionPass()` method of the mentioned class contains the initialization of the pass. A fragment shader is defined, in this case, the `tiger.perception.Composition` fragment shader, and all needed GLSL variables are passed to the shader.

Opacity modulation is handled by this shader. I will briefly describe its content. The `float combine(float A, float B)` method is used for a combination of several factors. The default combination is probabilistic sum as mentioned in section 2.4. Other possible logic operators for combinations are simply commented and its possible to replace the default value by them.

The `float curvature(vec4 X)` method is used when curvature based on [17] is calculated. It takes current fragment from depth texture as an input parameter. The x, y, and z values represent the normal vector. Last value a is the depth. In this method, neighbor fragments and depth is used to calculate surface curvature as mentioned in section 2.3.2.1.

The Gooch's shading is calculated by `void setGoochShading (float al)` method. The implementation was already presented in section 2.3.1. Similarly, the implementation of NPR shading in `void setNPRShading (float al)` method was presented in section 2.3.2. The `float intensityCurvature (vec3 N)` method that enables the three-lights intensity opacity modulation.

In `void main()` method the alpha of current fragment is calculated based on active visual cues that are set from GUI.

Blending with background is located in `backgroundBlend.frag` in `tiger.perception` package.

3.2 Test Setup - Generator

The generator of the test setup is another part of the visualization application. It allows us to simply generate JSON files and images to our test app. The implementation is in `tiger.perception.testGenerator` package.

The `Generator` is an abstract class, that contains `javax.swing.JFrame` objects as its GUI and JSON builder elements. `ShapePerceptionGenerator` and `DepthPerceptionGenerator` classes inherit from it. Each test consists of several `Parts`. Each `Part` has its own GUI (`javax.swing.JComponent`). For the transition between individual parts, the `PartManager` is used. It implements the `IPart` interface with methods for setting up parts. When the `next()` method on `PartManager` is called, the Generator GUI allows to interact with currently active part. The general concept is possible to amend to any GUI changes with several parts.

3.3 Test Application

Since we wanted to perform the test remotely, we decided to use web technologies like HTML, JavaScript and WebGL.

3.3.1 Technologies

As was mentioned before, for the test application implementation we have chosen web technologies like HTML, CSS, JavaScript, jQuery and Three.js library.

Three.js

Three.js is a cross-browser high-level JavaScript library and API that can be used to create and display 3D computer graphics in a web browser. This library uses WebGL technology and allows us to use WebGL in very simple and intuitive way. I used Three.js library for gauge figure task implementation.

jQuery

Another JavaScript library - jQuery - was used for implementation. It provides us simple way to handle events etc.

3.3.2 WebApp structure

This is the main structure of the test application project.

```
WebApp
├── build ............................................ three.js files
├── css
├── fonts
├── img ............ test images generated from visualization application
├── js ................................... js files of test application
├── index.html
├── depth.html ....................................... shape test
├── shape.html ....................................... depth test
├── scenes.json ............ JSON file with scene definitions for shape
├── depth.json ............. JSON file with scene definitions for depth
└── other html files
```

Firstly the participant needs to join the test. The `initProperties()` function is called where the `participantIndex` and `modelIndex` is obtained from server local storage. After that counterbalancing is performed based on `participantIndex`.

The sequence of models and methods are controlled by counterbalancing described in section 5.3. Redirection between models and to another experiment part is handled by `redirectToNextModel(path)` function.

The gauge figure task is handled from `js/overlayCanvas.js` file. The `skelet()` function is onload function and it is responsible for loading images based on their definition in `scenes.json` file. Then the scene setting up is processed.

To be able to display something with three.js, we need three things - scene, camera, and renderer. For our test, we use the orthographic camera. For the gauge figure task we use a scene that overlays the images, so we need to create `WebGLRenderer` with a transparent background.
The image of scene is displayed by `displayCurrentScene(currentScene)` function. Then the rendering process begins, and mouse dragging event is handled.

The `depth()` function from `js/depthTest.js` file is loading images based on their definition in `depth.json` file. Only simple GUI is needed for this test. Two buttons that enable the user to answer which of presented labels is closer, then two text input fields to enable rewrite the labels. The buttons are handled by `onclick` events as usual.

Chapter **4**

Results

In this chapter, a selection of output images rendered by the application is presented. The evaluation of feature perception is without any doubt subjective. I will again briefly discuss the methods and visual improvements in presented images. The test application walkthrough will be presented in Apendix A.

Opacity modulation based on curvature, light intensity and distance based modulation

The opacity modulation based on curvature with different shape power setting is shown in figure 4.1. This figure shows the influence of shape power value.

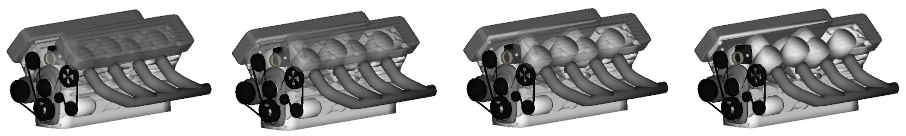


Figure 4.1: Opacity modulation based on curvature. From the left, the shape power $sP = 0$, $sP = 0.25$, $sP = 0.5$ and $sP = 1.25$

The opacity modulation based on distance between two layers with different distance power setting is shown in figure 4.2. This figure shows the influence of distance power value.

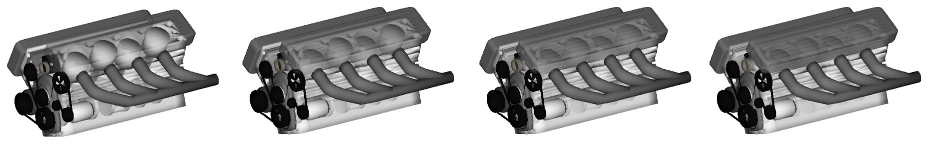
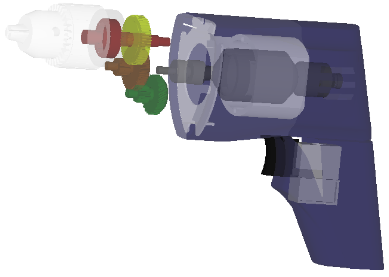
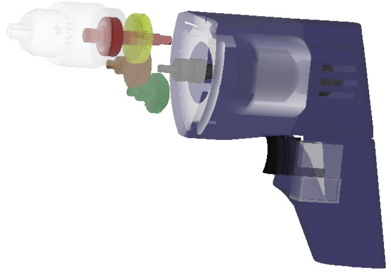
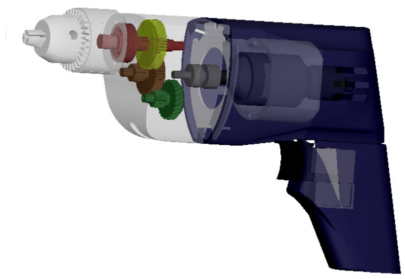


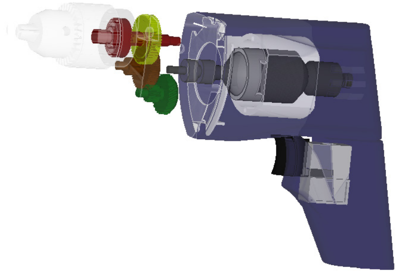
Figure 4.2: Opacity modulation based on distance between samples. From the left, the distance power $dP = 5$, $dP = 15$, $dP = 25$ and $dP = 50$

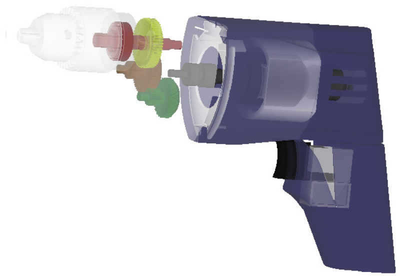
I present various visual cues in figure 4.3. In the first image, no additional visual cues are provided. In the second image, the distance fog is used to support depth perception (dP as distance power).


(a) no additional visual cues


(b) distance fog based on relative distance between two layers, $dP = 25$


(c) opacity modulation based on light intensity ($sP = 0.25$)


(d) opacity modulation based on shape curvature, $sP = 0.25$


(e) shape (curvature, $sP = 0.25$) and distance ($dP = 25$) is combined based on fuzzy logic probabilistic sum

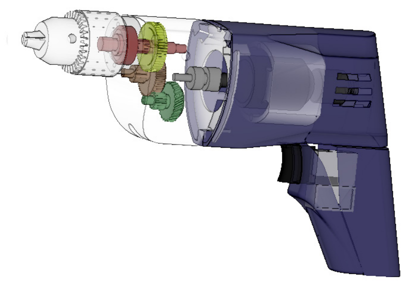

(f) same as (e) with edges

Figure 4.3: Result of distance and shape based modulation and combination of different visual cues.

The opacity modulation based on light intensity (with darker tones on edges) or shape curvature is shown in second row of figure 4.3 (sP as shape power). Also the combination based on fuzzy logic operators is presented. The last image shows the edges as additional cue.

Distance fog, NPR shading, Specular highlights

Depth perception is supported by distance fog/aerial attenuation in figure 4.5. From top to bottom the distancePower (dp) is increasing. The left images are presented with distance fog only. For the right ones, I combined distance fog with NPR shading for better shape perception. As you can see on the right side the curvature is clearly visible even in the fog.

We can also combine the NPR shading and distance fog with specular highlights as you can see in figure 4.4. The specular value is computed only for the primary light. The specular highlights are view-dependent cue so it helps mostly with some motion mechanism or in interactive systems.

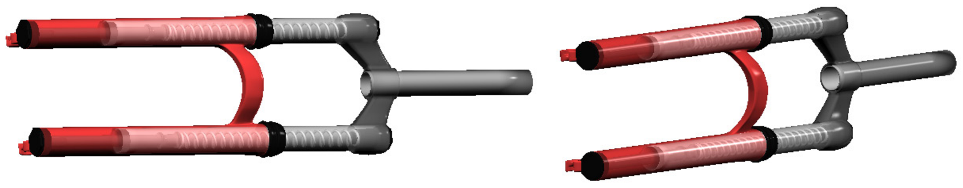


Figure 4.4: Specular highlights supports the NPR shading and distance fog, but they are view-dependent cue. The specular highligts are most visible when the surface with curvature leaning toward the light. $dP = 60$

Bruckner's combination with NPR shading

The Bruckner's opacity modulation technique combined with NPR shading (three point lightning) is presented in figure 4.6. The most left images in each row are used as reference images. The classic transparency (with different alpha values for each mesh) is used for both of them. Compared to the first line, in the second row the NPR shading is used as additional shape cue.

From the second to the fourth column the depth of cutting plane in Brucker's model is increasing, thus it reveals more of the inner structures. Since we use NPR shading that should support shape visual cues, you should focus on the curvature of semi-transparent object - for example on the head and skull. You can see that orbits are better shaded so you can see their curvature. From my point of view, the NPR shading also improve the spatial properties of the model.

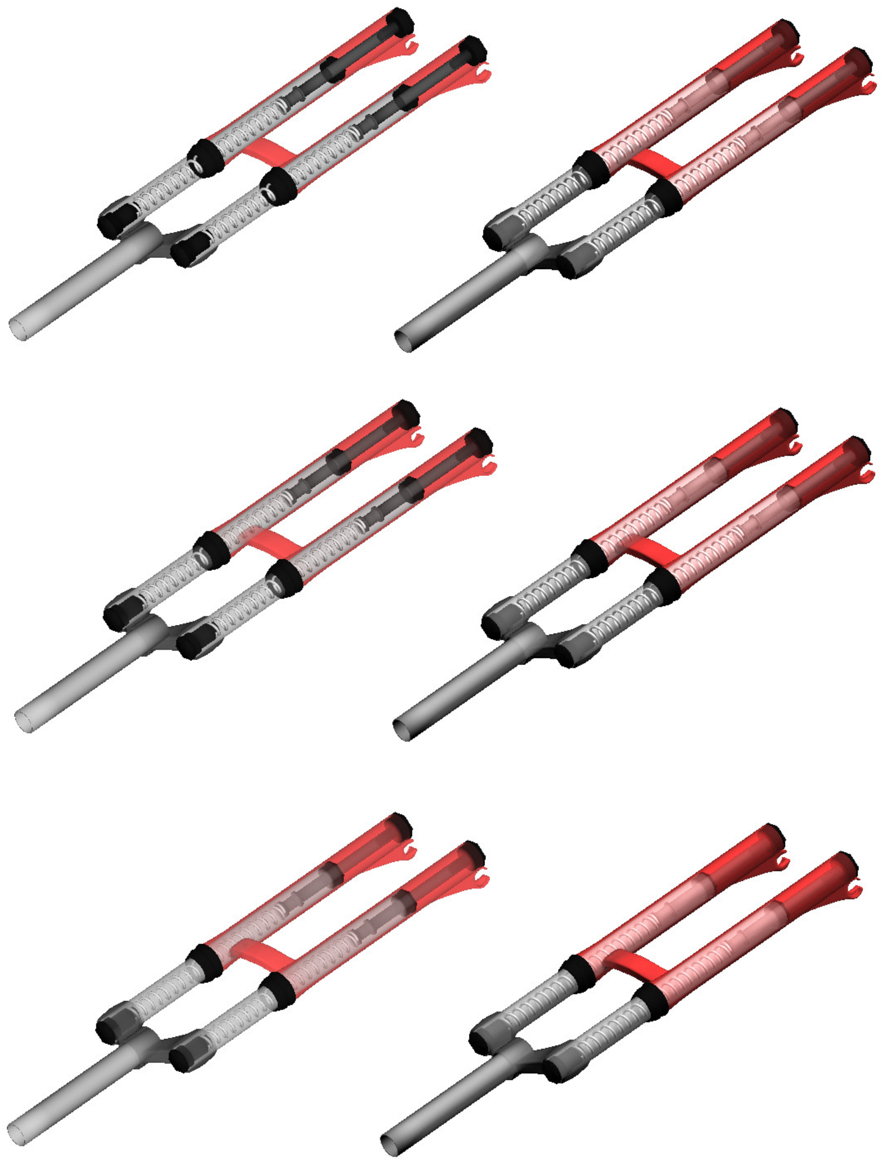


Figure 4.5: In the first column only the distance fog/aerial attenuation is presented. In the second column, the shape is enhanced by NPR shading method. $dP = 0$ in the first row, $dP = 10$ in the second row and $dP = 35$ in the last row.

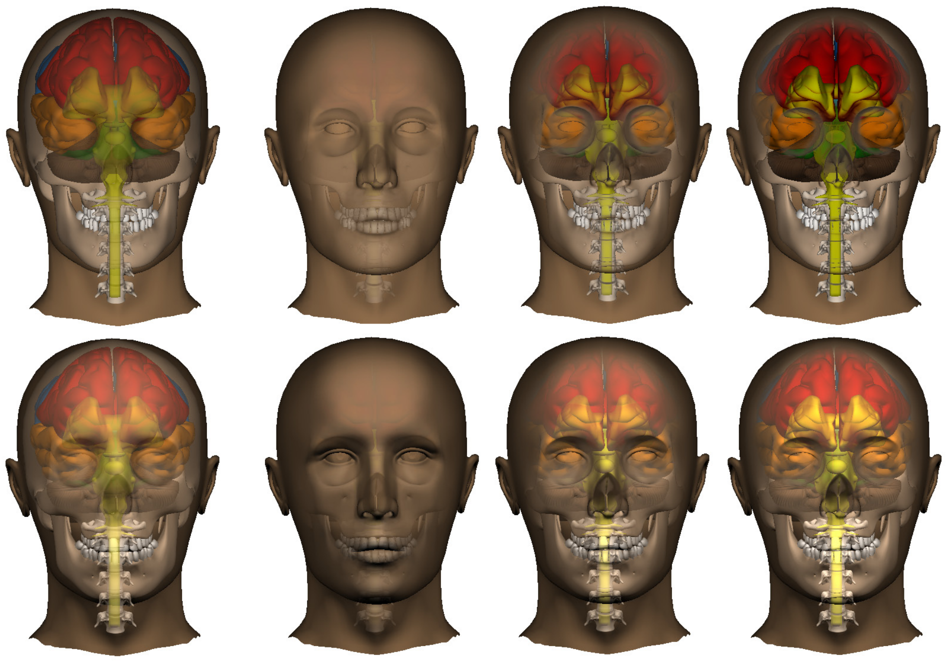


Figure 4.6: In the first column the classic transparency and NPR shading for the same transparency setting is used. This column is used as reference. Then in the first row, no additional visual cues are added. For the second row we use our NPR shading model. The Bruckner's composition was used for all of these images. From left to right the depth of cutting plane is increasing (dP from 1 to 3). The sharpness of the cut is set to minimum ($sP = 2.5$). Edges of the cut is then blurred.

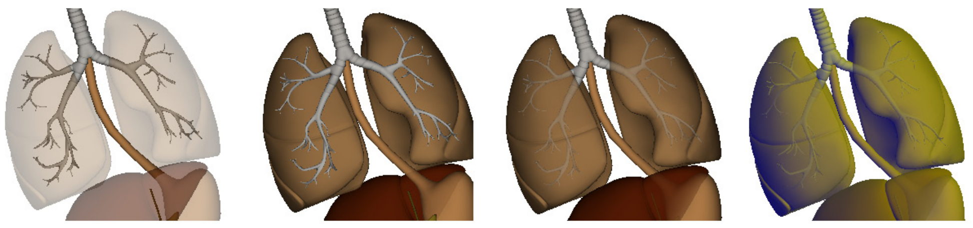


Figure 4.7: From the left side: The first image shows classic transparency without selective transparency. Same view with selective transparency is on the second image. For the third image, the distance fog is added ($dP = 25$). Finally, the Gooch's shading along with distance fog is presented in the last image ($alpha = 0.25, beta = 0.5, blue = 0.4, yellow = 0.4$).

Distance fog, Gooch's shading

As was mentioned before, the Gooch's shading only works in case of fully opaque objects. There is no additional cue in case of semi-transparency. Example of result is shown in figure 4.7. The distance fog is combined with Gooch's shading. There is no additional visual cue for better perception of shape on semitransparent objects.

NPR shading, Edges, Specular Highlights

Fuzzy logic based composition with curvature based opacity modulation, distance fog, NPR shading, edges and specular highlights are shown in figure 4.8.

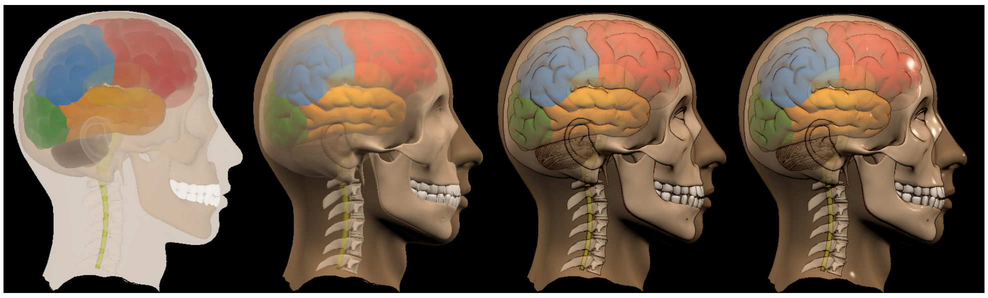


Figure 4.8: All images uses curvature based opacity modulation ($sP = 0.25$) and distance fog ($dP = 25$). From the left: The first picture shows visualization with attention. The NPR shading is presented in the second picture. Moreover, the edge detection is shown in the third picture and specular highlights in the last one.

Chapter 5

Perceptual User Study

To test visual cues and their combination, I have designed an empirical experiment that consists of three parts - one for shape perception, one for depth perception and text readability test. I try to find out use case that could be used in practice. After consulting with my thesis supervisor, I decided to use complex structures like proteins along with labels.

The shape is difficult to distinguish when it is occluded by a label. Thus it is necessary to find out a way to provide better shape perception and keep the label readable. In case of depth perception, the distance between the label and object surface could be a useful cue of spatial properties of the complex object. Even if the labels are occluded by an object, we still need to be able to read the text.

The possibility to study shape perception is important for nonphotorealistic techniques since we need to help our visual system reconstruct the surface of the visualized objects. Although there are several methods to test perception of shape, the most common is the gauge-figure task.

The gauge-figure task [25] is a method where users interact with a gauge and change its orientation until it aligns with object surface. The gauge consists of thin disc pierced in the center by a short line segment. The task with this method is to place disk tangentially to object surface and thus line segment perpendicularly to surface to represent a normal vector in specified point.

The depth perception needs to be supported by depth cues since the object is rendered on the two-dimensional display. Depth cues like occlusions (X and T junctions) and shading help us to reconstruct spatial properties of the models. For our experiment, we decided not to use perspective projection since we want to focus on depth perception based only on semitransparency. Therefore we use parallel projection to eliminate perspective-based depth cues.

As I mentioned before, the text readability is also important when we labeling any complex objects. The visibility of these labels needs to be resolved along with keeping the information of their position in 3D space in order to associate labels with objects.

5.1 Experiment Design

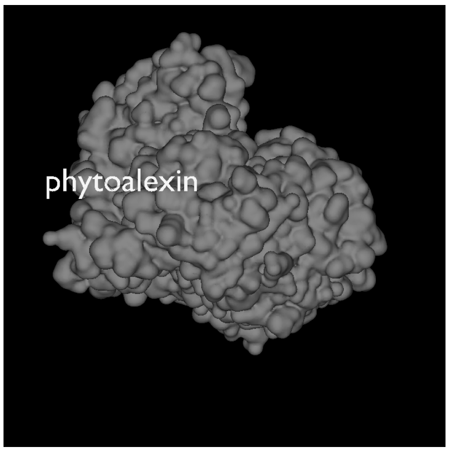
I select the between-subject design for the experiment. In the between-subject design, the various tasks are given to different groups of participants. The independent factor for both parts of this experiment is a visualization method. To avoid a learning effect and carry-over effect I test single visualization method on multiple scenes of similar complexity. These scenes are counterbalanced as I will discribe in section 5.3.

5.2 Methods as Levels of the Factor

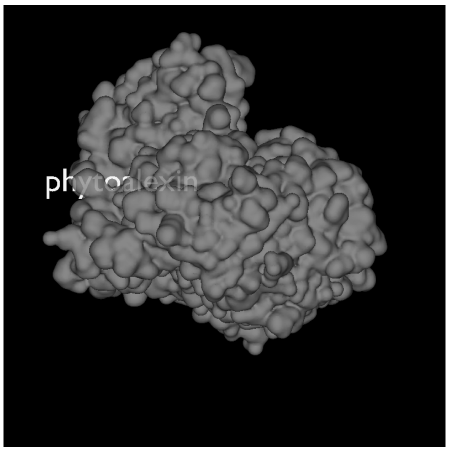
As already mentioned, the independent factor for the experiment is the method. The factor has four levels. In following sections, I will describe the levels for both parts of the experiment along with expectations of the results. Examples of the scenes with methods are presented in figure 5.1 and 5.2.

Shape perception

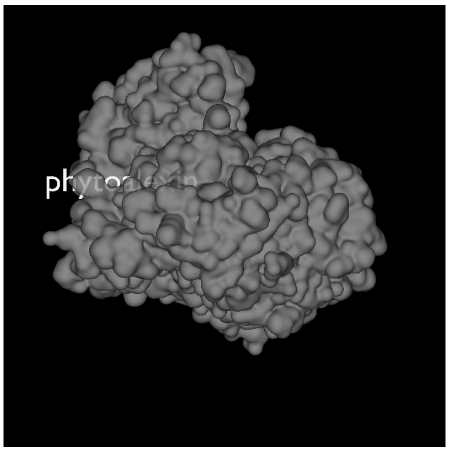
1. Labels over the object
    - Object is fully transparent. Participant does not perceive the shape of object surface precisely since it is occluded by text.
    - Text is clearly readable.
2. Classic semi-transparency
    - Object is semi-transparent using classic transparency. The perception of the shape of the surface should be improved.
    - Text will be readable.
3. Ghosting with shape and depth cues
    - Object opacity is modulated based on depth and shape cues. The perception of the shape of the surface should be vastly improved.
    - Text will be readable.
4. Resolving visibility based on depth test
    - The object is not occluded and its opacity is not modulated. The shape of the object should be perceived the best.
    - Text will not be readable.



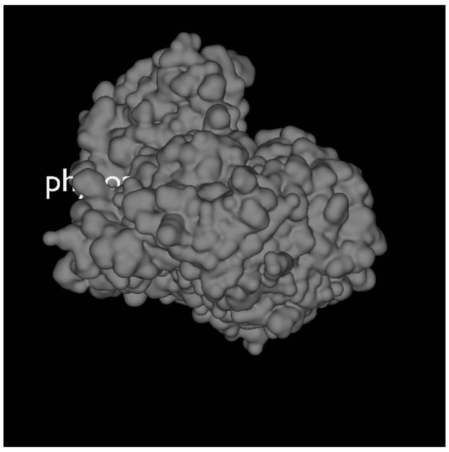
(a) method 0: fully transparent object, surface is occluded by label



(b) method 1: classic semi-transparent object, the perception is improved



(c) method 2: ghosting with depth and shape visual cues, the perception is significantly improved



(d) method 3: fully opaque object, the perception is the best, labels are not readable

Figure 5.1: Examples of the shape perception test scenes with different methods of visualization.

Depth perception

1. Labels over the object

   - No depth cues are not provided.
   - Text is clearly readable.

2. Classic semi-transparency

- Some depth cues are provided. The participant can see if the object is in front of the object. He can see the visibility discontinuities (where the label intersect the object). He can't decide how far from the surface the label is.
- Text will be readable.

3. Ghosting with shape and depth cues

   - This adds another depth cue - the participant can see how far behind the object the label is.
   - Text will be readable.

4. Resolving visibility based on depth test

   - The opacity of the object is not modulated. The depth is possible to decide only by visibility discontinuities.
   - Text will not be readable.

5.3 Counterbalancing

To eliminate ordering and carry-over effects I will counterbalance the models based on a balanced Latin square design. The balanced Latin square design is recommended when more than two conditions are used. It ensures that when each condition is tested, it would be preceded and followed equally often by every other condition [26].

Since I have four models in the experiment, the balanced Latin square will be 4x4 matrix presented in figure 5.3.

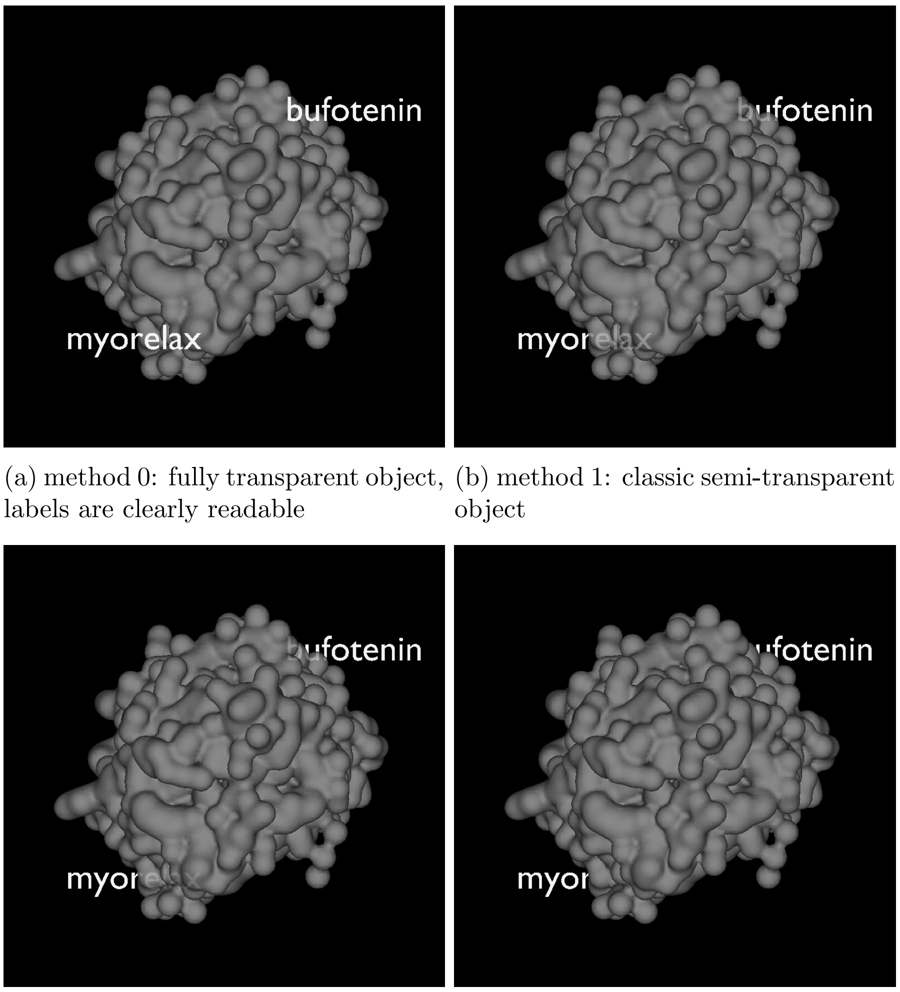
Each participant will be tested for single method in both shape and depth tests, thus we will obtain independent group when we evaluate all the models together. Furthermore, it should eliminate the learning effect. When we combine balanced Latin square with selected methods, we then obtain 16 rows of (model, method) pairs (figure 5.4).

5.4 Measured variables

In this section I will describe which variables from the tasks is crutial for the experiment evaluation. The variables will be described in detail in experiment setup (section 5.4).

Shape perception

For the shape perception experiment, we will have 4 gauge figure task for each model. We will measure especially error of gauge figure task (angle between



(a) method 0: fully transparent object, labels are clearly readable

(b) method 1: classic semi-transparent object

(c) method 2: ghosting with depth and shape visual cues

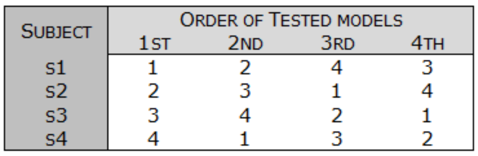
(d) method 3: fully opaque object, labels are not readable

Figure 5.2: Examples of the depth perception test scenes with different methods of visualization.

real normal vector and normal vector set by the participant), completion time for each gauge figure task and error of the text readability task. The error of tilt and slant will be measured in addition.

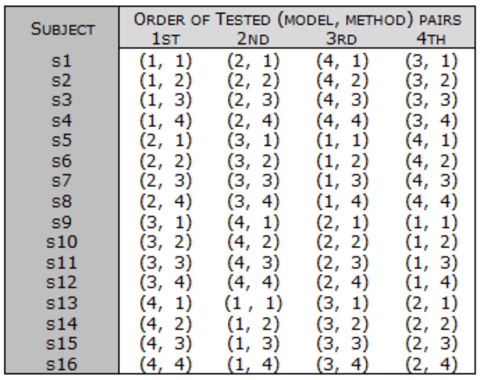
We should be able to get 4 x 4 = 16 values for gauge figure task completion time, 16 values of each gauge figure task errors and 4 values of text readability task errors from each participant.

If we complete the 16 rows of pairs twice, we will have 8 x 16 = 128 values for gauge figure task completion time, 128 values for each gauge figure task errors and 32 values of text readability task error for each of the methods.



Subject	Order of Tested models			
	1st	2nd	3rd	4th
s1	1	2	4	3
s2	2	3	1	4
s3	3	4	2	1
s4	4	1	3	2

Figure 5.3: Balanced Latin square for four testing conditions (models). Each participant (subject) receives models in the order of its row.



Subject	Order of Tested (model, method) pairs			
	1st	2nd	3rd	4th
s1	(1, 1)	(2, 1)	(4, 1)	(3, 1)
s2	(1, 2)	(2, 2)	(4, 2)	(3, 2)
s3	(1, 3)	(2, 3)	(4, 3)	(3, 3)
s4	(1, 4)	(2, 4)	(4, 4)	(3, 4)
s5	(2, 1)	(3, 1)	(1, 1)	(4, 1)
s6	(2, 2)	(3, 2)	(1, 2)	(4, 2)
s7	(2, 3)	(3, 3)	(1, 3)	(4, 3)
s8	(2, 4)	(3, 4)	(1, 4)	(4, 4)
s9	(3, 1)	(4, 1)	(2, 1)	(1, 1)
s10	(3, 2)	(4, 2)	(2, 2)	(1, 2)
s11	(3, 3)	(4, 3)	(2, 3)	(1, 3)
s12	(3, 4)	(4, 4)	(2, 4)	(1, 4)
s13	(4, 1)	(1 , 1)	(3, 1)	(2, 1)
s14	(4, 2)	(1, 2)	(3, 2)	(2, 2)
s15	(4, 3)	(1, 3)	(3, 3)	(2, 3)
s16	(4, 4)	(1, 4)	(3, 4)	(2, 4)

Figure 5.4: Sixteen rows of (model, method) pairs. Each participant will be tested on one row of these pairs.

Depth perception

For the depth perception experiment, we will have two labels on the screen with a single object. We will ask which label is closer to the visible surface of the object. For each model, we will test 4 different cases of label positions. We will measure completion time and error for each depth decision task and the error of the text readability task.

We should be able to get 4 x 4 = 16 values for depth perception task completion time, 16 values of depth perception task errors and 8 values of text readability task errors from each participant.

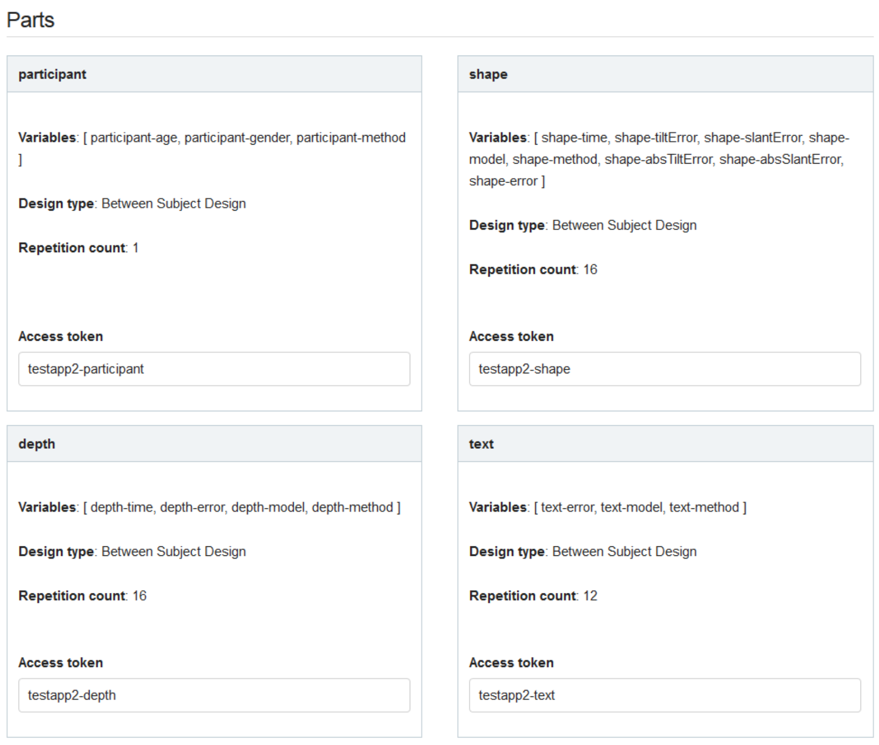
If we complete the 16 rows of pairs twice, we will have 8 x 16 = 128 values for depth perception task completion time, 128 values for depth perception task errors and 64 values of text readability task error for each of the methods.

Text readability

If we combine results for test readability from both previously described experiments, we will have 96 values of text readability task error for each method. However, we must ensure that one participant is tested for the same method in both of the experiments.

Experiment Setup

The experiment consists of four parts (figure 5.5). The access token of the whole experiment needs to be unique and it is the connecting key for the communication between test application and the web service. To define in which part the data should be stored, the part's access token is used. The repetition count parameter displays the number of tasks that single participant needs to finish in for each part.


Parts

participant

Variables: [participant-age, participant-gender, participant-method]

Design type: Between Subject Design

Repetition count: 1

Access token

testapp2-participant

shape

Variables: [shape-time, shape-tiltError, shape-slantError, shape-model, shape-method, shape-absTiltError, shape-absSlantError, shape-error]

Design type: Between Subject Design

Repetition count: 16

Access token

testapp2-shape

depth

Variables: [depth-time, depth-error, depth-model, depth-method]

Design type: Between Subject Design

Repetition count: 16

Access token

testapp2-depth

text

Variables: [text-error, text-model, text-method]

Design type: Between Subject Design

Repetition count: 12

Access token

testapp2-text

Figure 5.5: Experiment parts from administration application.

I will now describe the variables from the setup on web service. In this web service, there is not possible to calculate additional values from the variables, so it is necessary to send all potentially interesting data directly from the test application.

Participant The `participant` part will collect data about participants that joined the study. It collect following variables:

1. `participant-age`: age of the participant, integer values, normal distribution

2. `participant-gender`: gender, male/female, filter

3. `participant-method`: id of method that is assigned to this participant from counterbalancer, filter

Shape The `shape` part handles collecting of data for shape test. Variables are:

- `shape-time`: completion time of single gauge figure task
- `shape-tiltError`: deviation of tilt of normal vector that was set by participant in degrees, normal distribution
- `shape-slantError`: deviation of slant of normal vector that was set by participant in degrees, normal distribution
- `shape-model`: name of the model that is tested, filter
- `shape-method`: id of method that is assigned to this participant from counterbalancer, filter
- `shape-absTiltError`: absolute value of tilt deviation of normal vector that was set by participant, in degrees, normal distribution
- `shape-absSlantError`: absolute value of slant deviation of normal vector that was set by participant, in degrees, normal distribution
- `shape-error`: deviation angle of normal vector that was set by participant, in degrees, normal distribution

The most important variable from this task is the `shape-error` value. It defines how much the participant missed in determing the correct shape in gauge figure tasks. If we want to determine whether he failed in tilt or slant, it is possible to use other errors from the test. The completion time is only suplemmentary information.

Depth The `depth` part handles collecting of data for depth test. Variables are:

- `depth-time`: completion time of single depth task
- `depth-error`: 0 if the closer label was correctly determined, 1 if not, error rate
- `depth-model`: name of the model that is tested, filter
- `depth-method`: id of method that is assigned to this participant from counterbalancer, filter

The `depth-error` varible determines wheter the user can determine which of two presented labels is closer to the surface. The completion time is only suplemmentary information.

Text The `text` part collect data for text readability test. Variables are:

- `text-error`: 0 if the text was correctly rewritten, 1 if not, error rate
- `text-model`: name of the model that is tested, filter
- `text-method`: id of method that is assigned to this participant from counterbalancer, filter

5.5 User Testing and Results

I collected data from 32 participants (11 female and 21 male, aged 20 to 46). Most of the participants were students or former students of computer graphics, or the faculty members. Several participants were students of other disciplines.

Shape

I computed the deviation of the normal vector as presented in equation 2.11. It is the angle between the real normal of the surface and the one that participant set.

For each method, I provide a mean value along with a 95% confidence interval. For fully transparent object (method 0) I obtained mean value equal to 27.29 degrees with confidence interval (23.652, 30.927). The mean value for classic transparency (method 1) is 28.569 (24.55, 32.588) degrees, for ghosting (method 2) then 21.279 (17.204, 25.353) degrees. The mean value for fully opaque object (method 3) is 19.377 (17.31, 21.444) degrees.

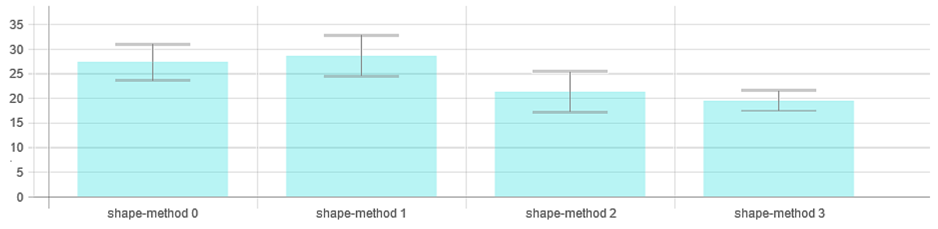


Figure 5.6: Error from shape perception test for different methods.
method 0: fully transparent object
method 1: classic semi-transparency
method 2: ghosting with shape and depth visual cues
method 3: fully opaque object

The confidence intervals for this error is shown in figure 5.6. It shows all models together filtered by the method that was assigned to the participant. On this chart is clearly visible that the method where the objects is fully opaque (method 3) is the best for the shape perception. It only confirms the assumption. The shape should be clearly visible if it is not occluded by anything. More importantly this chart shows, that the ghosting method (method 2) with depth and shape visual cues is significantly better than classic transparency (method 1).

To support this statement, I computed the confidence interval for the difference between two overlapping confidence intervals [27]. The ghosting method (method 2) is for the shape perception significantly better (lower error) then method 0 (6.011 ± 5.462) and better than method 1 (7.29 ± 5.7231) since none of those intervals intersect zero.

In addition, I also present the completion times for the shape perception method. I provide a mean value along with a 95% confidence interval. In case of fully transparent object (method 0), the mean value is estimated to 9320.700 milliseconds (ms) with confidence interval (8096.385, 10730.152). Classic semi-transparency (method 1) has mean value 9962.586 (8447.29, 11749.7) ms and ghosting (method 2) has mean value 11947.843 (9853.165, 14487.826) ms. Finally, the fully opaque object (method 3) takes in mean 6635.677 (5849.667, 7527.302) ms.

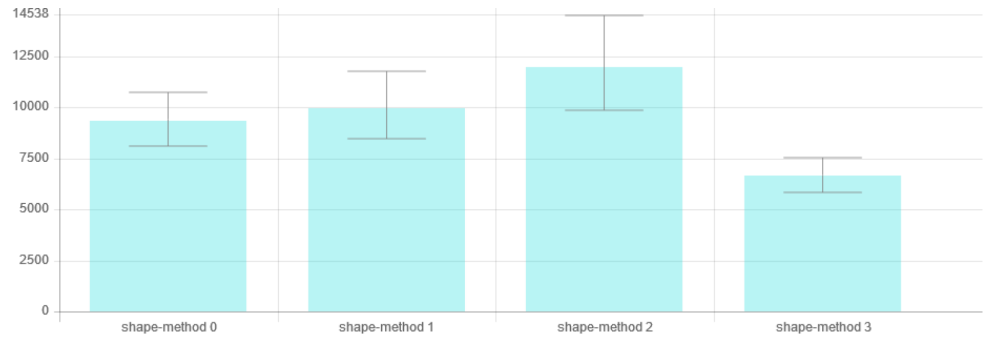


Figure 5.7: Completion time of shape perception test for different methods.
method 0: fully transparent object
method 1: classic semi-transparency
method 2: ghosting with shape and depth visual cues
method 3: fully opaque object

In figure 5.7, the completion times of shape perception test for different visualization methods are presented. Ghosting methods seems to be the most time consuming. I expected this result. Although the shape is strongly supported, it needs to be studied for a longer time than in the other methods. Again, the method with fully opaque object is the fastest to evaluate.

Depth

For the depth perception test only binary values are accepted wheter the participant choose closer label correctly or not.

For each method, I provide a mean value along with a 95% confidence interval. I used One Sample Proportion Calculator [28] to estimate the confidence intervals for depth error evaluation. With fully transparent object (method 0), the mean of error rate is 43.915% (35.91%, 51.92%), for the classic transparency (method 1) it is 45.99% (36.21%, 55.77%) and for the ghosting (method 2) it is 27.335% (17.79%, 36.88%). Finally, the mean of error rate for fully opaque object (method 3) is estimated as 28.53% (18.86%, 38.2%).

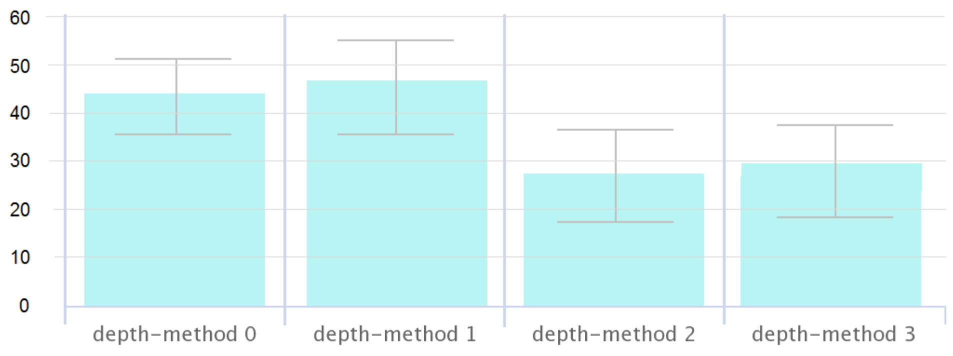


Figure 5.8: Error of depth perception test for different methods.
method 0: fully transparent object
method 1: classic semi-transparency
method 2: ghosting with shape and depth visual cues
method 3: fully opaque object

Results are presented in figure 5.8. The ghosting method (method 2) seems to be significantly better for depth perception than the classic transparency (method 1). It is also slightly better than method with fully opaque object (method 3). In this last method, the visibility discontinuities occurs, thus participant is able to decide which label is closer. On the contrary, I expected that the classic transparency would be better than the method with fully transparent object (method 0). The results may be distorted by tipping on the user side.

To support the statement, I computed the confidence interval for the difference between two overlapping confidence intervals [27]. The ghosting method (method 2) is for the depth perception significantly better than method 0 (16.58 ± 12.4574) and better than method 1 (18.655 ± 14.7765).

I also provide a mean value along with a 95% confidence interval for the completion time for depth perception tasks. In case of fully transparent object (method 0), the mean value is estimated to 7420.843 milliseconds (ms) with confidence interval (6671.373, 8254.51). Classic semi-transparency (method 1) has mean value 9605.451 (7969.221, 11577.629) ms and ghosting (method 2) has mean value 8023.163 (6715.066, 9586.077) ms. Finally, the fully opaque object (method 3) takes in mean 4458.867 (3753.035, 5297.445) ms.

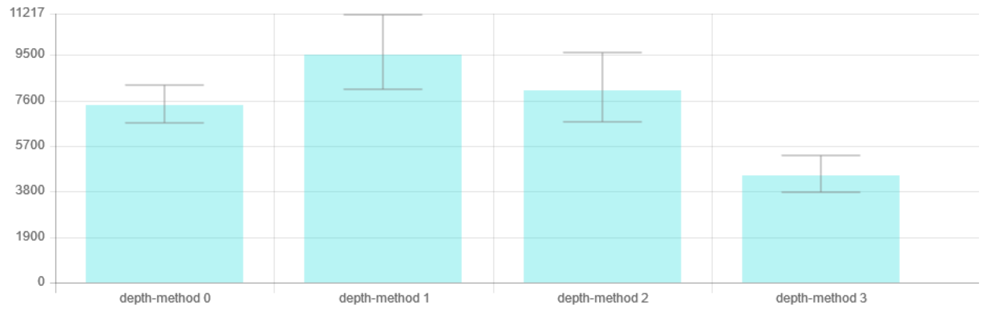


Figure 5.9: Completion time of depth perception test for different methods.
method 0: fully transparent object
method 1: classic semi-transparency
method 2: ghosting with shape and depth visual cues
method 3: fully opaque object

In figure 5.9, the completion time for the depth perception test is shown. The completion time for the method with fully transparent object is slightly higher than I expected. Resulting from interviews after the perception test, it is because the participants were confused whether they understand the task correctly since there is no visual cues to support the depth perception. The most time consuming method is the classic semi-transparency. This is an expected result. The participants try to figure out which of these labels is closer. But there is no significant visual cue that should support the decision. The ghosting is not significantly better in completion time.

Text

From both parts of the experiment the text readability data was collected and evaluated together. I evaluate only whether the label is rewritten completely correct. If not, the error rate increases. I provide a mean value along with a 95% confidence interval for each method. In case of fully transparent object (method 0) the mean error rate is 5.295% (1.14%, 9.45%), for classic semi-transparency (method 1) the mean error rate is 7.705% (2.19%, 13.22%) and for the ghosting (method 2) it is 6.145% (0.25%, 12.04%). Finally, in case of fully opaque object (method 3) the mean error rate is 97.41% (94.82%, 100%).

The results are shown in figure 5.10. As expected, the worst outcome is from scene where the object is fully opaque (method 3). The labels are occluded by the object. Errors that occurred in case of method 0, where object is fully transparent, I assign to misspelling errors since the text is clearly readable.

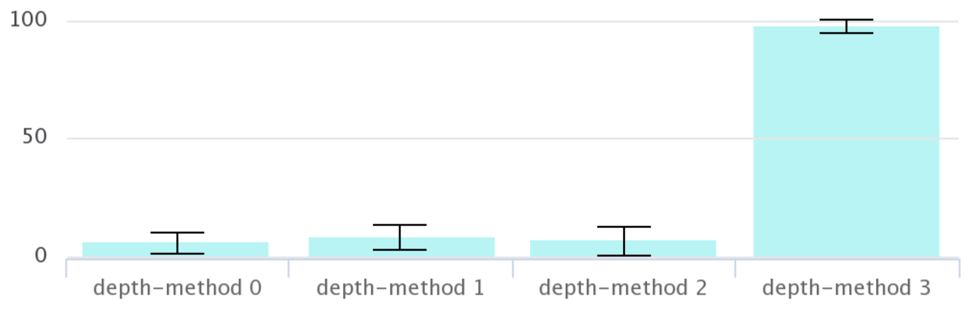


Figure 5.10: Text readability error for different methods.
method 0: fully transparent object
method 1: classic semi-transparency
method 2: ghosting with shape and depth visual cues
method 3: fully opaque object

I cannot say that the ghosting is significantly worse or better (at the probability $p < 0.05$ level) than classic semitransparency because of the large overlapping of their confidence intervals.

According to these results, I believe that I can say, that my assumption that ghosting method with additional depth and shape visual cues would be suitable for all of the presented tasks was correct. We can see that even on monoscopic displays it is possible to enhance the shape and depth perception of semi-transparent objects. For a further understanding of possibilities with shape and depth perception tasks on semi-transparent objects, it will be important to investigate also other factors, for example the perspective or other enhancement methods.

Conclusion

Thanks to this work, I have been studiing several approaches not just for enhancement of visual cues, but also for rendering of semitransparent objects in general. Semi-transparency is apparently a very important visual channel. During the research, I was dealing with lack of perceptually motivated studies that would be comparing feature enhancement methods to each other. Moreover, since the existing implementation is hardware-accelerated and thus interactive, another restriction for methods has been determined.

I provided an overview of different approaches to non-photorealistic visualization, that were created to enhance shape, depth or depth order perception. I implemented several enhancement method and combination of visual cues using Java, OpenGL and GLSL technologies. I tested these methods on prototype models.

I have also created a generator, that allows the user to export sequence of images of rendered scene along with their definition in JSON format. This output can be passed to the test application that was designed and implemented along with it. The test application allows us to test perception of the shape of the surface using gauge figure task, perception of depth and text readability tasks.

Finally, I have designed a perceptual user study that should compare the implemented methods. I prepared test scenes that consist of protein 3D meshes and labels. Study was performed remotely using the web service and web technologies. The user study was performed to verify the whole process that involves the image and scene description export, functional test application and collecting and evaluation of result data. There were no problems during the study. Only problem have been encountered in the application for collecting and evaluating data when calculating confidence intervals for the error rate. Thus these confidence intervals were additionaly calculated.

The results from the perceptual user study show that the ghosting method with depth and shape visual cues are significantly better than classic transparency in both shape and depth perception tasks.

Future Work

The application for visualization of visual cues and their combination is easily expandable by other enhancement methods. Combinations of other visual cues seems to be perspective for future work. Since the test application is provided along with possibility to export scenes, it is possible to use it for other perceptually motivated studies with other methods.

Bibliography

[1] Žára, J.; Beneš, B.; Sochor, J.; et al. *Moderní počítačová grafika.* Computer Press, 2004.

[2] Gooch, A.; Gooch, B.; Shirley, P.; et al. *A Non-photorealistic Lighting Model for Automatic Technical Illustration.* 1998, doi: 10.1145/280814.280950. Available from: `http://doi.acm.org/10.1145/280814.280950`

[3] Lawonn, K.; Preim, B. *Feature Lines for Illustrating Medical Surface Models: Mathematical Background and Survey.* arXiv 1501.03605,, 2015.

[4] Nienhaus, M.; Döllner, J. *Blueprints - Illustrating Architecture and Technical Parts using Hardware-Accelerated Non-Photorealistic Rendering.* 2004.

[5] Diepstraten, J.; Weiskopf, D.; Ertl, T. Transparency in Interactive Technical Illustrations. *Computer Graphics Forum*, volume 21, no. 3, 2002: pp. 317–325, ISSN 1467-8659, doi:10.1111/1467-8659.t01-1-00591. Available from: `http://dx.doi.org/10.1111/1467-8659.t01-1-00591`

[6] Zhang, L.; He, Y.; Xia, J.; et al. *Illustrative Context-Preserving Volume Rendering.* Eurographics Association.

[7] Interrante, V.; Fuchs, H.; Pizer, S. M. Conveying the 3D shape of smoothly curving transparent surfaces via texture. *IEEE Transactions on Visualization and Computer Graphics*, volume 3, no. 2, Apr 1997: pp. 98–117, ISSN 1077-2626, doi:10.1109/2945.597794.

[8] Preim, B.; Baer, A.; Cunningham, D.; et al. A Survey of Perceptually Motivated 3D Visualization of Medical Image Data. *Computer Graphics Forum*, volume 35, no. 3, 2016: pp. 501–525, ISSN 1467-8659, doi:10.1111/cgf.12927. Available from: `http://dx.doi.org/10.1111/cgf.12927`

[9] Joshi, A.; Qian, X.; Dione, D. P.; et al. Effective visualization of complex vascular structures using a non-parametric vessel detection method. *IEEE Trans. Vis. Comput. Graph.*, volume 14, no. 6, 2008: pp. 1603–1610, doi: 10.1109/TVCG.2008.123. Available from: `http://dx.doi.org/10.1109/TVCG.2008.123`

[10] Kersten-Oertel, M.; Chen, S. J.; Collins, D. L. An Evaluation of Depth Enhancing Perceptual Cues for Vascular Volume Visualization in Neurosurgery. *IEEE Transactions on Visualization and Computer Graphics*, volume 20, no. 3, Mar. 2014: pp. 391–403, ISSN 1077-2626, doi: 10.1109/TVCG.2013.240. Available from: `http://dx.doi.org/10.1109/TVCG.2013.240`

[11] Ritter, F.; Hansen, C.; Dicken, V.; et al. Real-Time Illustration of Vascular Structures. *IEEE Transactions on Visualization and Computer Graphics*, volume 12, no. 5, Sept 2006: pp. 877–884, ISSN 1077-2626, doi: 10.1109/TVCG.2006.172.

[12] Zheng, L.; Wu, Y.; Ma, K.-L. *Perceptually-Based Depth-Ordering Enhancement for Direct Volume Rendering.* Los Alamitos, CA, USA, 2013. Available from: `doi.ieeecomputersociety.org/10.1109/TVCG.2012.144`

[13] Hauser, H.; Mroz, L.; Bischi, G.-I.; et al. *Two-level volume rendering - fusing MIP and DVR.* Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Apr. 2000, human contact: technical-report@cg.tuwien.ac.at. Available from: `https://www.cg.tuwien.ac.at/research/publications/2000/Hauser-2000-TwoX/`

[14] Rautek, P.; Bruckner, S.; Gröller, M. E. *Semantic Layers for Illustrative Volume Rendering.* Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Apr. 2007, human contact: technical-report@cg.tuwien.ac.at. Available from: `https://www.cg.tuwien.ac.at/research/publications/2007/TR-186-2-07-05/`

[15] Lebedeva, A. *Web application collecting and evaluating data from user experiments.* Master's thesis, Faculty of Electrical Engineering, Department of Computer Science, Czech Technical University in Prague, 2018.

[16] Everitt, C. *Interactive Order-Independent Transparency.* NVIDIA OpenGL Applications Engineering. Available from: `http://gamedevs.org/uploads/interactive-order-independent-transparency.pdf`

[17] Čmolík, L. *Interactive Illustrative Visualization of 3d Models.* PhD thesis, Faculty of Electrical Engineering, Czech Technical University in Prague, July 2011.

[18] Bruckner, S.; Grimm, S.; Kanitsar, A.; et al. *Illustrative context-preserving exploration of volume data.* Proc. EuroVis '05, pp. 69-76,, 2005.

[19] Kindelmann, G. L.; Whitaker, R. T.; Tasdizen, T.; et al. *Curvature-based transfer functions for direct volume rendering: Methods and applicaitons.* Proc. Visualization (2003), IEEE, pp. 513–520. doi: 10.1109/VISUAL.2003.1250414.

[20] Bruckner, S.; Gröller, E. *Style transfer functions for illustrative volume rendering.*

[21] Zhang, L.; He, Y.; Xia, J.; et al. *Real-time Shape Illustration Using Laplacian Lines.*

[22] Bruckner, S.; Gröller, M. E. *Enhancing Depth-Perception with Flexible Volumetric Halos.*

[23] Carnecky, R.; Fuchs, R.; Mehl, S.; et al. *Smart Transparency for Illustrative Visualization of Complex Flow Surfaces.* Piscataway, NJ, USA, May 2013, doi:10.1109/TVCG.2012.159. Available from: `http://dx.doi.org/10.1109/TVCG.2012.159`

[24] Bernhard, M.; Waldner, M.; Plank, P.; et al. The Accuracy of Gauge-Figure Tasks in Monoscopic and Stereo Displays. *IEEE Computer Graphics and Applications*, volume 36, no. 4, July 2016: pp. 56–66. Available from: `https://www.cg.tuwien.ac.at/research/publications/2016/bernhard-2016-gft/`

[25] Jan J. Koenderink, A. M. L. K., Andrea J. van Doorn. Surface perception in pictures. 1992. Available from: `https://pdfs.semanticscholar.org/0c6b/682dc1597fb9621c29fcd605c1cfb0757256.pdf`

[26] Kantowitz, B.; Roediger, H.; Elmes, D. *Experimental Psychology.* International student edition, Cengage Learning, 2008, ISBN 9780495595335. Available from: `https://books.google.cz/books?id=2-5VL8PHLsIC`

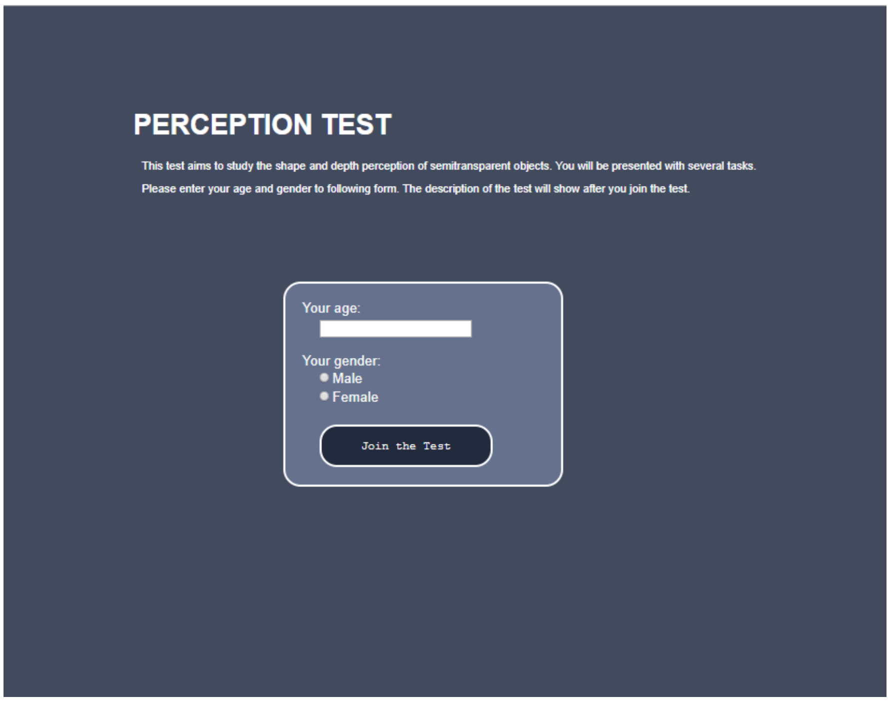
[27] Confidence Intervals for the Difference Between Two Population Proportions or Means. Available from: `https://onlinecourses.science.psu.edu/stat100/node/57/`

[28] MeasuringU. MeasuringU: One Sample Proportion Calculator. Available from: `https://measuringu.com/onep/`

Appendix A

Test Application Walkthrough

Let me introduce the test application walkthrough. Firstly the participant needs to join the test (figure A.1). When he join the test, the `participantIndex` and `modelIndex` is obtained from server local storage. After that counterbalancing is performed based on `participantIndex`.



PERCEPTION TEST

This test aims to study the shape and depth perception of semitransparent objects. You will be presented with several tasks.

Please enter your age and gender to following form. The description of the test will show after you join the test.

Figure A.1: Page for participant registration

A.1 Shape test and scene for practice

After registration, the test description page (figure A.2) is displayed. The participant is acquainted with whole shape test process. Moreover, he can try on test scene the gauge figure manipulation. Only single test scene for required test is available. This scene slightly differs from scenes in experiments. After practicing on test scenes, he can start the test by pressing the button. The shape perception task by gauge figure method is presented in figure A.3.

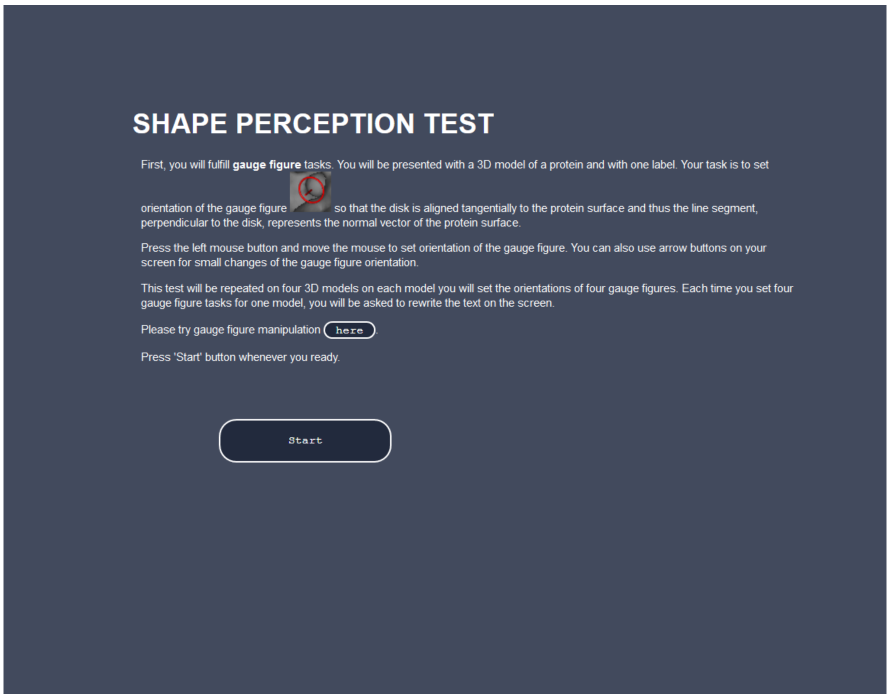
The sequence of models and methods are controlled by counterbalancing described in section 5.3. Redirection between models and to another experiment part is handled by `redirectToNextModel(path)` function.

A.2 Depth test and scene for practice

After finishing the shape test, the depth test description page (figure A.4) is displayed. The participant is acquainted with whole depth test process. Again, he can try on test scene the depth test task. Only single test scene for required test is available. This scene slightly differs from scenes in experiments. After practicing on test scene, he can start the test by pressing the button.

Again, the sequence of models and methods are controlled by counterbalancing. Redirection between models and to another experiment part is handled by `redirectToNextModel(path)` function. The depth perception test is presented in figure A.5.

After finishing the depth test, the whole test is completed.



SHAPE PERCEPTION TEST

First, you will fulfill **gauge figure** tasks. You will be presented with a 3D model of a protein and with one label. Your task is to set orientation of the gauge figure so that the disk is aligned tangentially to the protein surface and thus the line segment, perpendicular to the disk, represents the normal vector of the protein surface.

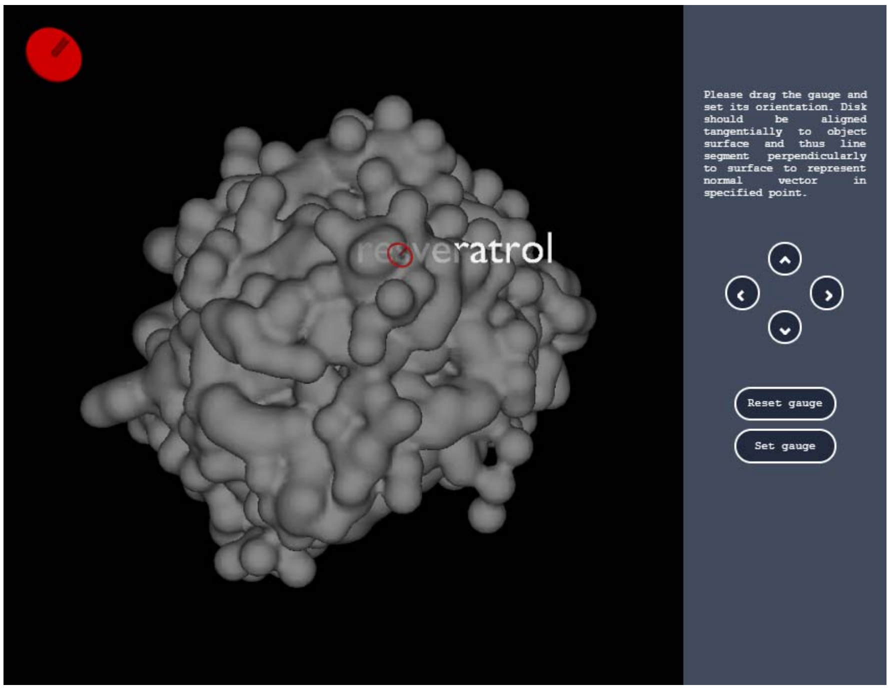
Press the left mouse button and move the mouse to set orientation of the gauge figure. You can also use arrow buttons on your screen for small changes of the gauge figure orientation.

This test will be repeated on four 3D models on each model you will set the orientations of four gauge figures. Each time you set four gauge figure tasks for one model, you will be asked to rewrite the text on the screen.

Please try gauge figure manipulation here.

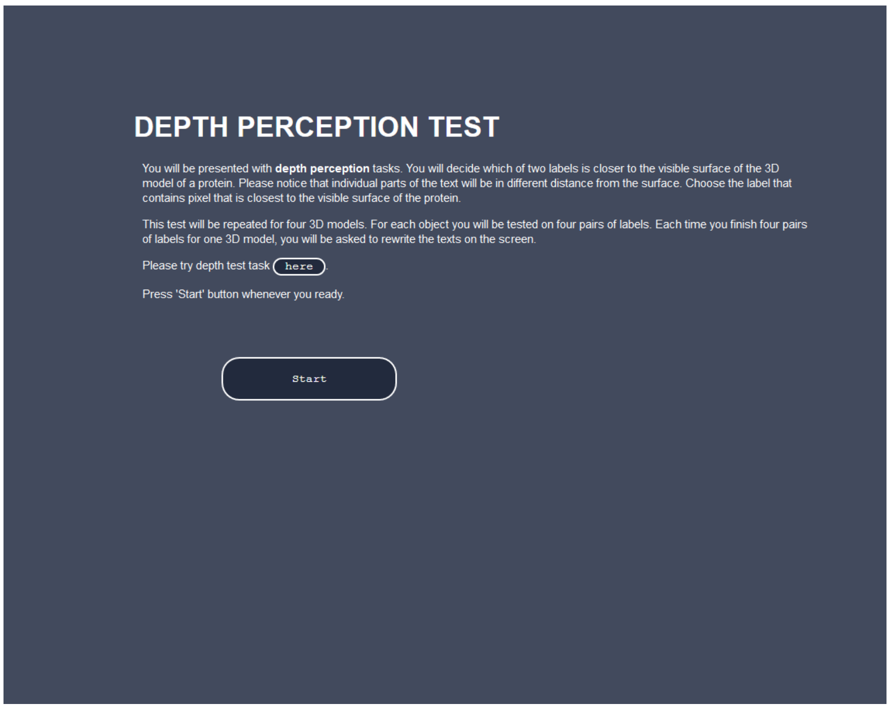
Press 'Start' button whenever you ready.

Figure A.2: Shape test description page



Please drag the gauge and set its orientation. Disk should be aligned tangentially to object surface and thus line segment perpendicularly to surface to represent normal vector in specified point.

Figure A.3: Shape perception test - gauge figure task



DEPTH PERCEPTION TEST

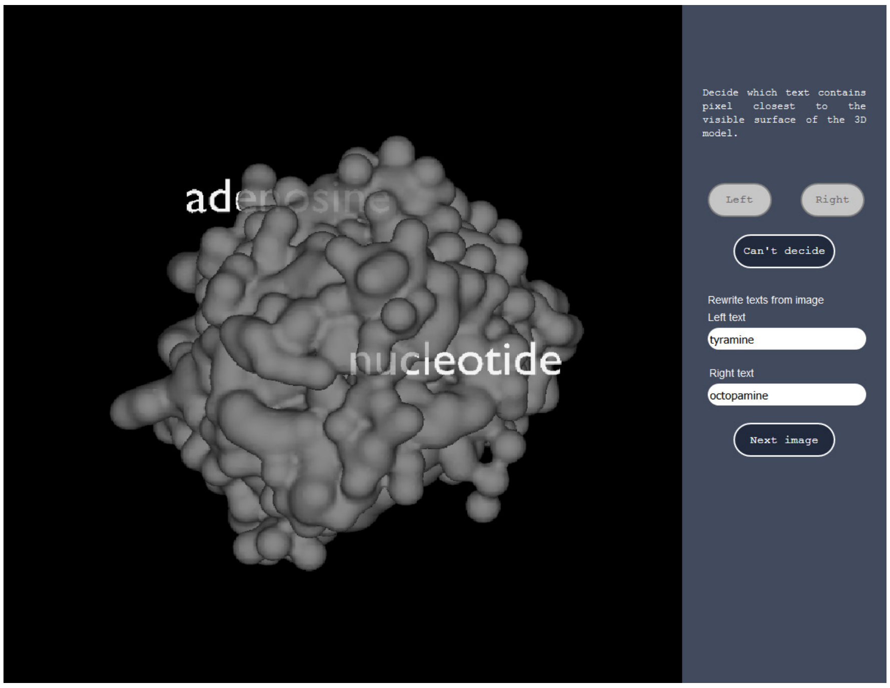
You will be presented with **depth perception** tasks. You will decide which of two labels is closer to the visible surface of the 3D model of a protein. Please notice that individual parts of the text will be in different distance from the surface. Choose the label that contains pixel that is closest to the visible surface of the protein.

This test will be repeated for four 3D models. For each object you will be tested on four pairs of labels. Each time you finish four pairs of labels for one 3D model, you will be asked to rewrite the texts on the screen.

Please try depth test task here.

Press 'Start' button whenever you ready.

Figure A.4: Depth test description page



Decide which text contains pixel closest to the visible surface of the 3D model.

Rewrite texts from image

Left text: tyramine

Right text: octopamine

Figure A.5: Depth perception test