Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Improving Spatial Perception in AR/MR Through Custom Shading Techniques

by Kexin Guo

A dissertation submitted to the University of Dublin,
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science (Augmented & Virtual Reality)
Trinity College Dublin, Ireland

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the Universitys open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Kexin Guo

September 5, 2021

Acknowledgments

Thanks to my dissertation supervisor, Professor. John Dingliana for helping me over the past few months. He helped me clarify the research goals and development ideas of the entire project through weekly meetings. In order to help me achieve my research goals, he helped me set up a weekly task plan, and put forward suggestions on the results of my phased tasks and the next revision direction. Because of John's guidance, I was able to achieve today's results.

At the same time, I would like to thank Dr. Michael Manzke, Strand Leader of Augmented & Virtual Reality, and thank Michel for his understanding of my special situation, and thus agreed to my extension application, so that I can better complete my dissertation paper.

Finally, I would like to thank my friends and family for their support. They cheered me on when I encountered difficulties and gave me the courage and determination to overcome difficulties.

Abstract

Augmented reality (AR)/mixed reality (MR) technology is a technology that can integrate virtual feature information with the real world. Its wide application provides users with a new way to obtain additional information, such as using AR applications to observe the internal structure information of the medical human body model. However, in this implementation process, it is easy to make users confused about the spatial relationship between the virtual object and the real object. That is, the perceived depth of an AR object deviates from the intended depth and cannot achieve the effect that the AR object appears inside the real object. This problem is also called lack of depth perception. For improving the depth perception in AR, the existing related literature mainly conducts research and experiments from three directions: translucent projection objects, creating virtual interior space of occlusion objects, and enriching depth cues. After the comparison, the construction of the virtual interior space method has the most effective depth perception between them.

But even so, some current excellent research methods still have some shortcomings. For example, the virtual item may obscure the characteristic information of the real environment, and the shader programming language they used is relatively basic and not easy to extend in the future. Therefore, the main purpose of the project is to use a new custom shader method to achieve the excellent cut-away effect in the previous paper and to enrich the depth clues of the scene on this basis. I used the C++ custom shader programming method in Unity to reimplement and improve some of the cut-away effects. The project mainly uses two different Render Pipeline rendering methods based on the Built-in Render Pipeline and Universal Render Pipeline in Unity to achieve the effect. Through the comparison and analysis of the two experimental results, it is found that the cut-away effect of the Built-in Render Pipeline is currently the most real and effective. It can easily allow users to perceive the correct spatial relationship between objects. The main contribution of this dissertation is using the Unity engine to achieve the cut-away effect. It improves the user's perceived spatial relationships between real objects and projected MR objects, and also provides some custom cut-away feature options for users to freely adjust. The implementation based on the Unity engine makes the project easier to introduce into other AR applications, furthermore, it also facilitates the later function expansion.

Keywords: Augmented reality; Spatial perception; Custom shading; Cut-away

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation mainly deals with how to increase the performed spatial relationships between the real object and the projected MR virtual object. This chapter will analyze the technical background and main advantages and disadvantages of the research goals, and formulate the main realization goals of the project based on the results of the analysis.

1.1 Background

Augmented reality (AR) is a technology that uses multimedia, real-time tracking, and sensing technologies to fuse virtual information with the real world. It can change the order and difficulty of information acquisition through simulation and virtual display, excavate deep-level information that people cannot easily obtain in daily life, and directly assign it as additional information to surface objects in reality to achieve information Maximum effect. In addition, the user can also interact with the rendered virtual object in real-time to obtain different information. For example, when purchasing furniture through AR, the user are allowed to rotate, switch, and move the furniture to ensure that the user can observe the product information in all directions, and at the same time make the product selection by the real room situation.

These characteristics of augmented reality make it widely used as a means to provide users with additional information in reality, such as game production, industrial design, medical treatment, home decoration, and teaching demonstrations. According to the statistical results of the existing research literature in Zhou.[19], to enhance the user experience and to integrate virtual objects into the real world more realistically, according to the different types of application scenarios, researchers mainly develop and improve the effects of augmented reality in the following areas: Tracking techniques, Interaction techniques, Calibration and registration, AR applications and Display techniques, etc. For example, Reifinger.[13] proposed an automatic gesture recognition system that can distinguish be-

tween static and dynamic gestures, which greatly shortens the execution time of interactive tasks; In terms of tracking techniques, to improve the accuracy of the tracker while reducing the latency, Comport.[3] proposed a 3D model tracking algorithm for the monocular vision system. The advanced theories proposed above all optimize the real effects of augmented reality to some extent.

However aside from interactivity and realism, there remain some issues that are not yet fully solved in common AR applications, for instance ensuring that users correctly perceive the intended spatial positions and scales of virtual objects in relation to the real world.

1.2 Motivation

Although the current augmented reality technology is gradually becoming mature, there are still some visual errors in practical applications. For example, some fields rely heavily on depth cues such as medical and industrial design, in these areas users usually need to achieve AR display effects of the internal structure of some items. At this time, it is necessary to achieve the effect of a virtual object being contained inside a real object. For instance, in industrial design, the virtual internal components of a car model need to be shown to the user in the form of a "perspective view", so that the user can avoid the cumbersome steps of dismantling the model from the exterior to the interior.

However, in reality, virtual object in AR displays are drawn as pixels overlaid on top of the real entity, so the biggest difficulty in the process of achieving object perspective is how to ensure the correctness of the perceived depth relationship between the virtual object and the real one. If the depth relationship between the objects is wrong, as shown in the image of a demo screen in the video I intercepted from ErickMendezARWork[5]. The car in the Figure 1.1 is in a visually wrong position at this time because it is well known that the junction of the two planes of the box is a geometrically convex state. In this state, it is theoretically impossible to place a car smoothly. Therefore, the AR effect in the picture looks very unreasonable; in addition, we can imagine that the original AR car is set in the interior of the box, but because of the lack of depth clues, it looks like it is floating diagonally above the box. This means that the projected AR object deviates from our ideal position and looks out of sync with the real world. More than that, it will also confuse the occlusion relationship between objects.

In addition, some rendering details will exacerbate this visual unrealism and depth perception errors. Such as missing object shadows, the appearance of virtual objects that do not match reality, and poor visibility of objects. Only by improving the above aspects can accurate perceived spatial relationships of virtual objects be achieved.

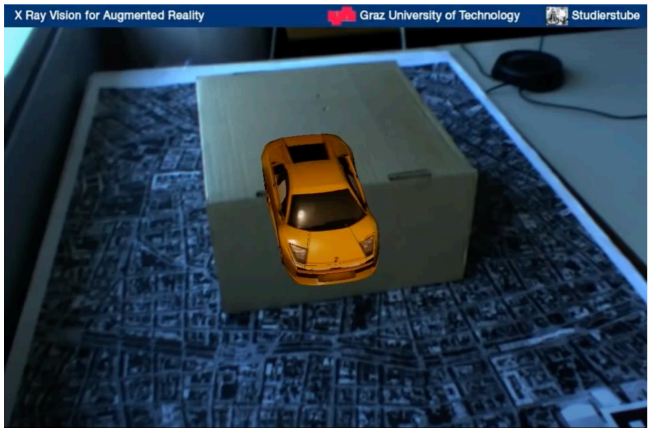


Figure 1.1: Example for wrong spatial perception[11]

1.3 Existing research

Because of the several major reasons that can confuse the spatial perception of virtual objects: wrong occlusion relationship, the appearance that does not match the external environment, and lack of depth cues, various types of solutions have been developed. The first method is to render the virtual objects that you want to put inside the occluder in a semi-transparent style to show a correct perceived spatial relationship. The disadvantage of this approach is that the translucency of the object will make it difficult for users to obtain effective information from the object. Because after rendering, the virtual item is very likely to be affected by the background image received by the camera, making it difficult to identify the original feature. To make up for this shortcoming, Fukiage[7],Roxas[14] et al. made improvements from the aspect of Visibility-Based Blending. Among which Fukiage[7] first proposed the blending model of Visibility-Based Blending, by continuously optimizing the blending parameter α in the blending process, the virtual objects can achieve the visualization effect of any target transparency level, and also strengthen its essential characteristics; Roxas[14] added the occlusion effect caused by the environment on the object on this basis and combined the rendering process with the input foreground probability map to adjust the local transparency of the object. While preserving the initial features, it realizes the soft transition processing on the boundary edge of the virtual occluded object and the real occluded object, which effectively expresses the occlusion relationship between objects.

The core content of the second method is to construct a fake geometric internal space of the occluder and place the target virtual item in the fake space to form an inclusive relationship, which is intended to create a visual perspective effect. The representative ones are Wimmer[18],Hansen[8] et al. Wimmer[18] proposed a user-defined clipping technology for medical applications. Users can customize the shape and position of the clipping volume (human cross-sectional area), and then use it

in conjunction with ghosting of the skin to achieve a correct perceived spatial relationship. Hansen[8] Combine AR and Reality Capture technology to generate a 3D underground scene model as the real one through a visual point cloud model, and then use Unity's occlusion shader to cover it, and finally achieve a genuine surface profile effect.

The third method is mainly used as an auxiliary solution to increase the depth cues of the model and improve the depth perception effect of the model. It mainly includes adding appropriate shadows to the virtual occluded objects and synchronizing the style of the object and the environment. The advanced shadow technologies are: Liu[10] proposed an ARShadowGAN model that can directly generate object shadows without illumination estimation; Wei[16] designed a method that can simulate dynamic shadows of AR objects based on detection and analysis of shadows in real-time video's frame. Another way to increase realism is to use special effects with distinct styles to act on both virtual objects and real environments to eliminate the sense of conflict between them. A representative article Fischer[6] created a Stylized augmented reality algorithm that applies artistic image rendering to both camera images and graphical objects and unifies the real levels of the two.

The three methods summarized above can enhance the spatial perception of objects to varying degrees. In a comprehensive comparison, the second method can fundamentally improve depth perception, as it has a more complete geometric spatial relationship, and can display richer features of occluded objects than the first method. So in the next part, I will focus on the research and realization of the cut-away effect of the real occluder.

1.4 Research objectives

After summarizing the problems that have occurred, we learned that if we want to achieve a good depth perception effect, we need to meet the following basic requirements.

- The visible distance presented by the virtual object projected on the plane needs to be farther than the visual distance of the physical plane;
- Maximize the information of virtual objects based on preserving the original features on the real plane as much as possible;
- The visual effect of the appearance of the occluded object should be as consistent as possible with the external environment color (such as lack of light, blur, etc.)

Therefore, to solve the above-mentioned problems, the main purpose of this project is to create a more realistic mask that matches the physical environment while ensuring that the information presented by the virtual objects inside is maximized. Based on the use effects of various masks, the random

hole mask is the most natural and real. So I plan to make improvements based on this to increase the authenticity of the mask.

The main contributions of this dissertation are:

1. Use the built-in rendering pipeline combined with stencil buffer and a custom shader to achieve the basic AR cut-away effect, and show the correct spatial relationship between the occluded object and the occluded object;

2. In addition to achieving the essential cut-out effect, additional adjustable detail features are added to it through the Universal Rendering Pipeline in Unity, so that its perspective effect is more compatible with the external environment;

3. In addition to maintaining the environmental feature information, the maximum display of the detailed info of the virtual occluded object

1.5 Overview of structure

To clearly explain the difficulties, technical points and implementation process of this project, the structure of this dissertation is as follows:

In the second chapter, we will introduce some academic research papers related to depth perception, perspective relationship, and mask realization involved in this project, and conduct a thorough analysis. After comparing and evaluating each article, the best method to achieve the correct spatial relationship effect is summarized as the guiding direction for the subsequent research and practice of the project.

The third chapter mainly elaborates the two methods I used in the project, including the platform used for its realization, the main technical methods, and specific steps, etc., and gives a detailed explanation of the technical difficulties in each approach. Finally, the two methods are comprehensively evaluated and summarized.

Chapter four shows the final results of the two implementation methods mentioned in the previous chapter and makes a comprehensive comparison with the other three practice effects. Judge the effectiveness and limitations of each way by analyzing the comparison results.

Chapter five gives a comprehensive summary of this dissertation, including the contribution and shortcomings of the project. In addition, it also mentioned the further research and improvement that needs to be carried out in the future for the deficiencies.

Chapter 2

Literature review

After the above analysis of three feasible solutions for improving depth perception, we determined that the main research and practice direction of this project is to get the correct perceived spatial relationship in the AR scene. The perceived spatial relationship is just a general term here, which is a general expression of a visual effect that allows the user to see the inside of the object through the surface. For a better understanding, we can simply compare it to the X-ray effect in medical treatment, that is, through the use of various advanced technical means to visually present the characteristic information (structure, etc.) inside the target object to the user. But the final visual effect here is not limited to the traditional X-ray 2D profile form. In this chapter, I will focus on some literature studies that are closely related to enhancing the AR perspective effect, and roughly divide them into categories according to their main implementation technologies, and then conduct a systematic comparison and evaluation.

2.1 Depth Perception & Depth Cue

Depth perception is also called the three-dimensional perception of distance perception. It is expressed as the human eyes ability to perceive three-dimensional space, such as the understanding of objects distance and proximity, as well as the perception of shapes. In biology, since the visual image of a single eye is a two-dimensional plane, our human body's judgment of depth generally depends on the visual difference between the two eyes. However, for AR applications, except the specific AR head-mounted displays that use binocular cameras to simulate the observed effect of the human eye. General mobile device AR applications usually rely on a single camera of a mobile phone/computer to observe the rendered AR scene. So when we cannot use binocular vision, another way to realize spatial perception is to use monocular vision, supplemented by sufficient depth cues to make judgments. Our starting point for enhancing the AR depth perception effect in this project is to add effective depth

cues to the constructed AR scene.

The academic understanding of depth cues is the information source of the human brains interpretation of depth, which can generally be divided into binocular cues and monocular cues. Binocular cues emphasize the feedback information generated by the coordinated activities of the eyes. This information mainly includes binocular parallax, binocular width, and motion parallax; monocular cues indicate the characteristics of visual stimuli in visual space perception, including relative size, occlusion relationship, Line perspective, brightness and shadows, and texture gradients. Here we skip the binocular cues mainly used in binocular cameras to introduce the monocular depth cues in detail.

- The relative size of the object mainly uses the principle of near-large and far-small. When the thing is closer to the visual starting point, it will appear larger, and vice versa, it will appear smaller. But in an AR scene that combines virtual and real, it is difficult for us to judge the distance change only based on the size of the object, so we usually make judgments on this basis with other depth clues;

- The occlusion relationship is that when two objects partially overlap in the current line of sight direction, the closer object will hinder the presentation of the far object to varying degrees. The degree of obstruction is related to the transparency and material of the closer one; The occlusion effect is usually combined with the relative size to act on the scene. Take the example image Figure 2.1 below as an example. Both pictures have the same two circles, one large and one small. The difference is that the small yellow circle on the left covers part of the large blue circle, while the blue circle on the right covers the yellow circle. We can see that there is a conflict between the occlusion relationship and the relative size of the left picture, so it is difficult to judge which color circle is closer to the front. On the contrary, we can find the blue color circle from the right picture have a closer spatial relationship than the yellow circle.

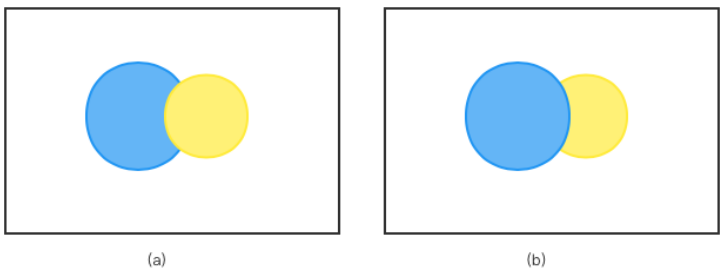


Figure 2.1: Example for relative size with occlusion.(a) has a conflict between depth cues,(b) is the right relationship.

- Line perspective is mainly related to the proportion of objects in the visual picture. The closer the object, the larger the proportion. So we will find that the parallel lines converge into a point when they extend far away. The more lines converge on this point, the farther the perceptual distance will be. However, the general AR scenes are mostly close-range scenes. On this basis, we only need to ensure that the geometric shape of the virtual object conforms to the geometric perspective effect of the three-dimensional space;

- Brightness and shadows can help us perceive the volume, texture, and shape of objects. The actual application in computer vision is the depth map (distance image). Its principle is to output the depth distance value from the image collector to each point in the scene as the pixel value to the image. As shown in the Figure 2.2, depending on the pixel output settings, the distance from far to near can be expressed as from black to white or from white to black. In addition to using the depth map in the AR scene, we can also add the shadow cast by the occluder to the occluded object to express the perspective relationship in space;

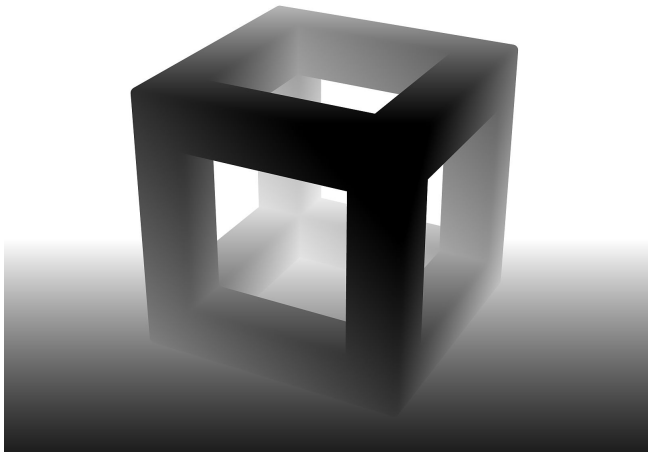


Figure 2.2: Example depth map from Wikipedia

- The texture gradient is expressed as the distribution of the same elements on the plane will look denser with the distance. As shown in the Figure Figure 2.3,This clue is mostly used to observe the extension of the plane with the material in the space.

After understanding the above-mentioned monocular depth cues, we can selectively add them according to specific scenes and missing elements to help users gain better spatial perception.

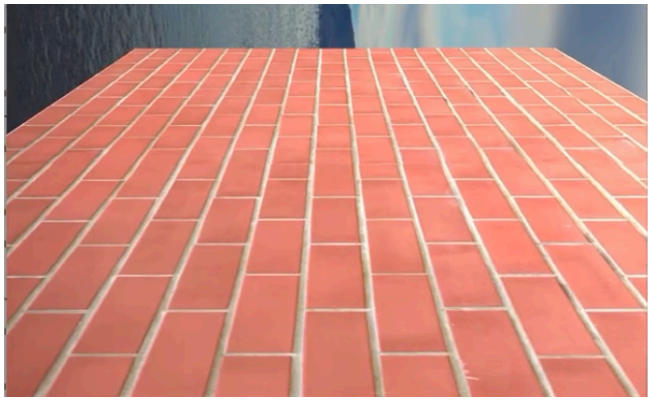


Figure 2.3: Example texture gradient effect

2.2 Simple Cut-away

To make AR objects appear inside the real occluder, the most intuitive way is to simulate the cross-sectional interior scene of the real object through depth cues around the virtual object. The so-called cross-sectional effect is the cut-away effect. As shown in the Figure 2.4, this cut-away effect preserves the complete outline of the external structure of the object while exposing the internal structure of the parts to the field of vision. In this way, the user can judge what the object is from the basic external contour, they can also observe the internal structure information of the target, and can correctly find the containment relationship between the two. It is conceivable that if this 2D cross-sectional view is transformed into a virtual interior scene in an AR scene, it will produce the same auxiliary effect.

Figure 2.4: Example Cut-away effect from Shutterstock

In this way, the virtual object with the internal geometric space environment of the occluder itself has rich and obvious depth cues, and the user can easily conclude the depth relationship between the virtual element and the real element through these artificial depth cues. Except for the Wimmer.[18] and Hansen.[8] which are mentioned in the previous chapter, the representative articles also include Broecker.[2].

Broecker.[2] published in 2014, it is one of the earlier articles using user location tracking technology to assist the rendering process, which can help us better understand the false geometric internal

space technology from the principle. It uses View-dependent rendering based on the Spatial Augmented Reality (SAR) system, replacing the previous method of rendering virtual scenes which use the main camera as the visual starting point. After tracing the users position, the purely virtual geometry of the current perspective will be rendered on the projected solid plane. On this basis, projective texture mapping techniques are used to set up the virtual content into the geometric space, and finally formed a cut-away rendering image independently related to perspective.

To further compare the effects of different types of depth cues on the depth perception results, the author rendered two forms of geometric space. One is to combine the pure virtual inside of a box with virtual wall depth, and the other is to attach a texture map (texture gradient depth cue) to the virtual inside of a box. The criterion for judging the influence of the two is the strength comparison between the pure "virtual" depth cues from the virtual content and the "real" depth cues provided by the physical model of the box and the environment. The closer the virtual depth cues are to the strength of the real depth cues, the better the spatial perception effect is obtained.

But its disadvantage is that when the projected image covers the surface of the occluder, it may occlude the important real-world information on the real object, thereby causing the problem of missing information. As shown in the Figure 2.5 – rendering example in Kalkofen.[9], the original black X-shaped mark on the real human body model is blocked by the projection of the virtual heart, and the user may not be able to work because he cannot receive relatively more important mark information.

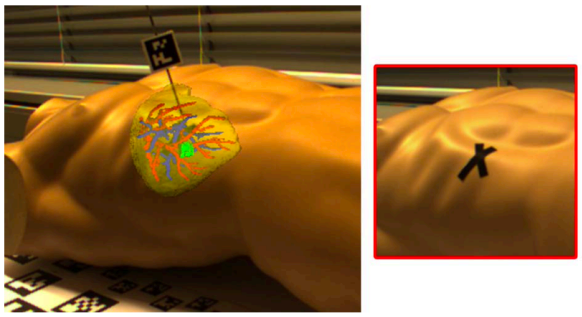


Figure 2.5: The blocked important real-world information[9]

2.3 Cut-away with Mask on Occulder

The main reason for the above-mentioned defects is that the primary and secondary feature information is not distinguished, and the secondary important virtual features erroneously cover the main real features so that the user's attention is forced to divert completely. Therefore, to add virtual information in AR while preserving the real original features and to optimize the depth perception effect, two more

effective solutions have been developed. The first category is to strengthen the importance of the characteristics of actual objects, that is, to add a mask to the real target object to retain or emphasize its important marks. This is not strictly a cut-away method, but as one of the ways to resolve the conflict between real and virtual depth cues, its principle is also applicable to cut-away scenarios.

For example, Kalkofen.[9] proposed an interactive AR application based on Focus+Context (F+C) visualizations technology. The application can provide contextual environment feature information around virtual target features to help users understand the scene, and use the Focus+Context information classification function to provide the occlusion effect of relatively important context information for digital targets. The specific effect is shown in the renderings image Figure 2.6 provided by the author in Kalkofen.[9]. The red wheels and engines are virtual projection features, and the gray and yellow contour lines are real contextual information. By adding contour lines to the context to emphasize the external structure of the car and the occlusion relationship between each other, the user can intuitively understand that the space between these three parts is gray car body-yellow seat-red engine and tires. It not only reserved the original features of the model but also add the virtual features in red.

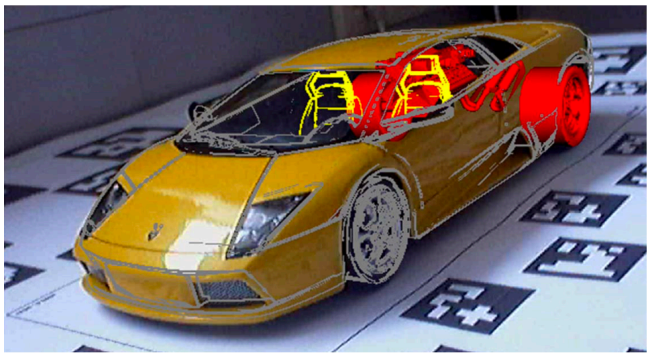


Figure 2.6: Focus+Context visualizations effect[9]

The framework of the algorithm is shown in the Figure 2.7. The creation of F+C visualizations is mainly divided into two steps. The first step is to input data into F+C Filter under the control of the user to classify, which should be the focal part and which should be used as context. In this way, the second level Focus and the second level Context are divided. The second step is to render the divided data with a distinctive and unique rendering style, through the outline to highlight the key features. At the same time, use the alpha mask on the context information to distinguish between primary and secondary so that the user can focus on the 'Focus' area. The advantage of this algorithm is that the original key features will not be lost under the premise of ensuring the correct spatial relationship between the projected object and the real object, and the information will be reasonably maximized. The disadvantage is that when the Context part is mixed with virtual objects, it may cause image

clutter problems and reduce the users perception.

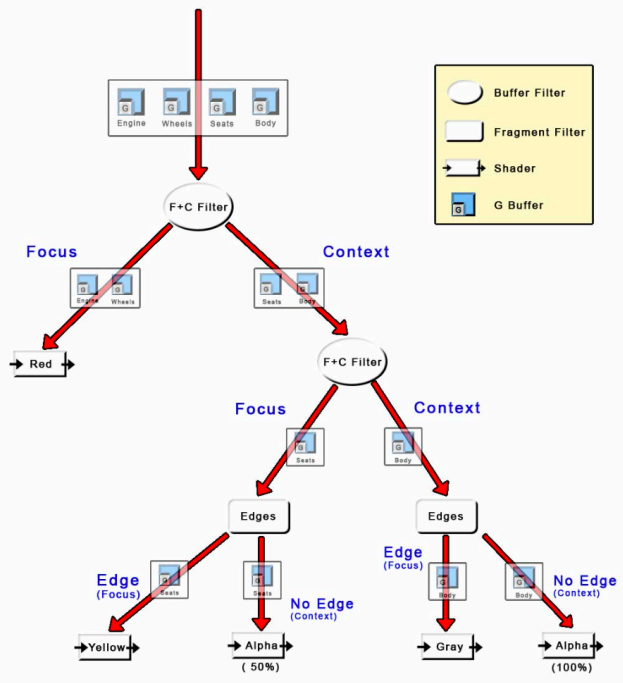


Figure 2.7: Focus+Context visualizations framework[9]

Another method to enhance real features is Zollmann.[20], which proposes a more ideal X-ray visualization technique: Image-based ghosting techniques, which allows users to better understand the depth order between objects in the scene. Its algorithm process is shown in the Figure 2.8. In the Image Analysis component, the depth cues (Edges, Saliency, and Texture) are automatically extracted from the image obtained by the camera through the algorithm, and then these depth cues are combined into a Ghosting Map. The obtained Ghosting Map will be combined with video images and digital objects in the rendering, the Map can also determine which important features in the environment need to be preserved. Finally, a very realistic rendering result of the underground virtual object is obtained.

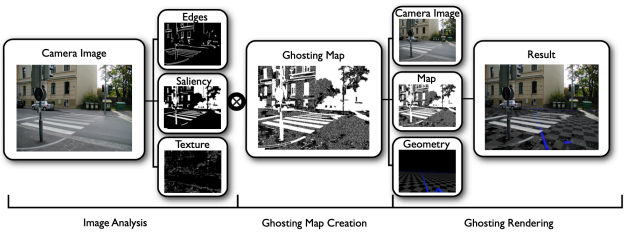


Figure 2.8: Image-based ghosting techniques[20]

The main application scenario of this article is a large outdoor venue, so when acquiring depth

cues from a frame of the video image, all the feature information in the scene is referred to. But if you only collect depth cues for a certain object indoors, it will inevitably collect a lot of useless interference information. Therefore, it is only used as an auxiliary reference article for analysis.

2.4 Cut-away with Mask for Occulded object

In contrast to the above method of adding a mask to the occluder, there is another optimization method, which is to weaken the importance of the features of the virtual object, reduce the exposure degree of the features of the occluded object, and retain the original external features. But although it is to decrease the characteristics of the virtual object, it is not directly rendered as a semi-transparent style as mentioned in section 1.3, but is weakened based on the cut-away effect. In this way, weaken based on the cut-away effect contains the basic geometric spatial depth information, which will have a stronger sense of three-dimensionality compared to the method in section 1.3. Therefore, this method of weakening the virtual features still adopts the technical means of attaching the mask to the occluder. By removing part of the surface of the real object, the AR target looks more like the inside of the object.

2.4.1 Remove surface

The types of masks used can be roughly divided into two types. The first type of mask is to partially remove parts of the real surface related to virtual objects ("digging holes") reasonably so that the inner object looks like it's inside the real object. For example, Otsuki.[12] combined two well-known phenomena pseudo-transparency and stereo-transparency to create a "stereoscopic pseudo-transparency" method. They use OpenGL to add a mask with randomly distributed holes to the surface of the actual object and then placed the internal virtual object in the mask area to achieve a natural internal existence effect. Compare with other video-based stereoscopic display algorithms of the same type, this random point mask can be used as an add-on surface feature to provide a depth cue on a plane with no obvious features. And it can also allow users to perceive the shape and colors of the original surface. The specific effect is as shown in the Figure 2.9 below.

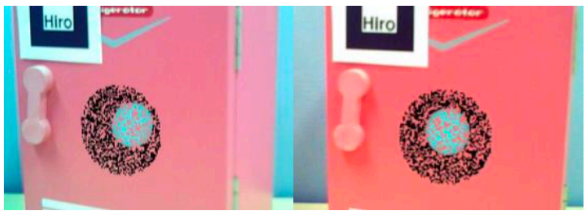


Figure 2.9: Random-dot mask on surface[12]

The author used five different types of masks to conduct a horizontal comparison test. The results show that the rendering effects of random-dot mask and Semi-transparent random-dot mask make it easier for the observer to think that he saw the circle through the surface and feel that the distance of the circle is farther than the surface. But this method still has some shortcomings, that is, there are still some visual gaps between the holes in the mask and the real holes; And due to the occlusion of the holes, the virtual items projected inside will not be completely exposed to the user's vision, so some information will inevitably be lost

2.4.2 Adjust Transparency

The second type of mask is to adjust the transparency of the appearance of part of the masked object (external real object), such as Mendez.[11] and Elmqvist.[4]. Their difference is that Elmqvist.[4] proposed an image-space algorithm to improve the effect of dynamic transparent masking. Dynamic transparency means that the transparent area can be dynamically adjusted according to different viewing angles, instead of fixing the perspective area at a certain point in the plane to restrict the field of view. The principle of implementing this technology is to perform a depth sorting of all objects in the scene and then render them in the order of depth from back to front, and then select the corresponding branch rendering process according to whether the currently rendered object is the target object. If it is a target object, after rendering its basic features and color information, the algorithm will perform the alpha mask rendering step again on the target area where it is currently located. The alpha mask step changes the transparency of the distractor object after the target object rendering order in this area to achieve the transparency mask effect. Otherwise, if the current object is a distractor, it will directly perform the normal rendering step. The final result is shown in the Figure 2.10.

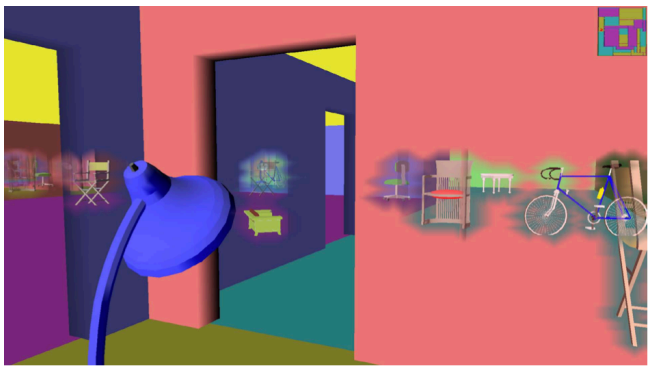


Figure 2.10: Random-dot mask on surface[4]

And Mendez.[11] presents a simple technique by using an importance mask associated with occluders, to enhance the perception of the spatial arrangements in the scene. It makes up for some

of the problems in the previous work, such as the unstable quality of AR rendering results problem. The AR rendering results will be affected by the light in the external environment, the material of the projected plane, and the angle of view, so it is not being able to produce a stable rendering outcome. The main process of this technology is that the user generates exclusive importance masks for specific obstructions through interaction or heuristics in the preprocessing step, as shown in the Figure 2.11. The mask generated by preprocessing only contains the value of the alpha channel, with a value range of 0 to 1, used to determine which areas of pixels should be replaced with virtual occluded objects, and which areas should be retained as the original pixels of the video input. The second step is to assign the mask as a texture map to the occluder, using the blending calculation Equation (2.1) and the value of the mask to determine the rendering ratio of the video feed and the occludes, and finally achieve the appropriate perceived spatial relationship.

$$gl\_FragData[0].rgb = (T_{video}.rgbT_{mask}) + (T_{occluded}.rgb(1.0T_{mask})); \tag{2.1}$$

Figure 2.11: Random-dot mask on surface[11]

The gl_FragData in the formula is the RGB channel value of the current pixel fragment shader, T_{mask} is the alpha single channel value of the mask, T_{video} and $T_{occluded}$ refer to the material RGB value of the video feed and occluder respectively. The final mixed rendering result is obtained by multiplying the material RGB values of the video feed and occluder by different weights(T_{mask} and $1.0 - T_{mask}$). The pixels on the screen are displayed as the background pixel values captured by the AR camera while the alpha channel value of the mask or occluder is 0. When the alpha value is not 0, the pixel RGB value blending operation is performed according to the Equation (2.1), so that the material outside the cut-away area can be eliminated, thereby displaying the original material of the real object.

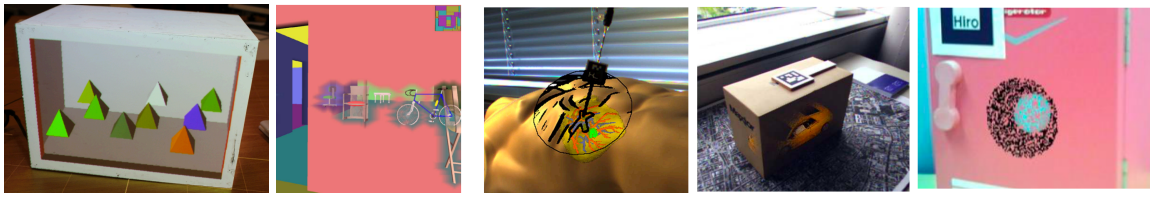
The advantage of this type of transparent mask is that it adopts the projection method of "X-ray vision", so that the virtual imaging will not obstruct the key information in reality, and it can also clearly express the depth relationship between objects. In addition, it only needs to use the single-channel fragment shader in the GPU, which greatly improves the computational efficiency.

However, the shortcomings of the two are that the image-space dynamic transparency algorithm of Elmqvist.[4] is currently only experimented in 3D scenes, and has not been applied to AR scenes, so it is impossible to estimate the effect for mixed with the real world in AR application. Although the edge of the semi-transparent oval mask used in Mendez.[11] can blend naturally with the surroundings, its mask can not be universally used. If you want to achieve the perspective effect on multiple objects, you need to create the mask for each occluder, so the preparation work in the early stage will be relatively complicated.

Looking at the three kinds of mask implementation processes mentioned above, the essence of the mask is to change the transparency of the occluder and replace the pixels of the corresponding area with the pixels of the inner object that you want to project. The only difference is the way the mask is generated and applied.

2.5 Comparison

To find a more suitable solution, using the weakest spatial expression technique Broecker.[2] as the baseline model, some performances of several different algorithms' in-depth perception were compared horizontally (only representing personal subjective evaluation), and the following Table was obtained. Figure 2.12 is the representative result graph of each article.



(a) [2] (b) [4] (c) [9] (d) [11] (e) [12]

Figure 2.12: A comparison of the five representative papers in depth perception: (a)Broecker.[2], (b)Elmqvist.[4], (c)Kalkofen[9], (d)Mendez.[11] and (e)Otsuki.[12]

For the performance criteria of virtual geometric spatial structure information, the degree of retention of important features of the real environment, the ease of obtaining the depth relationship, and the Intensity of the depth relationship perception, the ranking results from left to right represent the effect from good to bad. However, for the complexity of depth cues, a scene containing too complicated

Evaluate	Method
Virtual geometric spatial structure information	[11] > [12] = [9] > [4] > [2]
Degree of retention of important features of the real environment	[11] = [12] > [9] > [4] > [2]
Depth cue complexity	[9] > [4] > [11] > [12] > [2]
Ease of obtaining deep relationships	[11] = [9] > [12] > [4] > [2]
Intensity of deep relationship perception	[11] = [9] > [12] > [4] > [2]

Table 2.1: Method Comparison Result

depth cues may confuse the user's understanding of the scene, while too simple types of cues are not conducive to the correct judgment of the user's spatial relationship. Therefore, the median value of this sorting result can be regarded as the better one.

After a longitudinal comparison of the ranking results of the various indicators in the comprehensive table, we can conclude that the spatial perception effect of [11] is the best among several methods because it achieves the best effect in each standard; secondly, It is [12] and [9], their comprehensive performance is similar, but technically speaking, the technical complexity of [12] is lower than [9]. The difference between them is that [9] will focus more on the detailed accuracy of features, while [12] mainly controls the overall perception effect.

Generally speaking, under the same effect, we will give priority to the framework algorithm with lower complexity, which can not only reduce the computing cost of the GPU but also facilitate the expansion and extension of the technology in the future. Therefore, I finally chose the simpler realization process [11] and [12] to adopt the masking method as the basic realization method of the project goal. [11] and [12] both use the GLSL language in OpenGL to write the fragment shader used to control the depth effect. Although GLSL as the underlying shading language has a strong advantage in implementing some basic special effects, when making some advanced special effects, due to the lack of some packaged functional APIs, the process will become complicated and cumbersome. So in this project, I will reconstruct the cut-away effect of the mask in Unity, and add some adjustable depth cues to make its perspective effect more suitable for the external environment.

Chapter 3

Methodology

The main goal of this project is to use the Unity engine to achieve a correct cut-away effect, that is, to achieve the effect of placing the virtual model inside the real object. The ideal realization result of the project is to allow the user to easily determine that the physical visual distance of the virtual target object is farther than the visual distance of the real obstruction on the front plane. After comprehensively researching the implementation methods of the above several papers, I divided the implementation process of this project into three stages:

1. Copy a virtual occluder model;
2. Create a cut-away effect on the model;
3. Blending with the camera image.

3.1 Copy a virtual occluder model

When summarizing the final results of related papers such as Mendez.[11] and Otsuki.[12] that use masks as the main means to enhance depth perception, we can find that their mask representation is related to physical information such as the shape and size of the real occluder. If the object of the cut-away effect is not the physical feature of a large smooth plane like a wall, it has its unique shape features, such as toy cars, mannequins, etc. And if we want to make the virtual window in cut-away match the relative physical position of the occluder. For example, to create a virtual circular window on the front of the computer main box to observe the internal circuit board structure, we need to measure the main box length, width, and height and set the three-dimensional coordinate value of the window in the scene according to the measurement data to make it fit the plane of the main box as much as possible.

Therefore, a key step in the preparation phase of the project is to copy a digital occluder in Unity. This occluder should be as similar as possible to the implementation object in the real world. Including the ratio of length, width, and height of the object, the shape, and the unevenness of the surface. At the same time, the size ratio between the virtual window and the occluder must be considered and then mapped to the prepared virtual occluder model. About how to realize the modeling of real 3D objects, we can use 3D modeling software such as Maya, 3DMax, etc. to manually create digital models of the occluders. We can also use some open-source object scanning modeling software, such as Vuforia Object Scanner, which can automatically generate the corresponding 3D model only by collecting the 360-degree feature information of the object through the camera. But the research focus of this project is not how to create a complex occluder model, so I chose carton which is the most common thing in the real world with a simple geometric structure as the implementation object of the cut-away effect. In the same way, I also built a cube with the same ratio of length, width, and height in Unity as a virtual occluder object.

3.1.1 Vuforia

In the development of AR scenarios, I choose Vuforia as the development tool, which is the most widely used AR Software Development Kit. It supports the application of scenes to multiple platforms such as mobile phones, tablets, and head-mounted display devices, and can realize rich object and scene interaction functions. The most important thing in the realization of AR is the precise tracking and positioning technology. Vuforia mainly includes the following tracking functions according to the different tracking targets: image tracking, object tracking, and environmental tracking. The complexity of tracking targets gradually increases from the picture to the environment. The following Figure3.1 shows the target types that can be added to Vuforia Target Manager.

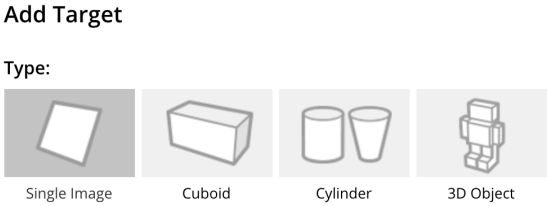


Figure 3.1: Vuforia Target Manager

The two crucial tracking technologies for this project are image tracking and object tracking. As mentioned in the previous section, if we want to create a complex virtual occlusion model, we need to use Vuforia's object tracking technology. It mainly includes model tracking and object tracking. The difference between them is that model tracking uses an existing 3D model and a paired physical object,

which are mostly used in industrial production and medical treatment. The target object using model tracking technology can be detected through its outer contour. Object tracking is mostly for small objects, such as toys, and the object needs to contain rich material feature information. It requires the use of Vuforia Object Scanner and supporting image targets to create 3D object targets, but this does not provide a mesh that can be used for authoring or occlusion. Through these two tracking technologies, we can implement creative and interactive functions based on the obtained 3D model targets. However, because the temporary obstruction target of this project is set as a simple cube, there is no need to use 3D object tracking technology, so its specific application process will not be discussed.

For convenience, this project mainly uses image tracking technology. Image tracking is to create one or more flat images in the scene and set them as tracking targets. The image targets need to contain rich original feature information (usually curves, intersections, etc.). When the AR application is running, the Vuforia engine will compare the features obtained from the image target with the image obtained by the current AR camera to detect whether there is matching feature information. Once the matching feature target is detected, the set enhanced content will be placed in the corresponding position. At the same time, it will track the location of the image target in real-time and update the enhanced content in real-time.

3.1.2 Box Tracking

In the preparatory work phase, I created a database in Vuforia Target Manager to store the picture targets that will be used. And prepared a black and white maze picture Figure3.2 as the picture target and then input it into the picture database, and at the same time printed a copy of the same size paper version of the picture target as the identification object. To associate the picture target with the real obstruction in real time, I attached the picture to one side of the box. So to ensure that the position of the box can be captured correctly in the camera picture, I set a cube with the same size and proportion in the same relative position of the picture in the Unity 3D scene and set the material as a wireframe to detect whether it exactly matches the real object.

Specify the shader in the wireframe shader to render transparently, and set two Pass channels to avoid the depth problem of transparent mixing. For the transparent rendering method, firstly draws the surface far away from the camera and then draws the closer surface to get the correct transparent result. Since the backside of the object must be farther than the front side, the backside must be drawn in the first rendering. The Blend is designated as $SrcAlpha\ OneMinusSrcAlpha$, which is the normal transparency blending effect mode. The main rendering algorithm of the wireframe is located in the fragment shader, as shown in the Figure3.3. By calculating lx, ly, hx, and hy to discover whether

Figure 3.2: Target Image

the rendered pixel is located in the square whose center is consistent with the UV center, with the $1-\_Width*2$ side length. If not, it is rendered as a line. The step function is used to judge whether the previous input value is less than or equal to the next input value. If the result is positive, it returns 1, otherwise, it returns 0 (for example, the judgment condition of lx is whether the width of the wireframe $\_Width$ is less than or equal to the x coordinate value of the current uv). The lerp function indicates that when the result of $lx*ly*hx*hy$ is 0, it returns the wireframe color $\_EdgeColor$, otherwise it returns the model color $\_Color$. The front of the model is rendered in the second pass, and the remaining parts are the same as in the first pass.

```
fixed4 frag(v2f i) : COLOR
{
    fixed4 col;
    float lx = step(_Width, i.uv.x);
    float ly = step(_Width, i.uv.y);
    float hx = step(i.uv.x, 1.0 - _Width);
    float hy = step(i.uv.y, 1.0 - _Width);
    col = lerp(_EdgeColor, _Color, lx*ly*hx*hy);
    return col;
}
```

Figure 3.3: The fragment buffer for wireframe shader

The final realization result is shown in the Figure3.4. It can be seen from the renderings that the wireframe matches the real box more accurately, regardless of size or position, so the next step will be to make the next cut-away effect on this basis.

3.2 Create a cut-away effect

The creation principle of the cut-away effect is to make the internal scene of the object visible only in a certain fixed area, and invisible outside the area. At the same time, the original occluder material

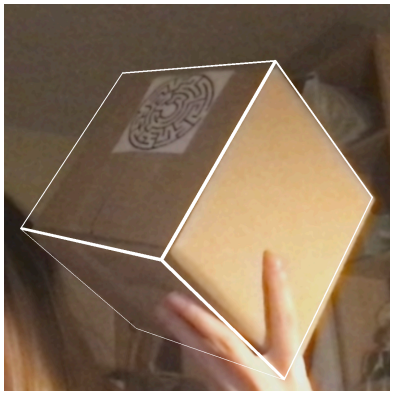


Figure 3.4: Wireframe match result in AR

outside this area should be kept or replaced with the background image obtained by the AR camera. The difficulty in this process is how to create this specified area, that is, a virtual window. In general game scenes, we can indeed see building models that include windows, and we can also observe the furniture placement in the house through the windows. However, the formation of these windows is based on the construction of 3D models. In the initial modeling process, corresponding holes have been dug in the wall model, and the position of the windows has been reserved. But we can't build a model for each target occluder and physically dig a hole in the model, which deviates from our original goal. What we need is a virtual window that can be formed later.

At the very beginning, I planned to use the $depthMask$ shader to try it out. The function of $depthMask$ is that when it is used for a certain mask object, it can hinder the drawing of all objects obscured by this mask object. It is mostly used in AR to make portal special effects or fake windows. As shown in the Figure3.5, I am trying to use $depthMask$ to create object occlusion effects. Its principal focuses mainly on the setting of render queues and the use of depth buffer. We set the pass condition to $LEqual$ in the $Ztest$ depth test, that is, when the depth value of another object (the value of render queues) is less than or equal to the depth value of the current object, it will pass and be drawn.

This is to mention Unity's built-in rendering queue, which includes $Background$, $Geometry$, $AlphaTest$, $Transparent$, and $Overlay$. The value of the $Background$ tag is 1000, this queue is usually rendered first, such as the skybox; the value of $Geometry$ is 2000, which is applied to most opaque objects by default; the value of $AlphaTest$ is 2450, which is mostly used for objects that need to enable alpha test; The value of $Transparent$ is 3000. After the rendering of the above queue is completed, the rendering pipeline renders all the objects (mostly using transparency blending) included in the $Transparent$ queue in the order back to front; the value of $Overlay$ is 4000, and this queue is used add some additional screen effects.

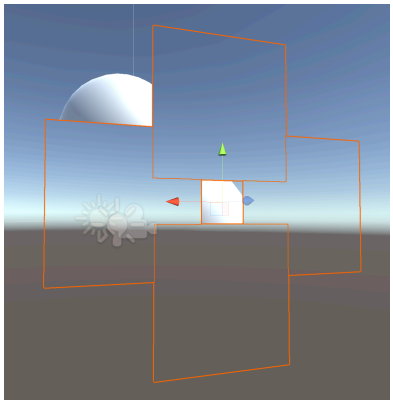


Figure 3.5: Effect for $depthMask$ shader

So if you want an object to be occluded by the $depthMask$, you need to set its render queues value to be greater than the value of the mask shader, which will change the rendering order of the two. Similarly, if you don't want to be occluded, you can change the rendering order to be earlier than the mask shader. Another important point based on implementing occlusion objects is to ensure that the pixels in the corresponding area are not rendered. This is mainly achieved by setting $ColorMask$ to 0, that is, setting it to not draw anything to the $RGBA$ channel except for the depth buffer.

But the disadvantage of the $depthMask$ shader is that it can only form a rectangular window through four depth planes, and cannot be set to other shapes. For example, if you want to replace the rectangular window with a circle, it is impossible to directly achieved by accumulating the geometric model. However, it is difficult to carve out a region that does not block the drawing of the occluded object on the depth plane that itself blocks the drawing of the occluded object The development process seems to have reached a deadlock. The method of using $depthMask$ to make cut-away effects is not working, so I changed my mind and tried to find mask effects in other application scenarios.

3.2.1 Stencil Buffer

In the UI settings of some game scenes, I have observed that the implementation of some special-shaped UI icons is to use Stencil Buffer to make a mask effect on the 2D image, which can also produce the effect of only observing the image in a specific area. If we understand how it works, we might also be able to apply it to a 3D scene as an object mask. So what is stencil buffer?

The original meaning of Stencil refers to a board engraved with a hollow pattern. In daily life, it is usually covered in the place to be sprayed. After the spraying is completed, certain specific shapes can be easily and quickly obtained. In the same way, the stencil buffer is another type of data buffer in addition to color buffer, pixel buffer, and depth buffer commonly used in computer graphics hardware such as OpenGL 3D graphics. The stencil buffer is a buffer of integer values in pixels, and a byte-

length value is usually assigned to each pixel. The depth buffer and the stencil buffer often share the same area in the random access memory (RAM) of the graphics hardware [17]. It is in the pipeline process as shown in the Figure3.6 below. After the Alpha test and before the Depth test, it is often used together with the depth test to produce various special effects. It is most often used to limit the rendering area. For example, it is used in portal rendering to restrict pixels outside the portal area from being rendered. After the user moves the perspective, the pixels are re-detected according to the updated line of sight range and then rendering correctly. Another application scenario is to render three-dimensional object shadows together with depth testing.

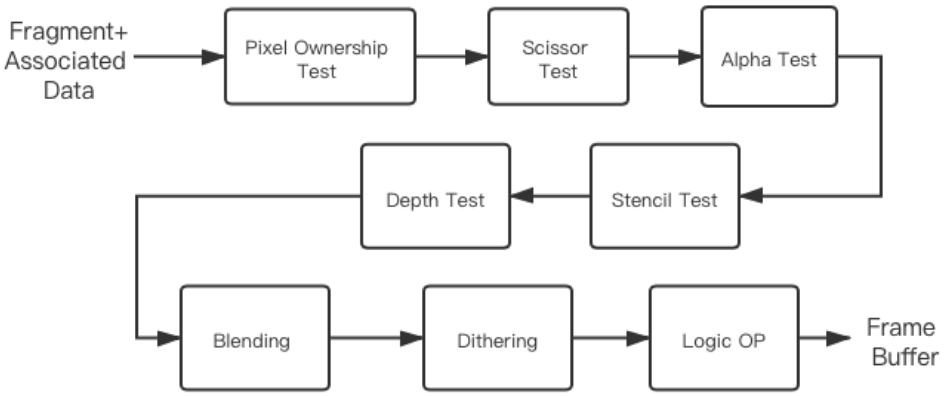


Figure 3.6: Per-Fragment Pipeline process[1]

It works by specifying any shape in the buffer as a stencil during the entire rendering process, then it performs stencil testing on every screen pixel and determines whether the pixel is rendered by the comparison result between the current $StencilBufferValue$ (that is, the buffer gives The value of the byte length allocated for each pixel) with the preset $ReferenceValue$. If the stencil test passes, the corresponding pixel is updated, otherwise, it is not updated. In this way, we can control the rendering of pixels in a certain area by setting different conditions for passing the test. As shown in the Figure3.7 below, the blue rectangle on the left is superimposed on the black and white flower mask in the middle, and finally presents a blue flower pattern without a background. The white part of the binary mask is considered as the area through which pixels can pass, while the black part of the pixel is rejected and rendered.

The mask method in the above figure is mainly for 2D images. For three-dimensional objects, the white area in the middle can be replaced by a geometric object, and the surrounding black part can be understood as all areas of the screen outside the geometric object. The blue rectangle can be replaced with any target internal model.

Figure 3.7: The way stencil buffer works

3.2.2 Built-in Render Pipeline & Universal Render Pipeline

After determining the basic implementation ideas, for different game application scenarios and game effects, we need to choose the most suitable Render Pipeline to implement the corresponding operations. At present, Unity mainly provides three kinds of render pipelines, namely Built-in Render Pipeline, Universal Render Pipeline (URP), and High Definition Render Pipeline (HDRP). Different render pipelines have different performance and performance styles. Built-in Render Pipeline is the default rendering pipeline in Unity, which only supports low-level custom operations. URP and HDRP belong to the category of Scriptable rendering pipelines, and both support a high degree of freedom and intuitive custom operations.

The rendering pipeline mainly uses the following three steps to determine how to render and display objects in the scene, as shown in Figure3.8 [15]. Determine which objects need to be rendered through culling. After deciding the rendering list objects, use the correct lighting to draw the objects into the pixel-based buffer according to the properties of each object. The final post-processing step is used to output the results of all the previous buffers. Realize some additional visual effects (such as depth of field, blur, etc.) to get the final output frame.

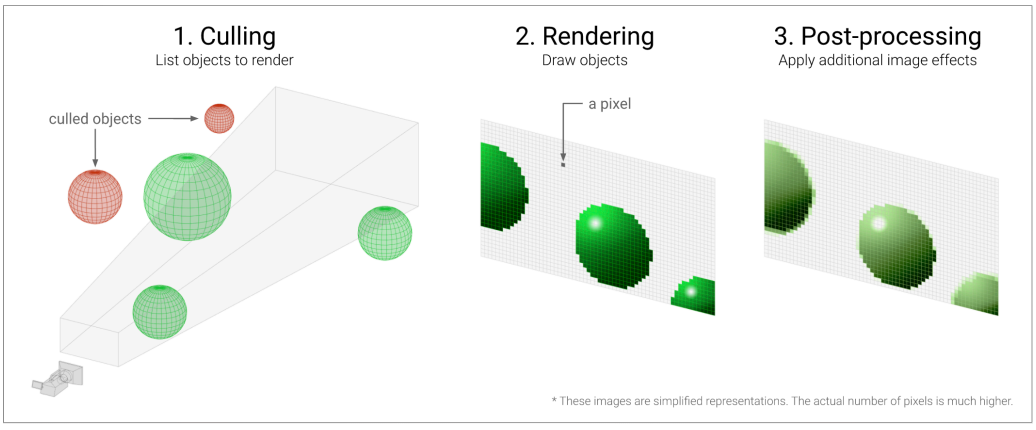


Figure 3.8: 3 steps for Render Pipeline from Unity documentation[15]

Built-in Render Pipeline provides two rendering paths, one is a multi-pass forward rendering path, all objects will be rendered in sequence. The other is the deferred rendering path, which is suitable

for scenes with multiple dynamic lighting, that is, the basic geometric model of the solid object is first rendered into the buffer, and finally the attributes such as material and color are rendered according to the lighting conditions. Its advantage is that users can use many basic functional APIs to write custom shaders to achieve various material effects. Including Stencil Buffer and $depthMask$, many custom shaders can only be applied to the Built-in render pipeline. But its disadvantage is that it uses traditional C++ to program the shade, so the user cannot intuitively see the intermediate results of each step during the shader writing process, which is not conducive to the user's real-time adjustment during the writing process.

Compared with the Built-in render pipeline, the Universal Render Pipeline provides a more efficient single-pass forward renderer, which is more suitable for the Physically Based Rendering (PBR) rendering pipeline of mobile platforms. And it supports Unity Shader & VFX Graph for shader writing. Unity Shader & VFX Graph is a special Shader editor that compiles logic code through visual link nodes, and then automatically compiles to obtain the final effect, which is more user-friendly. So URP is very suitable for making some complex pictures or material special effects. But the disadvantage of URP is that it does not support many custom C++ shaders, so many custom shader effects in Built-in cannot be implemented in URP. Including cut-away effects with Stencil Buffer, because of the render pipeline's limitations on the functions that operate the stencil buffer.

3.3 Fake Window shader & Virtual Occluded Object shader

Because the most basic cut-away effect requires Stencil Buffer to achieve, I choose to render it on the Built-in Render Pipeline. To achieve the binary mask and 3D occluded objects in Figure3.8, I created two different shaders. They are the StencilWrite shader for false windows and the StencilRead shader for occluded objects.

StencilWrite shader

Set the Render queue to $'Geometry-1'$ in the $StencilWrite$ shader to ensure that fake windows will be rendered first, and all objects whose rendering sequence is $'Geometry'$ can be blocked and restricted. The Stencil Buffer section set in $SubShader$ is as shown in the Figure3.9 below, $'\_StencilRef'$ is a custom input variable with a value range of 0 to 255, which is used to allow users to customize the $Stencil\ Reference\ Value$. In this way, if there are two or more windows and multiple target objects in the scene, you can set different reference values to allow different windows to selectively render the model. $Comp$ represents the judgment condition of Stencil Buffer when performing numerical comparison operations. Common conditions are $Greater$, $GEqual$, $Less$, $LEqual$, $Equal$, $NotEqual$,

$Always$, and $Never$. Here we use the $Always$ condition to indicate that this region's pixels will always pass. $Pass$ represents the pixel update operation after the stencil test is passed, generally including $Keep$, $Zero$, $Replace$, $IncrSat$, $DecrSat$, $Invert$, $IncrWra$p, and $DecrWrap$. Here is set to Replace in the $StencilWrite$ shader, so the stencil value of all pixels in this area that meet the conditions will be replaced with the $Stencil\ Reference\ Value$.

```
//stencil operation
Stencil{
    Ref [_StencilRef]
    Comp Always
    Pass Replace
}
```

Figure 3.9: Stencil Buffer settings in $StencilWrite$ shader

StencilWrite shader

The $StencilRead$ shader sets the Render queue to $'Geometry'$ so that the obscured object will be rendered after the fake window and then judge the value of the stencil. It also sets the $'\_StencilRef'$ variable, which is different from $StencilWrite$ in that it represents the Stencil value of the object itself. At the same time, because only if the object's pixels that match the comparison criteria (the stencil buffer value is equal to the reference value) can be passed and rendered in this area, the judgment condition of the Stencil test is set to Equal, as shown in the Figure3.10.

```
//stencil operation
Stencil{
    Ref [_StencilRef]
    Comp Equal
}
```

Figure 3.10: Stencil Buffer settings in $StencilRead$ shader

In the end, the effect of the joint of $StencilWrite$ and $StencilRead$ is shown in the figure. I created a circular geometry to represent the fake window in the cut-away effect and assigned the material with the $StencilWrite$ shader to it. I used material with a $StencilRead$ shader on the internal scene model and finally realized that only in this circular area can the internal car model and walls be seen. And as the viewing angle moves, the user will observe the car structure from different angles.

3.4 Improvement of cut-away

Although the most basic cut-away effect has been achieved, there is still room for improvement in the degree of depth perception. So I mainly divided it into two methods to improve the performance. The first is to add a blur effect on this basis. The principle is the same as Fischer.[6], which is to blur the boundary between the real world and virtual objects and turn them into a unified style. The second way is to switch to Universal Render Pipeline (URP). Through its Shader Graph editor, additional adjustable special effects can be made more easily, which allows users to flexibly adjust cut-away effects that better match the external environment.

3.4.1 Adjustable features in URP

Since the stencil buffer cannot be used in URP, the cut-away results previously achieved need to be re-implemented in URP. When observing some game content, I found a special effect essentially similar to cut-away, which is the see-through effect. The see-through in the game is generally set to the main camera of the scene to move with the character of the game. When moving to a place where the character is blocked by walls or buildings, a circular space will appear around the character. In this space the walls and other obstructions within the range are invisible. On the other hand, this is a function of only forming a appropriate perceived spatial relationship in a specific range to observe the objects behind the obstruction.

The see-through effect is mainly achieved through Shader Graph, I added a custom mask to the shader graph, passed the calculated mask in the shader's alpha channel, and used the alpha value of the mask to control the transparency of the surface material to achieve the cut-off effect. To allow users to adjust the size of the cut-away area through the UI, when there is only one cut-away area on a plane, I calculate the circular transparent area through the Formula3.1 and 3.2. The calculation process in the shader is as shown in the Figure3.11 below.

$$d = length((UV * 2 - 1)/float2(Width, Height)); \tag{3.1}$$

$$Out = saturate((1 - d)/fwidth(d)); \tag{3.2}$$

This Formula3.1 and 3.2 is the calculation formula of the Ellipse component that comes with Unity, where Width and Height are the length and width of the ellipse, respectively, and UV is the input material UV value. After the intermediate value d is calculated, it is brought into the $saturate$ function to normalize the result to the range of 0 to 1, and the final output is the binary black and white result image shown in the Figure3.11. In this project, I set both $Width$ and $Height$ to the same parameter $Size$, so that the output result can be guaranteed to be a perfect circle.

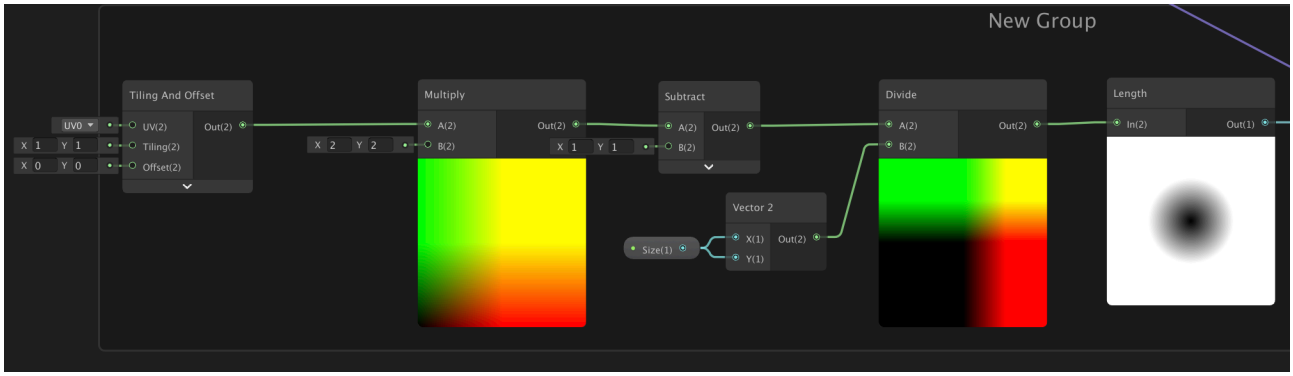


Figure 3.11: Unity Shader Graph implementation for calculating an elliptical mask used for the cut-away hole

In addition to the basic adjustable size circular binary mask, I also implemented the following functions:

Adjustable number of fake holes

To increase the number of holes, I added a custom integer variable $NumberOfHole$, and $Comparison$ and $Brunch$ components, as shown in the Figure3.12. Judging by the Comparison function, when the value of variable $'NumberOfHole'$ is greater than 1, a circular texture noise map is used, and the scaling of the map is controlled by $Tilling\ and\ Offset$ functions to achieve the effect of uniform distribution of holes on the surface. The switching condition of Brunch is whether the output result of Comparison is true or false. If it is false, the current number of holes is less than 2, so the result of the previous step is output. Otherwise, the output result of the Noise map in this step is used.

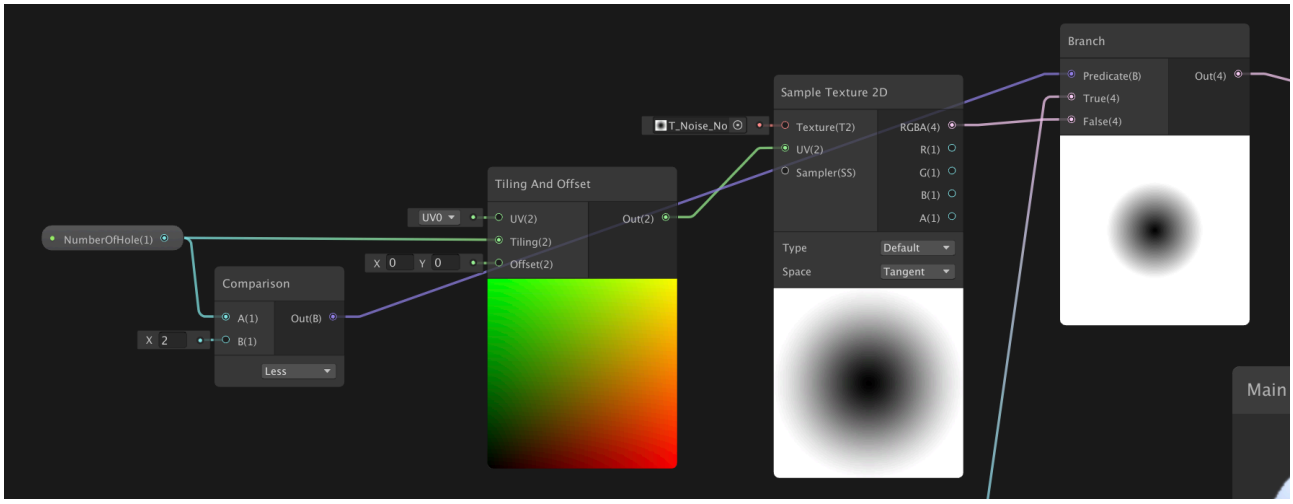


Figure 3.12: Unity Shader Graph implementation for adjusting the number of cut-away holes

Smooth transition strength at the edge of the cut-away

After performing the $OneMinus$ operation on the result of the previous step, enter it into the $Smoothstep$ function, set the value of one side to 0, and the value of the other side to the custom floating-point variable $Smoothness$ with a value range of 0 to 1. , As shown in the Figure3.13. Smoothstep can be used to generate a smooth transition value from 0 to 1, so it can make a smooth transition from the boundary between 0 (outermost circle) of the circular mask to the smoothness value (how far away from the outermost circle).

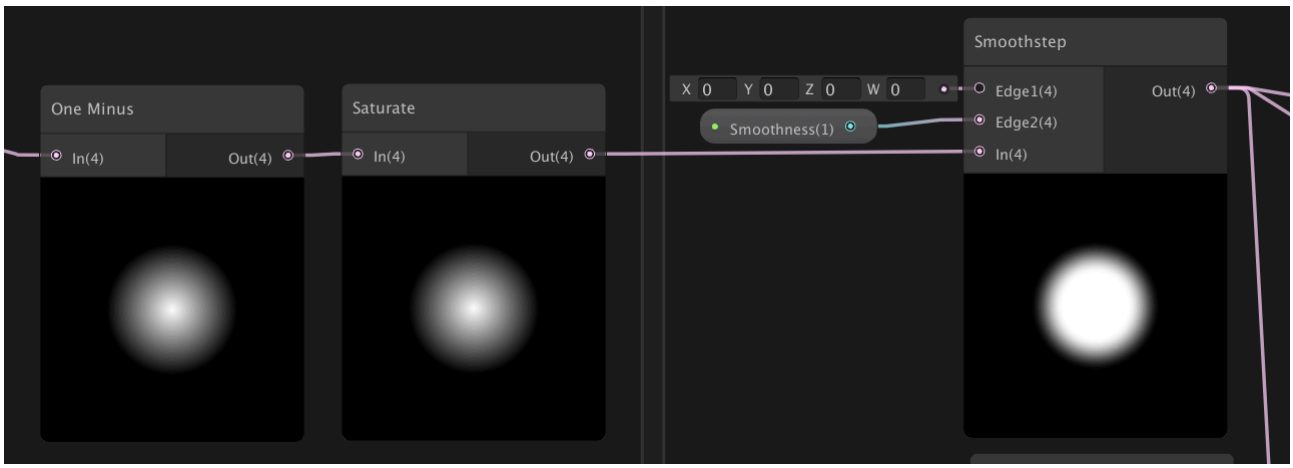


Figure 3.13: Unity Shader Graph implementation for smoothing the transition strength at the edge of the cut-away hole

Add noise (rough texture)

Because the actual external occluder set is a carton, its surface material is non-smooth material. So if you want to increase the sense of reality, you can add some noise to the circular mask to simulate the rough graininess of the carton surface. The process is shown in the Figure4.2c. After finishing the transition operation of the previous step, multiply the result with the Simple Noise that comes with Unity to get the mask as shown in the second result image of the second row. Whether to add noise effect is determined by the custom Boolean variable $Noise$. In the same way, if we only want the edges of the transparent area to be noised, we only need to multiply the smooth result of the previous step on this basis.

Adjust transparency of fake hole

The operation of adjusting the transparency is placed in the last step as shown in the Figure3.15, because it is equivalent to the last post-processing step in the execution step of the rendering pipeline,

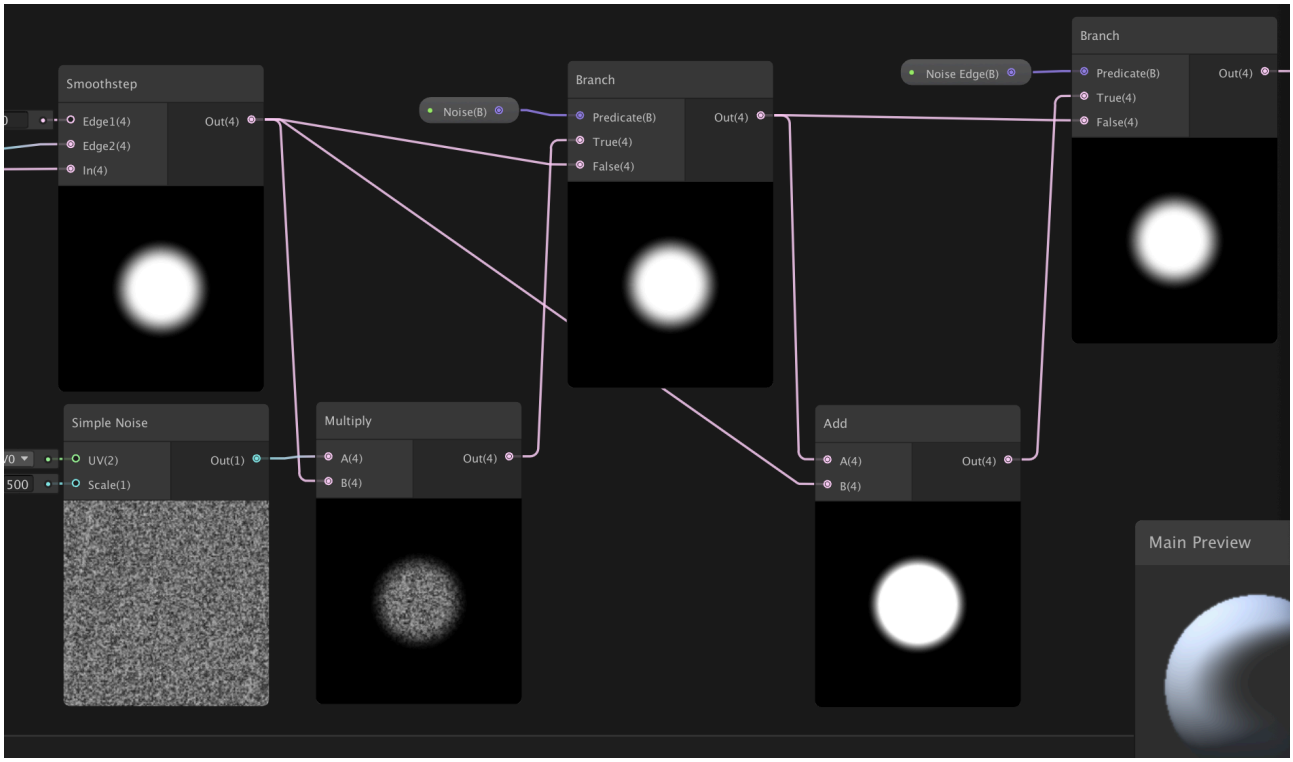


Figure 3.14: Unity Shader Graph implementation for adding noise on transparent cut-away hole area

that is, adding an additional effect of changing the transparency on the mask. After all the above steps are completed, just multiply the result with the custom floating-point variable $Opacity$. To prevent the result from spilling out of a valid value range, the result mask is limited to values between 0 and 1 by the $Clamp$ function before being entered into the Shader's alpha channel. Then it is reversed by $One\ Minus$, and finally rendered and output as the value of the alpha channel.

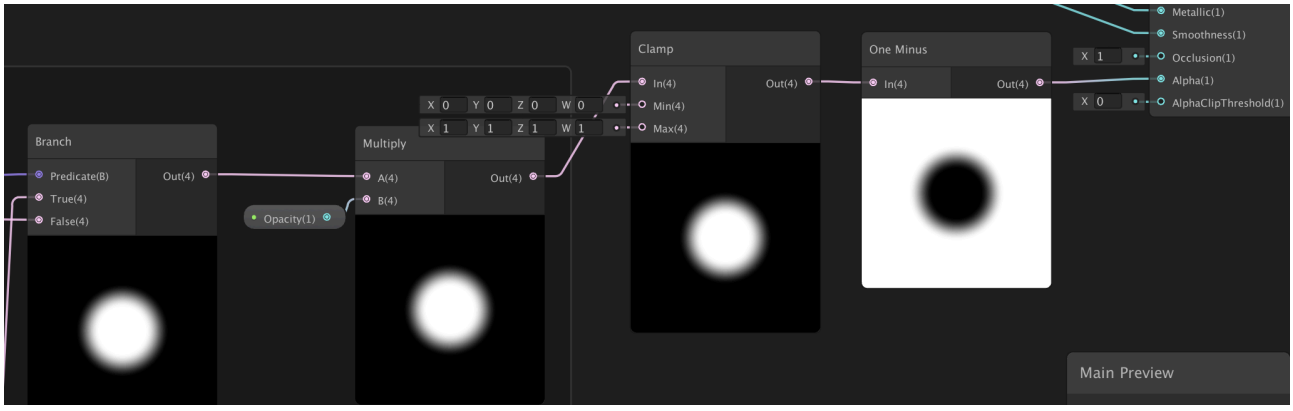


Figure 3.15: Unity Shader Graph implementation for adjusting transparency of cut-away hole

3.5 Blending with the camera image

The process of blending in the last step will vary depending on the use of the Render pipeline. For Built-in, the function of limiting pixel rendering in stencil buffer itself does not need to be attached to redundant virtual materials. Therefore, when it is directly rendered onto the occluded object through the AR camera, the background around the virtual window is always the real occluder material, and no further adjustment is required. However, the cut-away effect in the URP method depends on the existence of the virtual material. If the material other than the virtual window area is transparent, it will be integrated with the perspective area calculated in the middle, and no effect can be produced. Therefore, only opaque materials can be used around the virtual window. When projecting it directly onto a real object, the virtual material will cover the real material. At this time, the aforementioned Formula2.1 is needed to perform blending rendering of the scene and the camera background image. The basis of blending is the value of the alpha channel in the shader.

Chapter 4

Results & Discussion

4.1 Data Source

The car model resources used in this project mainly come from Unity Asset Store. The design of the AR scene uses the Unity 2019.4.28f1 version and the Vuforia 10.1 version of the SDK extension package. And the results of both methods mentioned in the previous chapter will be rendered to the actual target occlusion via WebCam.

4.2 Evaluation

Built-in Render Pipeline Method

The rendering result of the Built-in Render Pipeline is shown in the Figure4.1 below. The picture Figure4.4a on the left is a common cut-away effect. This method successfully achieves the result of virtual objects appearing inside the occluder, and the different angles of the internal car objects can be displayed as the view angle changes. This effect looks like a round hole is opened on the side of the carton, through which the user sees the car model inside the box. Its advantage is that it can retain the original feature information of the external carton, such as the material of the carton and the picture features attached to the top, based on adding virtual internal structure information.

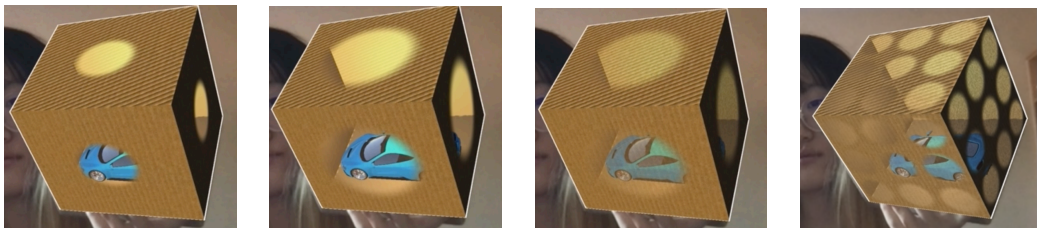
The picture Figure4.1b on the right shows the implementation result with the blur effect added. It achieves a unified style of virtual objects and the external environment by changing the clarity of the box and the cut-away part inside the box. The blurry image effect smooths the abruptness of the virtual window boundary in the cut-away effect, making it looks like a window that exists on the carton as a whole. Compared with the left picture, it increases the depth perception and becomes more reality.

(a) Simple cut-away (b) Cut-away with blur

Figure 4.1: Cut-away effect in Built-in Render Pipeline

Universal Render Pipeline Method

The rendering result of the Universal Render Pipeline is shown below. The Figure4.2a and the Figure4.2b respectively show that when each plane of the box has only one virtual window, the user can change the size of the hole in real-time through the UI. The value of the $Size$ variable at this time is 0.4 and 0.8 respectively. By changing the size of the perspective area, the user can obtain the characteristic information of a different number of virtual obscured objects. The larger the hole, the more virtual information can be obtained. Compared with the Figure4.2b,and Figure4.2c adds a noise map, which increases the roughness of the perspective area, making it look like the cartons surface has been thinned by a layer, which increases the realism. The Figure4.2d on the bottom right is the result of increasing the number of holes on each face. The number of holes on each face is 3x3, which is also a more reasonable way to increase the amount of information of the virtual feature.

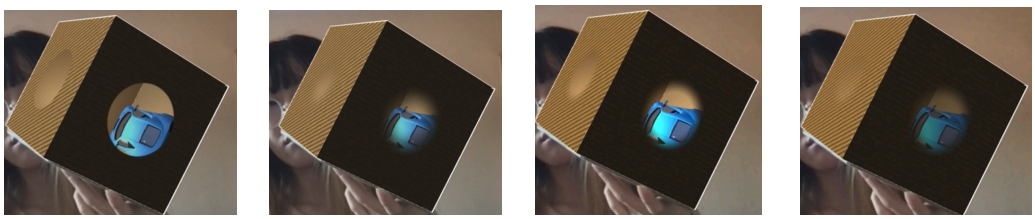


(a) Small size for hole (b) Big size for hole (c) Add Noise Map (d) Increase number

Figure 4.2: Cut-away effect in Universal Render Pipeline (1)

The Figure4.3a and Figure4.3b of this group of pictures represent the value of the $Smoothness$ variable of 0 and 0.6 respectively. It can be seen that as the smoothness increases, the transition between the virtual window and the surrounding ordinary materials gradually becomes softer. When the $Smoothness$ is 0, the virtual window looks like a flat image independent of the occluder; but after increasing the $Smoothness$ value, the depth of the car model looks greater than the depth of the box surface, increasing the spatial sense of space. The Figure4.3c and the Figure4.3d shows the situation

when the $Opacity$ variable values are 1 and 0.2, respectively. As the $Opacity$ value decreases, the internal car model gradually becomes invisible, but it is more realistic than the fully visible $Opacity$ of 1. Because the translucent perspective area will look thicker, the principle of Figure4.3c is similar to the previous set of pictures Figure4.2c.



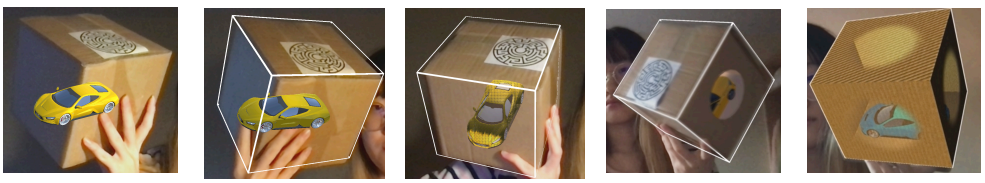
(a) Sharp edge (b) Gentle edge (c) Higher Opacity (d) Lower Opacity

Figure 4.3: Cut-away effect in Universal Render Pipeline (2)

The advantage of the URP method is that it has very rich adjustable features. Users can customize the desired cut-away effect through the UI panel, which has a very high degree of freedom. But its current shortcoming is that because the blending function of this method is still under development, the entire solid virtual box is projected on the real occluder instead of only the middle perspective window. This caused the false occlusion of the real features and completely lost the information of the real occluder, so it seems to have a certain deviation from the real cut-away.

4.3 Result Comparison

To make a more comprehensive judgment on the rendering results of the two techniques, I added three other different depth projection methods with the first two ways Virtual Hole (Built-in Render Pipeline Method) Figure4.4d and Alpha Mask (Universal Render Pipeline) Method) Figure4.4e for comparison. The additional approaches added are Simple MR Figure4.4a, Wireframe with MR object Figure4.4b, and Translucent MR object with wireframe Figure4.4c.



(a) Simple MR (b) Wireframe MR (c) Translucent MR (d) Virtual Hole (e) Alpha Mask

Figure 4.4: A comparison of the five methods: (a)Simple MR, (b)Wireframe with MR object, (c)Translucent MR object with a wireframe, (d)Virtual Hole and (e)Alpha Mask

Simple MR simply projects the MR object onto the occluder; Wireframe with MR object adds wire-

frame contour information representing the occluder; Translucent MR object with a wireframe has lowered the transparency of internal MR objects (car models) based on the Wireframe with MR object approach. The comprehensive evaluation results of the five methods are as follows (only representative of personal subjective evaluation).

Key Point / Methods	Simple MR	Wireframe with MR object	Translucent MR object with a wireframe	Virtual Hole	Alpha Mask
Depth perception effect	1	2	3	5	4
Amount of information	5	4	4	3	3
Hole Size	/	/	/	/	$\surd$
Number of Hole	/	/	/	/	$\surd$
Opacity	/	/	/	/	$\surd$
Edge transition	/	/	/	/	$\surd$
Rough texture	/	/	/	/	$\surd$
Blur	/	/	/	$\surd$	$\surd$

Table 4.1: Method Comparison Result

The numbers 1-5 in the table respectively represent the performance strength of the corresponding method in this index. The larger the number, the better the performance. From the table, it can be seen that in the horizontal comparison of the Depth perception effect, Virtual Hole has the best effect, with a level of 5; while Simple MR has the worst effect, with a level of 1. The depth perception effects of the other three methods are enhanced in turn from Wireframe with MR object to ALpha Mask. However, the opposite of the spatial perception affect ranking is the number of information users can obtain. In this comparison, Virtual Hole gets a lower 3 points, while Simple MR gets 5 points.

The remaining Dynamic adjustment part in the table is a functional comparison of each rendering method. If the method can adjust the corresponding features, fill in $\surd$, and fill in / if it cant. It can be found that Alpha Mask can achieve all the functions, but the best-performing Virtual Hole can only achieve the Blur effect.

4.4 Discussion

After comparing the five methods horizontally, we concluded that the best overall performance method is Virtual Hole (Built-in Render Pipeline). Because the importance of the $Depth\ perception\ effect$ indicator is greater than the $Amount\ of\ information$ users can obtain, it takes a larger proportion in the final result analysis. The reason why the sorting result of the $Amount\ of\ information$ is opposite to the sorting of the $Depth\ perception\ effect$ is that the realization of the cut-away effect will limit the user's observable field of view. If the field of view is limited, then it is certain that all the information of the internal MR objects cannot be seen at the same time, and the $Amount\ of\ information$ will be reduced. As for the remaining functional feature adjustments, they can only help to enhance spatial perception. Therefore, although Virtual Hole can achieve fewer functions than Alpha Mask, its performance is temporarily better than Alpha Mask. But the most critical step of Alpha Mask is to realize the blending operation with the camera screen, so after it is thoroughly developed, there will be a substantial performance improvement.

Chapter 5

Conclusion

5.1 Overall

After comparing and analyzing different types of enhanced depth perception technologies, this dissertation proposes two depth perception methods based on different rendering pipelines in Unity to achieve the cut-away effect of real occlusions. Combining actual application effects, the Virtual Hole method based on the Built-in Render Pipeline of the two methods can provide a more realistic depth relationship between objects. The difference between the best-performing Virtual Hole rendering model and some previous similar research papers Otsuki.[12] Mendez.[11] is that Virtual Hole uses a higher-level Unity engine for overall construction and uses a new shader technology. Technically speaking, it is easier to extend the function later than other depth perception technologies that use OpenGL. And it combines the advanced AR extension package, Vuforia SDK, which makes its application in AR scenes more possible. It will not be limited to simple geometric obstructions but can be applied to more complex scenes, and you can also add richer interactive functions.

5.2 Future Work

Based on the summary and analysis of the shortcomings of the existing results, the future development plan of this project is as follows:

- Target tracking technology for AR, to improve the stability of the augment effect (that is, to avoid the jitter of the virtual image, and to recognize the target without being restricted by the viewing angle), and to expand the optional range of obstructions. I plan to use Vuforia's Model Target function in the future to generate 3D models of a variety of real occlusions and use them as AR tracking targets. In this way, the problem of instability of the AR projection scene caused by inappropriate image angles in image tracking can be avoided, and more precise cut-away area

adjustment applications can be carried out for each kind of obstruction;

- For the Virtual Hole solution, to increase the diversity of its cut-away effects, the project will improve the perspective area material (such as glass) and customize the area size;
- The focus of the future development of the Alpha Mask solution is to realize the blending operation with the camera image and replace the virtual material of the non-perspective area with the corresponding weighted camera image according to the value of the alpha mask.

Bibliography

[1] Unityshader09:stencil buffer&stencil test. https://blog.csdn.net/u011047171/article/details/46928463, 2015. [Online; accessed 1-September-2021].

[2] Markus Broecker, Ross T Smith, and Bruce H Thomas. Depth perception in view-dependent near-field spatial AR. In *Proceedings of the Fifteenth Australasian User Interface Conference-Volume 150*, pages 87–88, 2014.

[3] Andrew I Comport, Eric Marchand, Muriel Pressigout, and Francois Chaumette. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions on visualization and computer graphics*, 12(4):615–628, 2006.

[4] Niklas Elmqvist, Ulf Assarsson, and Philippas Tsigas. Employing dynamic transparency for 3D occlusion management: Design issues and evaluation. In *IFIP Conference on Human-Computer Interaction*, pages 532–545. Springer, 2007.

[5] ErickMendezARWork. Some more importance masks for x-ray visualization in AR. https://youtu.be/YWmQ--Q_BbU, 2010.

[6] Jan Fischer, Douglas Cunningham, Dirk Bartz, Christian Wallraven, Heinrich H Bülthoff, Wolfgang Straßer, and R Hubbold. Measuring the discernability of virtual objects in conventional and stylized augmented reality. In *12. Eurographics Symposium on Virtual Environments (EGVE 2006)*, pages 53–61. Eurographics Association, 2006.

[7] Taiki Fukiage, Takeshi Oishi, and Katsushi Ikeuchi. Visibility-based blending for real-time applications. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 63–72. IEEE, 2014.

[8] Lasse Hedegaard Hansen, S Swanström Wyke, and Erik Kjems. Combining reality capture and augmented reality to visualise subsurface utilities in the field. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 37, pages 703–710. IAARC Publications, 2020.

[9] Denis Kalkofen, Erick Mendez, and Dieter Schmalstieg. Interactive focus and context visualization for augmented reality. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 191–201. IEEE, 2007.

[10] Daquan Liu, Chengjiang Long, Hongpan Zhang, Hanning Yu, Xinzhi Dong, and Chunxia Xiao. Arshadowgan: Shadow generative adversarial network for augmented reality in single light scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8139–8148, 2020.

[11] Erick Mendez and Dieter Schmalstieg. Importance masks for revealing occluded objects in augmented reality. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, pages 247–248, 2009.

[12] Mai Otsuki, Hideaki Kuzuoka, and Paul Milgram. Analysis of depth perception with virtual mask in stereoscopic AR. In *ICAT-EGVE*, pages 45–52, 2015.

[13] Stefan Reifinger, Frank Wallhoff, Markus Ablassmeier, Tony Poitschke, and Gerhard Rigoll. Static and dynamic hand-gesture recognition for augmented reality applications. In *International Conference on Human-Computer Interaction*, pages 728–737. Springer, 2007.

[14] Menandro Roxas, Tomoki Hori, Taiki Fukiage, Yasuhide Okamoto, and Takeshi Oishi. Occlusion handling using semantic segmentation and visibility-based rendering for mixed reality. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, pages 1–8, 2018.

[15] Unity Technologies. Choosing and configuring a render pipeline and lighting solution. `https://docs.unity3d.com/Manual/BestPracticeLightingPipelines.html`, 2021. [Online; accessed 1-September-2021].

[16] Housheng Wei, Yanli Liu, Guanyu Xing, Yanci Zhang, and Wenjia Huang. Simulating shadow interactions for outdoor augmented reality with rgbd data. *IEEE Access*, 7:75292–75304, 2019.

[17] Wikipedia contributors. Stencil buffer — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Stencil_buffer&oldid=1018450087`, 2021. [Online; accessed 1-September-2021].

[18] Felix Wimmer, Christoph Bichlmeier, Sandro M Heining, and Nassir Navab. Creating a vision channel for observing deep-seated anatomy in medical augmented reality. In *Bildverarbeitung für die Medizin 2008*, pages 298–302. Springer, 2008.

[19] Feng Zhou, Henry Been-Lirn Duh, and Mark Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 193–202. IEEE, 2008.

[20] Stefanie Zollmann, Raphael Grasset, Gerhard Reitmayr, and Tobias Langlotz. Image-based x-ray visualization techniques for spatial understanding in outdoor augmented reality. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design*, pages 194–203, 2014.