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

School of Computer Science and Statistics

Analysing the Percentage-Closer Soft Shadows rendering technique.

Pradnya Ashok Shinde

Supervisor Professor: Dr. Michael Manzke

August 19, 2022

A dissertation submitted in partial fulfilment

of the requirements for the degree of

M.Sc (Computer Engineering)

Declaration

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

Signed: _____ Date: _____

Abstract

To make a computer scene realistic and visually engaging efficient shadows are needed. The process of generating shadows is preferably complex. When a object completely or partly obstructs the light, a dark area will be created inside the bright area which is often called a **shadow**. With the help of shadow the correlation between light source and object can be determined. In 1977 Frank Crow came up with the idea of shadows and in 1978 with the help of depth buffer Lance Williams proposed the idea of of implementing shadows which became popular in 2005 after the techniques were used in Doom 3. Shadows are an outcome of three different areas **umbra,penumbra, antumbra**. For a naturalistic shadow all these parts should be considered.Shadows can be categorised in two sections, hard shadows which have sharp edges and those edges are very well defined. One of the challenges which is faced while rendering the shadow is aliasing . When jagged edges can be noticed in resultant shadow it is because of aliasing. A new technique known as Percentage-Closer Soft Shadows was invented to render shadows. This technique is capable of creating significantly precise soft shadows with the use of single shadow map. The computation cost will be less as there is no necessity of pre-computation and post-computation or any extra calculations. It can also replace the traditional shadow map and reduces its complexity. The use of single sample and no extra processing makes this technique worth of trying. In PCSS shadow rendering technique, instead of using classic pixel shader for shadow mapping, PCSS shader is used. This research focuses on analysing the PCSS technique and summarising the advantages and disadvantages of PCSS.

Acknowledgements

My sincere thanks to Dr. Michael Manzke for supervising the entire process of dissertation and guiding me on the minute details of the research work and for the constant encouragement. I would also like to thank Dr. John Dingliana for his valuable guidance. With this special thanks to my friends and family for motivating me throughout the process.

Contents

List of Figures

Nomenclature

PCSS	Percentage-Closer Soft Shadows
PCF	Percentage-Closer Filter
FPS	Frames Per Second

1 Introduction

1.1 Overview

Collaborative computer graphics has evolved magnificently in last couple of years. In earlier days only two-dimensional depiction can be achieved through computer graphics but creating accurate shadows is one of the primary concerns faced in computer graphics. Shadows are advantageous in numerous ways among them two important uses are: they significantly enhance the quality which can lead to great realism and by providing visual cues, they assist in understanding relative position of object in 3D scene (1). Shadows can be described as a way of light getting blocked because of the object resulting in a darkness which called as shadow (2). Shadow rendering plays a vital role in graphics if one has to make a scene and image realistic. Shadow rendering is an area which deals with applying the shadows to 3D computer graphics objects. In 1978 Lance Williams came up with the idea of attaching shadows to the objects in his "Casting curved shadows on curved surfaces" (3). If shadows are rendered in the proper manner, it can enhance the quality of scene in the best possible way. In about game development, it plays a vital role as it showcases practicality of the characters in the scene. By generating shadows, we can get to know that whether the object is blocked by any light source. Rendering shadows were challenging after 3-dimension objects where introduced. Shadows are also responsible for imparting enormous information about second object view. If soft shadows are considered, its really hard to replicate them. The following image shows an approach named as Percentage Closer Soft Shadows which can precisely create soft shadows. This technique is developed with the reference from shadow mapping and percentage-closer filtering (4) ,which has the following features:

- Creates soft shadows which are precise.
- No extra processing is needed.
- Shadow map can be effectively replaced .

Since there is no need for any extra steps for execution and it is simple to achieve, this method is a good choice for replacing shadow mapping method.

1.2 Motivation

Even though computer graphics technologies are developing immensely these days, rendering the shadows for scenes is still a challenge. Shadows can be constructed in many ways, for example shadow mapping, shadow ray or PCSS. Hard-edged shadows can be developed by ray casting method but on the other hand for developing soften edge shadow, shadow mapping method is more efficient. Though shadows play a vital role in enhancing the scene, it is a complicated task to achieve the desired effect. Earlier because of the hardware limitations, hard shadows were widely used. To achieve more realism soft shadows were instigated. As there are various methods to generate shadows, every method has its own cons and pros. The raytracing method and radiosity render the most realistic images with shadows but they require very complex computations and the result cannot be generated in real time. As the execution is complex it can lead to damaging the scene, unnecessary delays to generate the output. So this paper (4) which developed an interesting method for soft shadow rendering with percentage-closer soft shadows to reduce less complexity and faster execution. With the addition of a lightweight buffer for tracking dynamic scene objects, we can robustly and efficiently detect all screen space fragment that need to be updated, including not only moving object, but also their soft shadows. With implementation this research aspires to summarise the uses of this technique and compare the end result and computation time. Though there are multiple shadow rendering techniques which exist, rendering a shadow is a difficult task as it requires high computational resources. One technique to minimize aliasing in shadow map is to update shadow map projection areas to minimize utmost minification and magnification. In the area of shadow map there is humongous research is going on. Shadow mapping , ray tracing are few of the widely used rendering techniques in computer graphics. PCSS was invented by Nvidia Corporation which can be a solution on aliasing but it is still not widely used. The main idea behind this research is to analyse the reasons why percentage closer filtering did not become a widely used technique as compared to others.

1.3 Research Objectives

After analysing the shadow rendering methods and the challenges which can be faced while achieving the realistic shadow for images, the PCSS features could be capable of rendering the shadows which are closer to real world. Adding shadows to the scene can offer a great realism if shadows are rendered accurately. To offer certain influence on clients lighting on the scene and the way shadows are rendered plays a significant role. The aim of this research is to categorise the various type of shadows and its features. As per the characteristics and scene requirement the shadow algorithm will be chosen. If the shadow algorithm is categorised there will be two main sections, first one is shadow volume and other one is shadow mapping. Depth

map plays an important role in achieving accurate shadows. The light source and its distance from the the shadow map is the quickest method to store the information of closet object from the light if certain direction is considered. Further this data is used to estimate the objects which are obstructed. Due to inconsistency between depth maps data and sampling data, image based shadow rendering algorithms suffer from aliasing problem (5). This research is an attempt to list the challenges and possible solutions for accurate shadows which are closer to real world. Shadows can be rendered using: shadow mapping, shadow volume, and PCSS. This research will focus on PCSS which is constructed on the way it filters the sample and with this it can handle the aliasing problem and can generate soft shadow boundaries. In the following chapters, a clear summary of PCSS, its use, advantages, and disadvantages are mentioned. This entire research is focusing on following points:

- Basic working principles of PCSS.
- The importance of z-buffer in PCSS while storing the information which will be used to render the shadows for objects.
- Understanding the problems with PCSS specially the aliasing issue.
- Listing the possible solutions for the issues faced by PCSS.

1.4 Existing Research

As shadows play a vital role in enhancing the quality of the scene there are various ways to generate shadows. Computer graphics has immensely grown in these years, but still generating realistic shadows are challenging. Shadows can be hard or soft, depending on the circumstances, leading to its own challenges. In past years, various methods have been developed to render shadows including shadow mapping,shadow volume , ray tracing, planar shadows, PCSS, to name a few. A shadow consists of three parts namely, umbra,penumbra and antumbra. To generate realistic shadows of images all these three parts should be rendered. Most frequently used technique is shadow mapping as it is considered to be one of the most efficient methods to render shadows. Algorithms which are used to render shadows are categorised into two major sections - shadow mapping (3) and shadow volume (6). In shadow mapping, to calculate shadow maps per light just a single rendering pass is needed and this method is image based. However, this method has aliasing issues due to inconsistency between projected sampling of pixels in images to the shadow map and between sampling operation in shadow map. The aliasing issues can be handled by z-partition method and developing soft shadows by adaptive filtering. Using single shadow map is not feasible when we have virtual environments

which are huge. The reason behind this is restriction on graphics hardware when high texture resolution and not enough parameters can be generated for global re-parameterizations of a single shadow map.(7) The solution to this is dividing the view frustum to various depth layers after which for every split section it will develop an independent shadow map. To filter the result of shadow and get realistic shadows, PCSS will get applied for filtration purpose. There is huge research available on casting shadows hence this section will focus on techniques which are more relevant.

1.5 Thesis Structure

This section explains how the thesis is structured.The detailed background is given in Section one which is introduction. This section includes background research about the percentage-closer soft shadows which is followed by motivation for doing research on this topic and the overall overview of the topic, this will also put a light on the inspiration behind the research. The next part in introduction section is the study which is done till date for shadow rendering, its techniques, the challenges included in rendering shadows. The section two is about detail literature survey on shadow rendering. This includes various discussions by various authors on the impact of shadows in the scenes, How to improve quality of shadow in the scene, how to cast the shadows if the surface is curved, With this also the importance of shadow map, the process of rendering accurate shadows. The literature review section includes the research which is going on and already present about the topic. The methodology section includes the methods which are there to generate the shadows. This section also includes the advantages and disadvantages of those methods and their process of generating shadows.The next section covers the implementation part and what all steps were taken to achieve the desired outcomes. The last section will summarize the the information about PCSS and its efficiency will be discussed.

2 Literature review

2.1 Background

The chapter analyses the background required for this thesis. It starts from the introduction of shadow rendering and then the various methods for shadow rendering. After that it continues to describe the better approach for shadow rendering. It also includes the theory on comparison between PCSS and few other techniques.

2.1.1 Shadow algorithm for computer graphics

Shadows are a great promoter of conception and improving realism in computer images. Shadow calculation, categorising objects to shadowed or unshadowed regions and incorporation of shadow volumes are the three approaches for categorising shadow algorithms (3). The research by Lance Williams discusses the three approaches which were present in that year for casting shadows. This provides a significant idea about the characteristics a shadow rendering algorithm should have which is relevant in today's time also. In that time there were not so many algorithms which can generate shadows so majority of computer shaded images were generated without shadows which was considered as one of the major limitations. Casting a shadow will make a huge difference in the scene since it can make the image look more realistic. The first category of algorithm which is invented by Apple is capable of perceiving boundaries as raster scan is used to develop the image.

2.1.2 Casting Curved shadows on curved surface

To enhance the accessibility of a scene, electron microscopy shadowing is being used (3). To determine shadows in computer scenes, multiple methods are available. Relative position and shape of the object can be more intelligible with rendering shadows. According to Lance Williams, (3) algorithms which were there in that time to render shadows have the limitation of constructing planar polygons. This paper proposed an algorithm which uses z-buffer. Visible surface for z-buffer will be used to calculate shadow by smooth surface patches which modelled the object. To render a scene without shadows is much easier than with shadows, because

rendering the scene with shadow will double the computation and hence cost will be high (8). The algorithm which is proposed in this paper can generate shadows even if the surface is curved, and this will cost less as no extra utilities are added to the scene. For excellent outcomes all the operations should be handled carefully. With this at different stages the result of dither, geometric quantization and interpolation should be assessed properly. The algorithm discussed in this paper is as follows:

- As per light point of view the scene will be designed. Instead of shading values just z values will be calculated and stored.

- As per observers point of view the scene will be built. To map X,Y,Z to observers X,Y,Z view perform linear transformation. For each point calculated the visibility to light will be checked as it has to be transformed as per observers view. It is considered in shadow if the calculated point is not in vicinity of observers view.

The proposed algorithm can face quantization issues as quantization of visible points to the resolution of the z buffer will be performed. The complexity of the scene will not decide the cost of alteration as it will be decided only after all points are calculated. Only the points which can be seen in the final scene will be transformed as a part of post process operations. So, the resolution of the image will decide the expense.

The user must ensure that all the objects which might emit shadow must be inside light source of the image. Shadows can be rendered only for the points which are in vicinity of the light source. Sphere illumination has to be sorted to various views if the casting of shadows in every direction is inside observers light source. This computations will be a bit costly as compared to calculating single view which will have all objects from the scene. When a point gets shifted to a light source surface from the observers space preferably it should settle on surface of which it is a part of. It will either be below or above the surface the reason behind this is z buffer quantization and inaccuracy of arithmetic capabilities of the machine. A bias gets subtracted from the z value of the point which is an outcome of light source space to get targeted point which will emerge as illuminated.

2.1.3 Shadow Maps

Shadow maps are an extensively used implementing technique for capturing graphics in a real-time scenario (9). However, one limitation of these maps is that they need differential filtering from colour textures, which may cause serious aliasing problems. In order to overcome these limitations, conceptualized variance shadow maps are used. These maps store the mean and mean square values of depth in a statistical distribution which makes it easy to efficiently

compute the variance and in turn, derive the upper bound of the occluded portion of shaded fragment. This value is found to be consistent for the purpose of rendering. Thus, the aliasing problems are eliminated and is efficient to implement (10).

2.1.4 Accurate Shadows

PCSS is a shadowing technique used for generating sharper and accurate soft shadows from light sources (11). The main advantage of these shadows is that there is no requirement for pre-processing, post-processing and it does not need to factor in additional geometry (12). This technique is a stable alternative to traditional shadow map query by not considering the degree of complexity and can incorporate displacement mapping. This technique can be deployed on current consumer graphics. The three-step algorithm (Blocker search, Penumbra estimation and filtering) is found to perform effectively for more genuine scenes with a high level of interaction.

Shadows are recommended due to their ability to increase the understandability of images and add a new layer of realism to computer graphics. Shadow algorithms can be of three types: scan out; demarcating objects into portions with and without shadow and incorporation of shadow volumes in the object. The third approach is found to dominate the other two approaches in shadow rendering capability (6).

Exploiting the temporal coherence which is inherently present in scene movement allows for estimation of a new shadow value only when regions are reopened following camera adjustment or when the shadow situation changes due to object movement (13). This novel method efficiently calculates soft shadows in real-time applications by overcoming the high computational effort required for visibility estimation of each frame. All moving objects can be detected along with the soft shadows they cast by extending the typical shadow mapping algorithm with an additional light-weight buffer for tracking dynamic scene objects (14). By employing this strategy to the popular PCSS, the rendering performance can be doubled while preserving the visual quality in scenes with both static and dynamic objects, which is commonly seen in various 3D game levels.

2.1.5 Percentage-Closer Soft Shadows

As a solution for aliasing issues percentage closer filtering can be used (15).For an accurate shadow mapping algorithm filtering the depth maps comparison rather than filtering depths is require. As object contacts each other and as they move apart they get more blurry and the shadow becomes sharper. To get the softer shadows PCF kernel size has to be increase.

The crucial stage in this algorithm is approximating penumbra.PCSS uses parallel estimation and similar triangle theorem. As per light size and blocker/receiver distance from light the penumbra width is calculated. The formula to calculate this is (7) ,

$$w_{Penumbra} = (d_{Receiver} - d_{Blocker}) \cdot w_{Light} / d_{Blocker} \tag{2}$$

2.1.6 Parallel-split Shadow Maps

This technique is based on z-partitioning method which focuses on estimating perfect distribution with the help of shadow maps. The structure can be seen in Figure 1. In this method view frustum gets divided into many sections with respect to z-axis and then individual fixed sized shadow map will be computed for every sub-frustum. By following this technique the density of sampling gets minimize for every consecutive sub frustum as larger and larger region get covered by shadow map with same resolution. The important part in this method is how to divide split planes. It can be done by inferring resampling error of the shadow map. With the reference of big texel from global shadow map which will lead to z-partitioning let arbitrary function of democratization. To calculate the split position, the following equation (16) will be used,

$$C_i^{log} = z_n(\frac{Z_f}{Z_n})^{\frac{i}{m}} \tag{1}$$

In equation 1 number of partitions are denoted by m. Because this assigns resolutions closer to plane which will increase the times occupancy with the scene. A better approach for dividing scheme logarithmic and simple equipartition scheme has proposed in (17).

2.1.7 Parallel-split Percentage-closer Soft Shadows

For large scale virtual environments if soft shadow has to be rendered parallel-split PCSS can be used (7) . In this method entire view depth is considered as multiple range of layers and shadow map size will be decided based on varying filter kernel with multiple shadow maps. With parallel split PCSS assessing resources and using resolution of shadow map is preferable. This process will have five steps which are as follows:

- **Step 1:** Dividing view frustum which will result into multiple depth sections.
- **Step 2:** Distributing frustum of light into various minute parts where it will cover single part and shadow will be cast for that particular part.
- **Step 3:** For every divided part shadow map will be cast. Then we have to loop through shadow map and depth values which are near to light as compare to shaded point and perform mean operation on them.
- **Step 4:** Calculating width of penumbra with help of parallel planes estimation.
- **Step 5:** With changing the kernel size execute PCF step on depth comparison which is proportional to width of penumbra and the size of shadow map. After which shadow for entire scene will be rendered.

For split schemes there are three views which can be seen in following image.

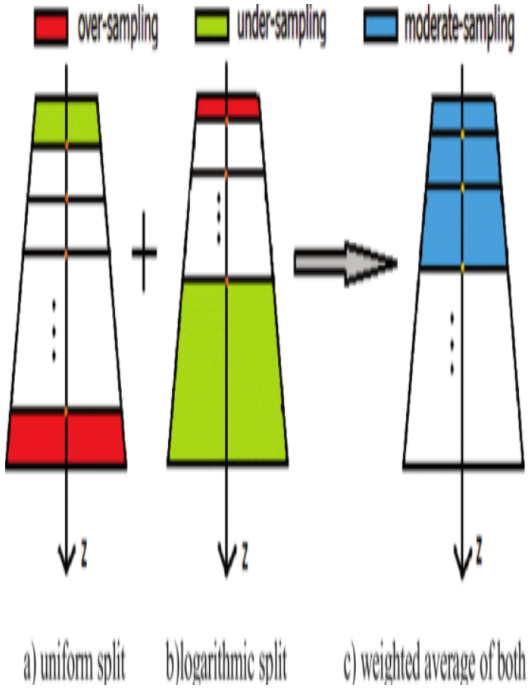

Figure 1: Split PCSS

3 Methodology

3.1 History

The history of shadow rendering began in 1977 by Frank Crow. Despite proposing the idea of the technique it was not frequently used in images or scenes to create shadows and it became popular after its use in Doom 3 which release in 2005. The method used in Doom 3 for rendering shadows is known as Carmack's Reverse. The effect of shadows in Doom 3 can be seen in Figure 2. Shadow mapping was invented in 1978, one year after shadow volume

Figure 2: A scene from Doom 3

technique was proposed. One of the older methods to render shadows is projection shadows. In this method the shadow of object will directly be projected on floor. The only concern in this method is that in order to render the shadow the floor has to be flat.

3.2 Casting Curved Shadows

In 1987, Williams L. talked about casting curved shadows on curved surfaces (3). This paper is the foundation to the extensive research carried out in shadow rendering. This research

focuses on z-buffer visible surface algorithm which was proposed by Catmull (18),it was the first method to develop computer-generated shaded images of bicubic surface patches. Though the algorithm is exceedingly simple to implement but its not memory efficient. In computer graphics the way to store digital pictures is "**frame buffer**". The z-buffer is an add on to mass-memory method in computer graphics which is about visible surfaces. This technique will store the depth data at every point in the picture.

3.3 Composition of Shadow

The shadow is combination of **umbra,penumbra** and **antumbra** (19). To render a shadow which will be closer to real world these three parts should be rendered properly. The region

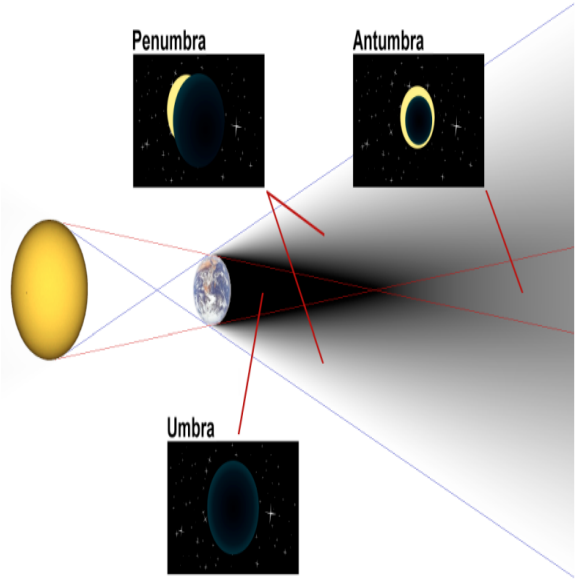

Figure 3: Composition of shadow

where a light is completely blocked by the object is known as **umbra**. Umbra is the darkest shadow as the light cannot reach in this area. There is a lighter region next to umbra which is called Penumbra where just a portion of light is blocked. Some regions get highlighted as the light passes which is called partial eclipse. The last type of shadow is Antumbra. In this type the object will appear as if it is completely covered by light. If it shifts directly to umbra region from antumbra region then size of the blocked object will keep increasing until it reaches umbra region. It is computationally expensive to calculate all the components of shadow. This can be seen in Figure 3.

3.4 Hard vs Soft shadows

Considering a source which emits light by single point and there is a single ray hitting the floor. In this case shadows can be generated of two types **Hard or soft shadows**. Usually all the shadows which are developed will be hard by default. When hard shadows are the shadows which are not suggested for use (20). Hard-edged shadows can be achieved by ray casting method. These kind of shadows will not give a realistic effect on the scene.The effect of hard shadow can be seen in Figure 4.

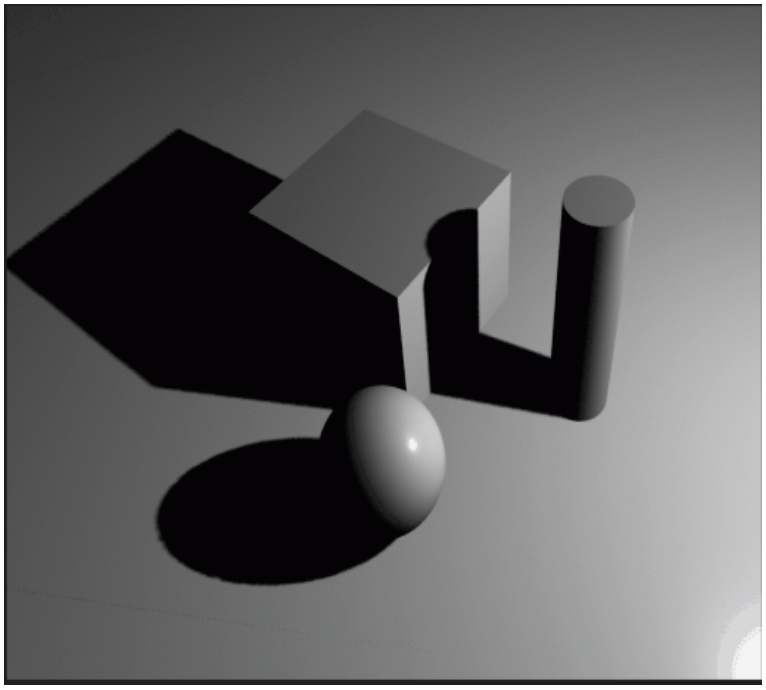

Figure 4: Hard shadows

In hard shadows the shift in shadows and light will be sharp and metallic.Here are some scenarios where hard shadows can be used:

- To generate hard light and shadow on bright sunny day.
- When there is need for developing fixed shapes which will have defined sharp shadows.
- In space where light reaches to object without getting diffused.
- In movies or circus when a harsh focus is needed with the help of spotlight.

The drawback of hard shadows is that they are less realistic. To achieve realism in images

or in scenes **soft shadows** can be used as they have softer effect on the edges. These kind of shadows tend to fade off which gives them a realistic look. In Figure 5 the effect of soft shadow can be seen. Here the shadow is not as dark as hard shadow shown in Figure 4, it is slightly faded which is giving realistic effect to the objects. Here are some scenarios where

soft shadows can be used:

- Showcasing the light which is transmitting by transparent materials.
- When there is need to generate natural light in cloudy weather.
- Lights which gets reflected from the surfaces which generally have soft effect.
- In movies to make characters more realistic and relatable.

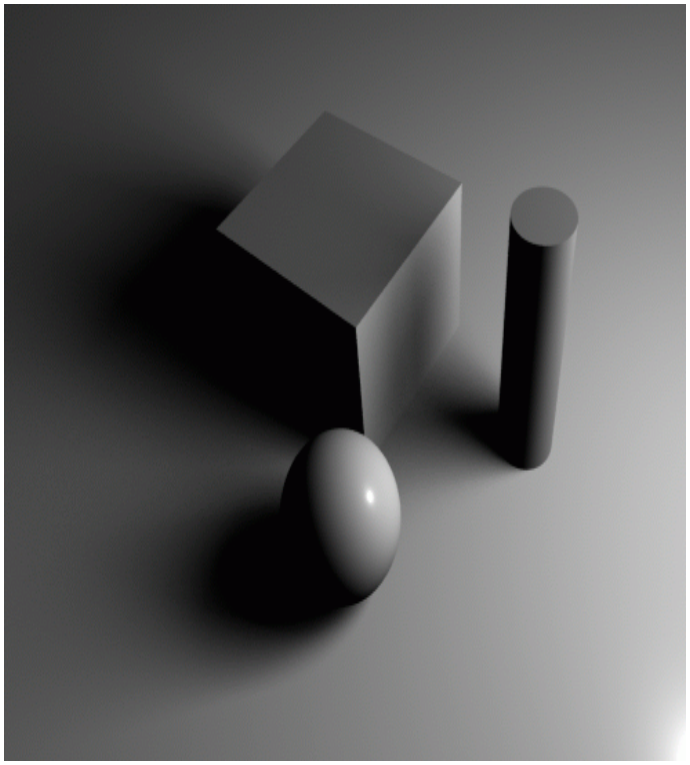

Figure 5: Soft shadows

In soft shadows the source of light will be considered as an object so the rays of shadow will not look as if they are straight, instead it will look like cones. In following image we can observe that only umbra element is present in hard shadows contradictory to this soft shadows are consists of umbra and penumbra, which is the reason for their realism. The structure of hard and soft shadow can be seen in Figure 6.

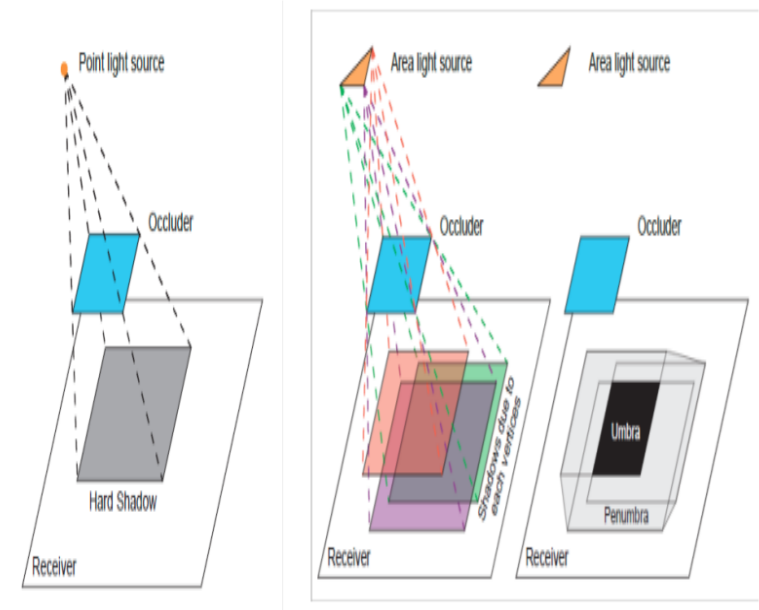

Figure 6: Architecture of soft and hard shadow

3.5 Soft Shadows with a Single Shadow Map

Rather than examining source of light and rendering shadow maps at every sample as a solution **single shadow map** can be used. In the region of projected point it can be sampled multiple times after which all the outcome can be merged.

3.5.1 Blurring Hard-Edged Shadows

To achieve softer edge effect blur the edges of shadows which are hard. The best way to achieve this is shadow map. Comparing shadow maps multiple times with offset coordinates which will followed by merging all the outcomes. The offset here plays a very vital role as this will act as a decision maker on blurriness of the final shadow. Because of the depth values the original data of shadow map can not be blurred. Though by executing multiple lookup at every iteration the results of shadow map can be blurred in a scene. This technique is very similar to traditional soft shadow rendering technique in which the object will be rotated by 90 degrees and the high computational shadow map will become less expensive to compute. Sadly, only when penumbra area is small enough this method is suitable or else it will have banding artifacts which is shown in Figure 7.

Figure 7: Banding effect on shadow

3.6 Techniques

In this division lets look at the various techniques used for shadow rendering.

3.6.1 Shadow Mapping

In this technique lights' perspective is used to render the shadows in scene. In simple way things which are visible from lights viewpoint will be lit and things which are not visible from lights perspective will render as shadow. Consider a surface which have box in between light and the box on the surface. As there is a box in between as per the light direction there will be floor section which will be in shadow. This can be seen in Figure 8.

In the figure all the regions which are visible by light are shown by blue color where as the black color represent the blacked area which can't be seen by light.Lets say if a line is drawn from light source to the object which is on the floor the light will hit the middle object first before reaching to floor object which leads to middle object having blue color and bottom object having mix of both colors. We have to get the **closer point** of the ray where it will hit the middle object and then use this point to compare with other point on ray.Then keeping this closer point as a reference we can determine whether the point is in light or shadow. The problem with this method is that looping through every point on light ray is not feasible and it will be exceedingly difficult to compute this much data. But instead of computing light rays depth buffer can be used. If we consider camera's point of view and fragment set to [0,1] from it that will be values depth buffer. Shadow map is the image where the **z values** values are stored for every pixel of the scene. The transposed light coordinates will be calculated and if it is less than one of the values in shadow map, it is in shadow or else it is lit. Depending on the pixel present in shadow or not the color is decided. If the light source

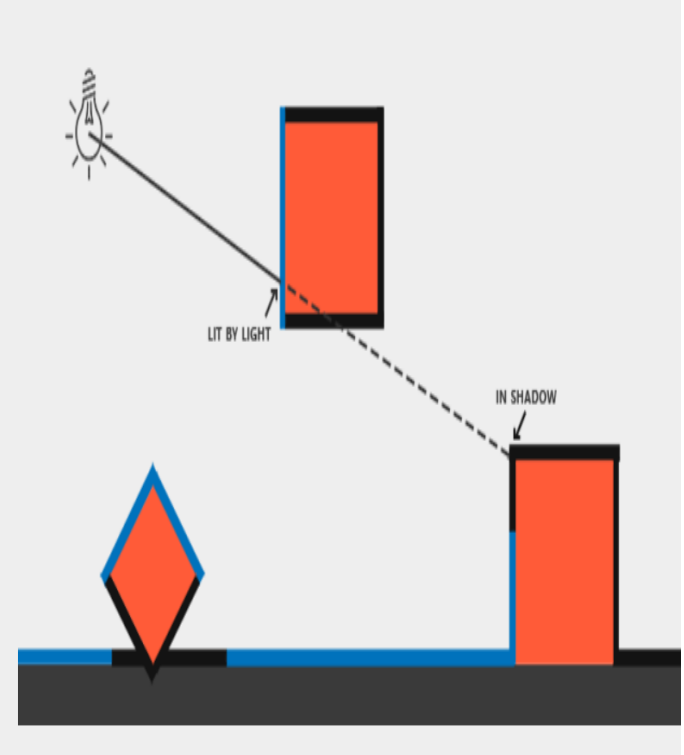

Figure 8: Shadow mapping with blocked object

is definite in correspondence to the viewer it will lead to aliasing issues.This technique can be purely developed with the graphics hardware.

3.6.2 Shadow Volume

This is a geometry based method to render shadows. In shadow mapping you need to calculate shadow map direction vector and because the light will be distributed everywhere it is hard to compute this vector. This problem can be solved by using shadow volume. This method is invented by Michael Songy and William Bilodeau (21).For masking off the pixel this technique uses stencil buffer which capable of rendering pixels even if there no light source (21). Shadow rendering uses the approach to enhance the object silhouette. When light drops on the silhouette of object, it will be enhanced after which using some simple stencil operations, volume will be rendered using stencil buffer. This can be accomplished by iterating through every adjoining edges. The front polygons wins the test if the object is in shadow on other side back polygons will fail this test.This is done calculating dot products of normal's of triangles and opposite edges of triangles of light source. Then the common edge which is silhouette edge will be inserted in a list. **Depth Pass** was previously discussed, but due to its inefficiency, the other option which was used to develop stencil operations is **Depth Fail**. The disadvantage with depth pass is that it is not efficient when the viewer is inside shadow volume. This problem can be solved by depth fail.In Figure 9, at bottom left there is a bulb

and a object which is green in color. This object is responsible for rendering the shadows because its blocking the light. In total three objects are there A, B and C. Object A and C are responsible for shadowing B. To render the shadow by **stencil buffer** we have to cast green color object with A, B and C. All this objects will get stored in **depth buffer**. As an outcome of this process we will get all the nearer pixels. To generate a shadow we have to iterate through all the objects. The following rules are used while rendering the shadows with shadow volume technique:

- Value in stencil buffer get incremented if the test fails while casting back facing polygons.
- Value in stencil buffer get decremented if the test cases while casting front facing polygons.
- If stencil test fails and depth test pass no operation will be done.

Consider object A, it fails the depth test as it is surrounded front and back facing triangles. Values of pixel get incremented or decremented as it is surrounded by object A which will eventually become zero. Front facing triangles of object B depth test will win the test and back facing triangles will fail the test as a result stencil value get incremented. In case object C depth test wins and therefore no change will be there in stencil value. Here we considered only one blocking object but there can be n-number of blocking objects.

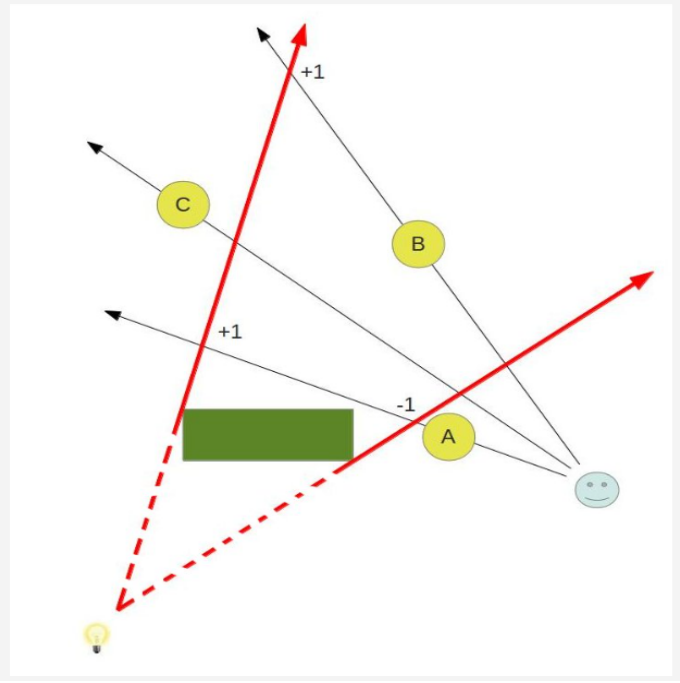

Figure 9: Depth Fail method architecture

3.6.3 Percentage-Closer Filtering

This technique uses projection of current screen pixel. This pixel enlarges into shadow map then sampling is performed on resulting area using standard texture filtering. All the sample which we get as an outcome gets compared with reference depth which will generate binary result. The result of this comparison gets merged to calculate texels percentage in filter area which is near to reference depth. To render the light this percentage is used. In 1987 Reeves come up with the idea that is you have to achieve soften edges of the shadow by enlarging the size of filter area (22). It can be seen in Figure 10.

Figure 10: Soft shadow effect with PCF

Soft shadows can be achieved by making size of filter minimum. If the filter size is large it will need shadow map which have high resolution to get every detail preserved. Fix area of adjacent shadowmap texels sample is a commonly used variant in algorithm. This technique solves the problem of aliasing by using **depth maps**. This technique will be explained in detail in coming sections.

4 Implementation

4.1 Percentage-Closer Soft Shadows

Shadows contribute to strengthen the realism in images created by computers. However, shadow rendering algorithms which are available in the market have the limitation on modeling the objects and they are extortionate to execute. PCSS is based on z buffer algorithm which is proposed by Williams (3) and it works in two stages. Simple scene point of view can be seen in Figure 11 (15). As per the point of view of light source a scene is casted, with this z-buffer will store values of z for objects which are closer to light, this will be done in first stage. As part of the second stage, the scene will be rendered as per camera position. Every point is altered to light source space from surface. After which, the object point which is nearer to light is determined. This will happen as per the point stored in depth maps. If the altered point is not in front of z which is stored in depth map then the point is declared as in shadow. For implementing the algorithm which is proposed in (3) two aliasing issues can be faced, the first one is producing depth maps and the second one is sampling these depth maps.

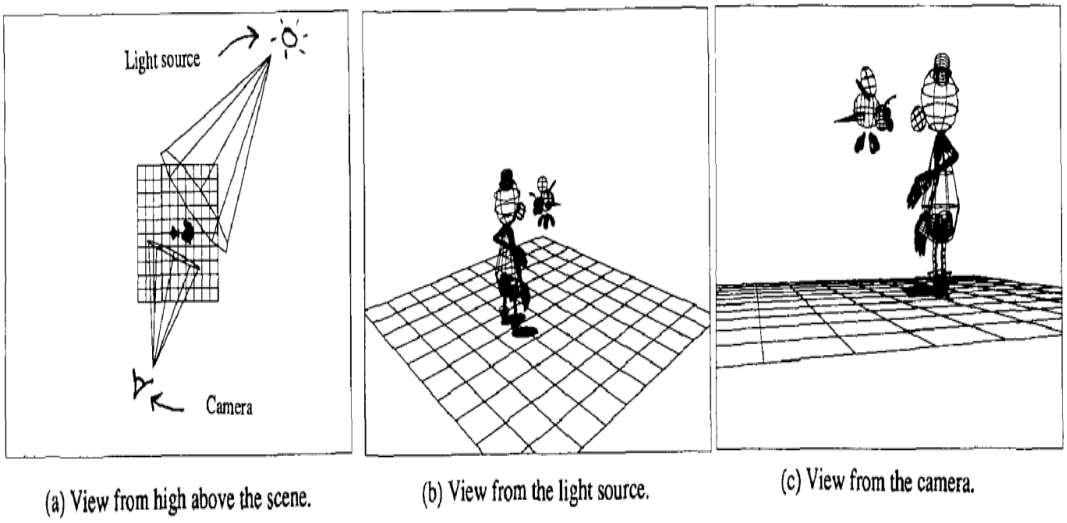

Figure 11: Simple scene point of view

To implement PCSS technique following stages are required:

- **From the bounding box of the area randomly selecting samples.**
- **From the same box selecting the sample from one of the distribution, for example Gaussian.**
- **Distributing the bounding box to subareas and performing sampling for every partition.**
- **Making sure that geometry boundary of the area will have single sample position.**

In this technique average of various shadow map comparison are computed. The percentage of the surface nearer to light which means that it is not shadow will get calculated. This method was originally invented by Reeves in 1987 (15) which is stated as sampling the area by involving random variable and mapping the area which is shaded in the shadow map. The aliasing issue can resolve by implementing stochastic sampling (23). For sampling, percentage closer filtering will be used. For acquiring texture values texture maps can be accessed which also filters this values. Though with depth maps this technique will not work, the reason behind this is the values from depth map will be compared with surface values to check whether the point is in shadow or not. With this generating soft deinterlacing edges are not possible as the outcome of the above comparison will be in binary. In solution proposed in (15), the order of comparison and filtering steps get reversed. All the z values present in depth map for the area are get compared with surface depth which are going to rendered. We will have binary image which is converted from depth map as part of sample transformation process. To get a proportion of the area in a shadow, from this results values get filtered and as a outcome soft antialiased shadows get rendered. The major distinction between normal texture map and percentage closer filtering can be seen in figure 12.

4.2 Algorithm implementation

The Percentage closer filtering works as following:

4.2.1 First step : Creation of depth maps

By calculating depth values for every objects from the image for each light source the depth map is created. X and Y coordinates signifies the location of pixel in the depth map as

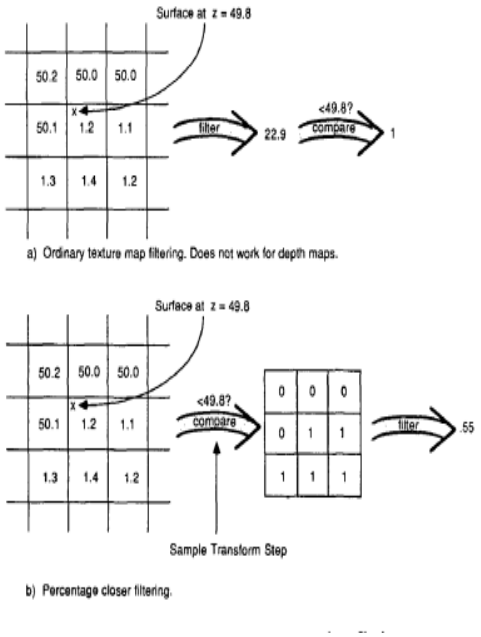

Figure 12: Implementation of Percentage-Closer Filtering

per screen space of light. The distance from light in world space can be represent by z-coordinate. To represent x,y,z values **light space** is used. While producing depth map rather than using stochastic sampling regular sampling will be used. The reason behind using stochastic sampling is getting relevant sampling. Sometimes the outcome with regular sampling is less noise per pixel as compare to stochastic sampling. Its not hard to execute first pass. To get better results rather than storing color values, nearest z value gets stored this can be seen in Figure 13. Texture file will save this floating point value. As compare to shaded image depth map takes less time for execution. With depth map there is no requirement of texturing, lighting and shading computations and objects which are unable to cast shadows are completely disregard. The best part of depth maps are it only requires single sample per pixel.

This algorithm is implemented in openGL with `Function_depthBlock`. Which can be seen in following code snippet.

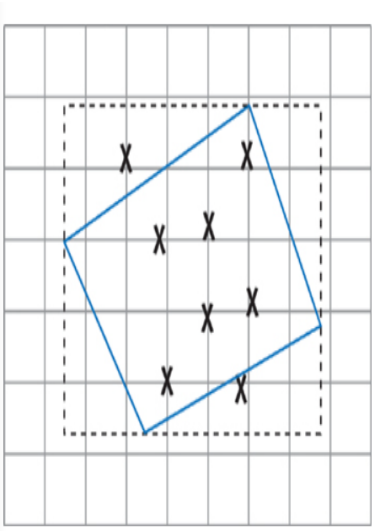

Figure 13: Representation of depth search.

```
float Function_depthBlock(vec4 varLightSpace, float light_wValue)
   {
    int numberOfSample = 8;
    float addingBlocker = 0;
    int countForBlocker = 0;

    vec2 texelVariable = 1.0 / textureVariable(shadow_map_variable
       , 0);

    vec3 coordinatesProjection = varLightSpace.xyz / varLightSpace
       .w;
    coordinatesProjection = coordinatesProjection * 0.5 + 0.5;
    float depthValue_current = coordinatesProjection.z;

    float rangeForSearch = light_wValue * (depthValue_current -
       0.05) / depthValue_current;
    if (rangeForSearch <= 0) {
        return 0;
    }

    int rangeValue = int(rangeForSearch);

    int windowCount = 3;
    for (int i = -windowCount; i < windowCount; ++i) {
        for (int j = -windowCount; j < windowCount; ++j) {
            vec2 shiftValue = vec2(i * 1.0 * rangeValue /
```

```
            windowCount, j * 1.0 * rangeValue / windowCount);


        float sampleDepthValue = texture(shadow_map_variable,
            coordinatesProjection.xy + shiftValue *
            texelVariable).r;

        if (sampleDepthValue < depthValue_current) {
            addingBlocker += sampleDepthValue;
            countForBlocker++;
        }
    }
}
```

4.2.2 Second step : Scene Rendering

As per the camera point of view a scene will be created in second step.Every computation done for shading associated with some area on surface. What each area represents is dependent on which rendering system is being used. It can represent correct pixel areas to small sub pixel areas. Every region which is shaded will be depicted in light space and will have a area in depth map too. The main idea behind using percentage filtering is deciding nearer values to the light than surface from all the z values from depth map. As per the outcome the intensity of light gets decided. This process gets repeated if shadows are casted by multiple lights in the scene.

To implement this algorithm initially REYES rendering engine was used so the area which is shaded can be represented by four-sided micropolygon (24) shown in the Figure 13.

4x4 texel sample area can be used everywhere rather than computing shaded area for the shadow map (24). The resultant area will be notably sufficient enough to minimize aliasing issues. However it cannot achieve good result as it is not that capable enough to store many samples. As hardware shader's do not work on micropolygons instead they consider pixels while operating.As per the original implementation conversion the four sided polygon to the pixels of shadow maps is necessary to calculate sample area. Instead in this implementation fix size sample areas are considered. These samples can allow us to skip operations which are complex in nature. By using this way accurate shadows percentage will be calculate rather than roughly considering probabilistic sampling (24).Figure 14 is showcasing the sampling for while going through the process (24).

To implement the PCSS this is referred (25). Function **calculatingShadows** is used in openGL, the following snippet shows the implementation.

```
float calculatingShadows(vec4 varLightSpace)
{
    // perspective_divide_calculations
    vec3 coordinatesProjection = varLightSpace.xyz / varLightSpace
       .w;

    // transforming operations for rangevalue [0,1]
    coordinatesProjection = coordinatesProjection * 0.5 + 0.5;

    // getting nearest depth value
    float nearest_depth = textureVariable(shadow_map_variable,
       coordinatesProjection.xy).r;

    // calculating depth as per current light perspective
    float depthValue_current = coordinatesProjection.z;

    // biasValue computation
    vec3 normalVariable = FunctionNormalize(NormalValue);
    vec3 directionOfLight = FunctionNormalize(positionLight -
       positionFragment);
    float biasValue = max(0.05 * (1.0 - dot(normalVariable,
       directionOfLight)), 0.001);

    float lightW_value = 7.0;

    float depthAvgValue = avg_block_depth(varLightSpace,
       lightW_value);
    if (depthAvgValue == 0) {
        return 0.0f;
    }

    float penumbraWValue = penumbraWValue(depthValue_current,
       depthAvgValue, lightW_value);

    //considered as shadowValue if value of penumbra is zero
    if (penumbraWValue == 0) {
        return 0;
```

```
    }


    // checking if position is in shadowValue or not
    float shadowValue = 0.0;
    vec2 texelSize = 1.0 / textureSize(shadow_map_variable, 0);
    int counterValue = 0;

    //deciding penumbra width.
    int rangevalue = int(penumbraWValue / 0.09);

    //set a rangeValue if its too big
    rangevalue = rangevalue > 40 ? 40 : rangevalue;
    rangevalue = rangevalue < 1 ? 2 : rangevalue;

    //return penumbraWValue;

    for (int counter = -rangevalue; counter <= rangevalue; ++
       counter)
    {
        counterValue++;
        for (int iteration = -rangevalue; iteration <= rangevalue;
            ++iteration)
        {
            float depthValue_pcf = textureVariable(
               shadow_map_variable, coordinatesProjection.xy +
               vec2(counter, iteration) * texelSize).r;
            shadowValue += depthValue_current - biasValue >
               depthValue_pcf ? 1.0 : 0.0;
        }
    }
    counterValue = counterValue > 0 ? counterValue : 1;
    shadowValue /= (counterValue * counterValue);


    if (coordinatesProjection.z > 1.0)
        shadowValue = 0.0;


    return shadowValue;
}
```

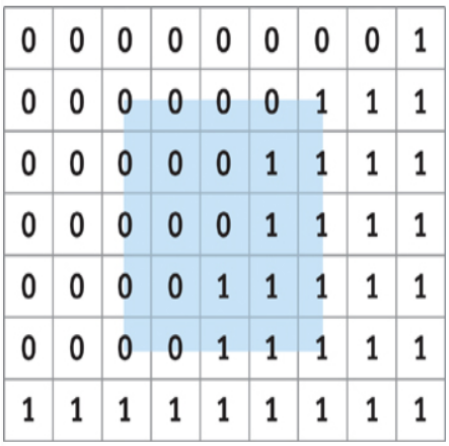

Figure 14: PCSS sampling process for 4x4 region.

4.3 A Brute-Force Implementation

Percentage closer filtering for shadow map sampling is integral part in NVIDIA GPUs. To bilinearly interpolate the shadow map fractional part of the texture coordinate is used with this comparison of four depth will be considered by hardware. The outcome of shadow is percentile of texel size region is shown in Figure 15.

To project texture map which can be analyse by offset present in texel unit. If we have to calculate **texmapscale** it can be done by float2 value which is 1/width and 1/height of shadow map.

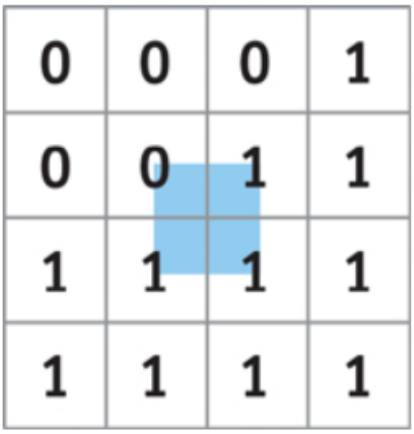

Figure 15: Using hardware for processing shadow map

To estimated penumbra parallel planes are used (26). This calculation is useful for estimation distance between size of light and receiver with respect to light. This is calculated by following lines of code,

```
float penumbra_calculations(float receiverPos, float blockerPos,
   float lightPos) {
    //calculating penumbra estimation and return the value
    float estimated_penumbra;

    estimated_penumbra = (receiverPos - blockerPos) * lightPos/
       blockerPos;
    return estimated_penumbra;
}
```

4.4 Using Fewer Samples

The brute-force method gives better results that one might think.Textures picked from texture cache as there is assurance that it will be near each other. If the sampling pattern is updated expected result can be achieved by four samples per pixel. The result which can be achieved by dithering black and white data same result can be provided by four sample technique to cast a grayscale image. Only four samples from 16 sample for each pixel will be used by keeping sample area size unchanged. According to screen location the set of four samples will keep changing.

0	3	0	3
2	1	2	1
0	3	0	3
2	1	2	1

Figure 16: Sampling Pattern

Figure 16 depicts the pattern for sampling which is four sample which is picked from 16 possible locations per pixel.

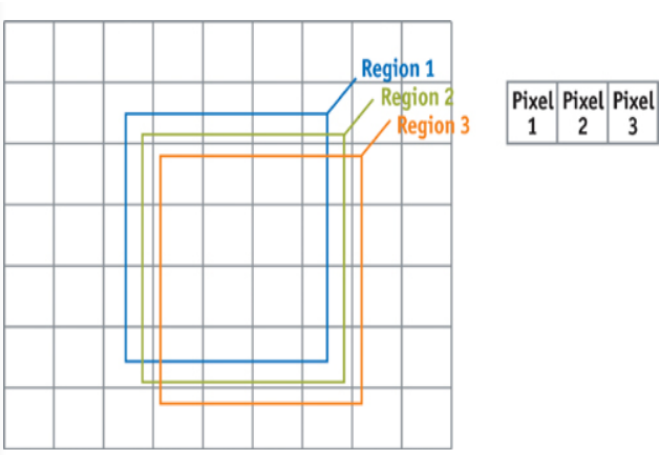

Figure 17: Using hardware for processing shadow map

4.5 Reason behind its working

Can texture projection magnifies the shadow map if antialias shadow with a fixed size sample region? The solution is, there will be less difference between shadow values if the texture map samples are near to each other and the region which are sampled are significantly huge. The reason behind this is there will be overlapping in sample regions. The way overlapping occurs in adjoining pixel after magnification of shadow map.This can be seen in Figure 17

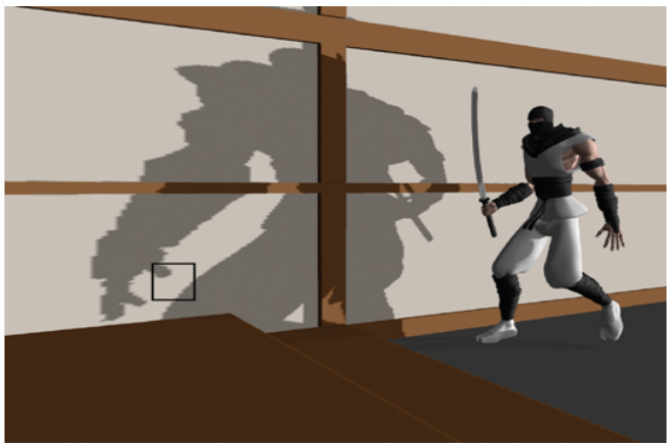

Figure 18: One sample per pixel Ninja Shadow

The difference between adjoining pixels getting smaller and transformation between unshadowed and shadowed areas will be smooth if shadow map is magnified (27),There will be always a smooth shadow with utmost magnification because the hardware computes percentage of shadow by eight bits of precision. For eight bits per component the least notable bit will have some difference. This is shown in following images. The following image figure 18 , 19 and 20 (24) shows the samples of ninja model.

On the other hand figure 19 is depicting thumb shadow of ninja. If observed clearly

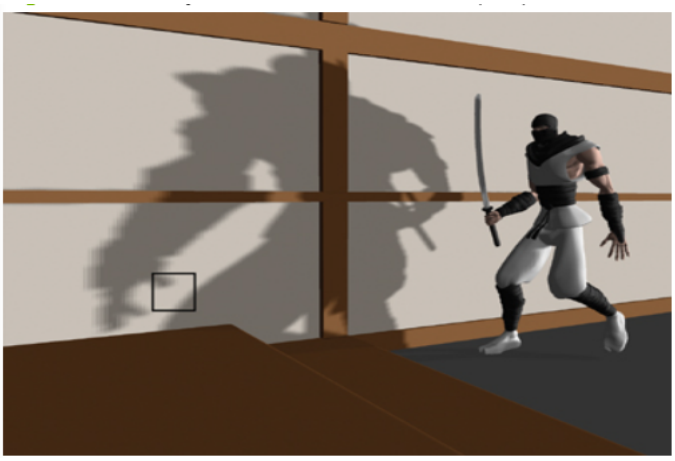

Figure 19: Four dithered samples per pixel ninja shadow

the picture with 16 samples has better quality of shadow. For the desired shadows filter size can vary with samples. Executing the application with more samples requires **more computational cost**. As the sample count various the change in shadow is obvious. The sample count can be change as per the requirement. This will lead to storage issue as for every sample data gets stored.

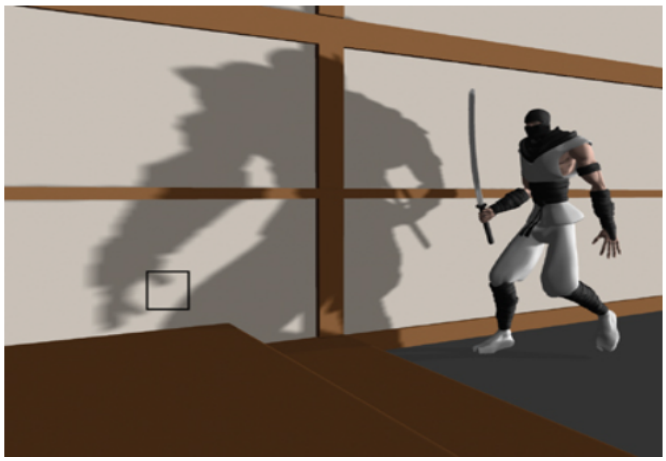

Figure 20: Sixteen samples samples per pixel ninja shadow

4.6 Challenges with Percentage-Closer Filtering

- **Transforming Region**
  Alteration of the area which is out of depth map is challenging. To solve the issue if the area is out of source of light it will not consider as shadow.Complex mapping scheme is used to cast the shadow in all directions.

- **Reduction of banding effects**
  When penumbra size is huge and insufficient for samples which were stored while PCF process and as a outcome if there is added blur this will induce reduction of banding artifacts.

- **Shadows are costly to render**
  The major issue while casting the shadows is, it is super costly to generate shadow.Though current graphics card are super efficient and highly capable of generating accurate shadows. Still the issue of complexity exists while generating shadows.

- **Estimating penumbra width**
  The area inhabited by shadow will be smaller, if the space between viewer and light is enlarge. So according to the distance penumbra width has to get recalculated which sometimes hard to manage.

- **With multiple lights is hard to render shadow**
  If multiple light sources are present in the scene, it is highly complex to render shadows. Virtual cube map which is supported by hardware is capable of developing shadows which have great quality even if hundred light sources are in used.

- **Aliasing**
  Aliasing is one of the issue which often gets with the silhouette of the shadow (28). Whenever there will be mismatch of pixel. As a solution to this issue is sampling ,filtering and portioning can be performed (29).

5 Evaluation

Following are the results for the implementation done for PCSS. This implementation is tested on various object to analyse the resultant shadow. From various objects which are used as a testing samples, shadows for two objects are discussed below.

5.1 Experiments

A floating value will be returned from PCSS implementation as a alternative to conventional depth maps. With the use of various shadow maps which attempts to implement distribution of samples which is similar to z-partitioning. In this method as examination, with respect to z-axis view frustum is divided into multiple sections. After sectioning the frustum distinct shadow maps will be computed for every sub frustum. For the accurate shadow results filtering of the outcomes from the depth comparisons rather than just comparing depths plays a vital role (30) and this what is considered while implementing this research. If the objects are moving apart from each other shadow gets softer, on the other side if objects coming near to each other the shadow gets sharp edges. As the value of PCF kernel getting expanded the shadow will be rendered as softer. If all these findings are considered, though the PCSS follows conventional shadow mapping methods, it allows for experimenting on **filter size** till the precise level of softness is achieved.

To estimate the place of split planes following formula (30) is used:

$$C_j = \lambda C^{uniform}_{\sum_{k=1}^{j} l_k} + (1-\lambda) C^{log}_{\sum_{k=1}^{j} l_k} = \lambda[z_n + (z_f - z_n)\sum_{k=1}^{j} l_k/m] + (1-\lambda) z_n (\frac{Z_f}{Z_n})^{\frac{\sum_{k=1}^{j} l_k}{m}}, \forall 0 \leq j \leq M()$$

5.2 Results

This implementation is done in openGL and follows shader based approach. Some of the outcome of the PCSS implementation are discussed below. The light source will keep moving and the changes can be seen in the shadow which is being casted. Though the shape of the object is getting rendered properly but after careful examination the minute details of the lamp specially near the pole its not rendered properly. To test exactness of the minute details testing this implementation on multiple objects is necessary. Objects which have complex geometry structure are difficult to render but with the result it is proved that PCSS managed to generate decent enough soft shadows.

5.2.1 First Object : Lamp

The first object used for testing is lamp which includes detailing in its design. Usually if a object having this kind of detailing its hard to render its shadow (31). With PCSS the results are good enough but if closely observed there is noise involved and the shadow is slightly blur at the edge specially at the pole of the lamp.

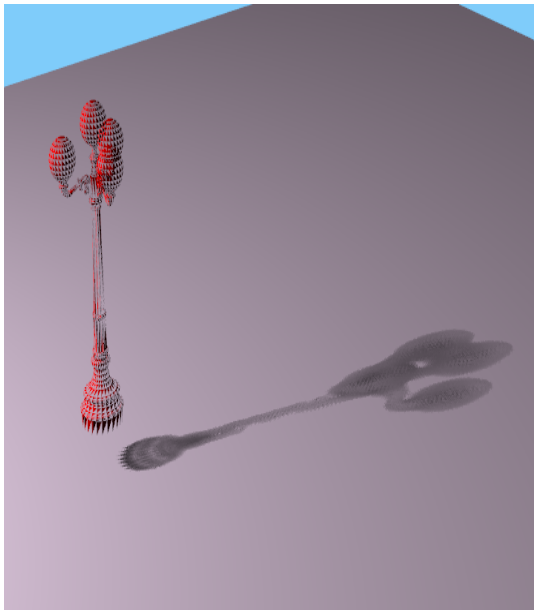

Figure 21: PCSS Lamp shadow result 1

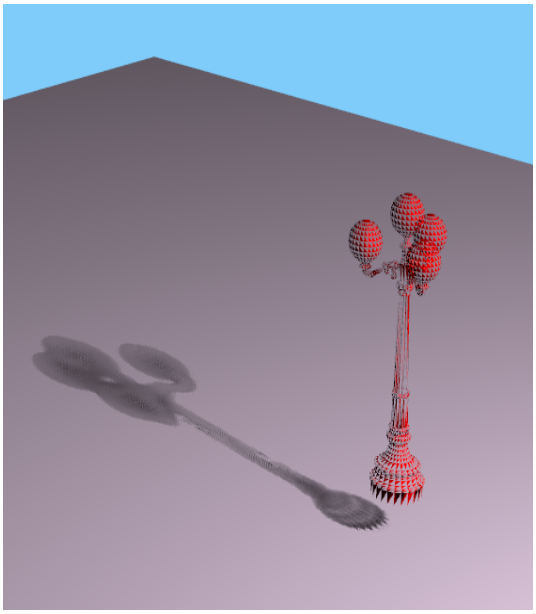

Figure 22: PCSS Lamp shadow result 2

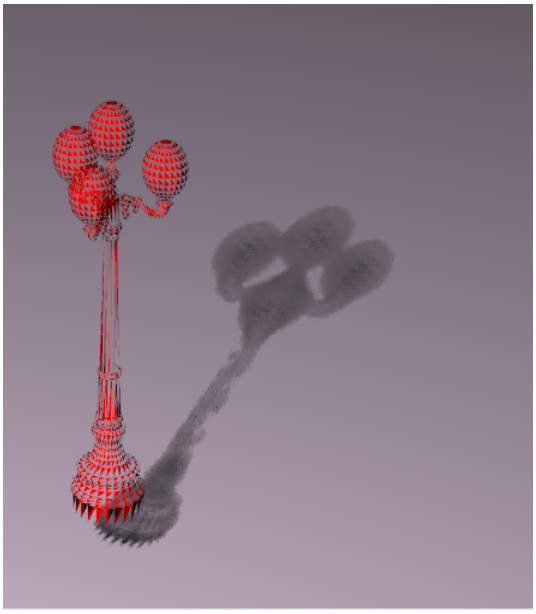

Figure 23: PCSS Lamp shadow result 3

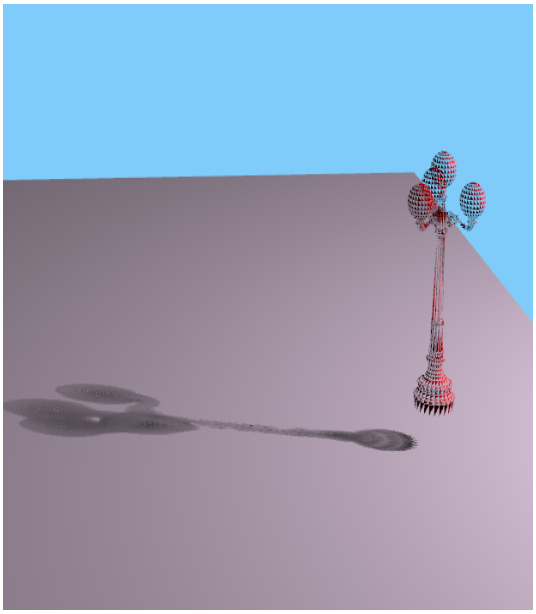

Figure 24: PCSS Lamp shadow result 4

5.2.2 Second Object : Cow

The second object which used for trial on PCSS shadow rendering is cow, which have less minute details as compare to lamp object. If this two results are compared, the shadow of cow is more accurate. This can be determined on the following parameters: depth size, filter count, light size. From the outcome, an estimation of how close the result is from real world shadow is made. If results are properly observed for the shadow for object two its clearly understandable that this shadow is more accurate as the outline of the object is not that harsh. The resultant outcome soft edged shadow which is making it more close to real world as its a soft shadow. In Figure 28 the light source is moved apart from the object as compared with the other other images so a small shadow can be seen there. On the other side if Figure 29 is considered the light source is closer to the object so a bigger shadow can be seen in that particular case. The object color, platform color can be changed using some of the variables.

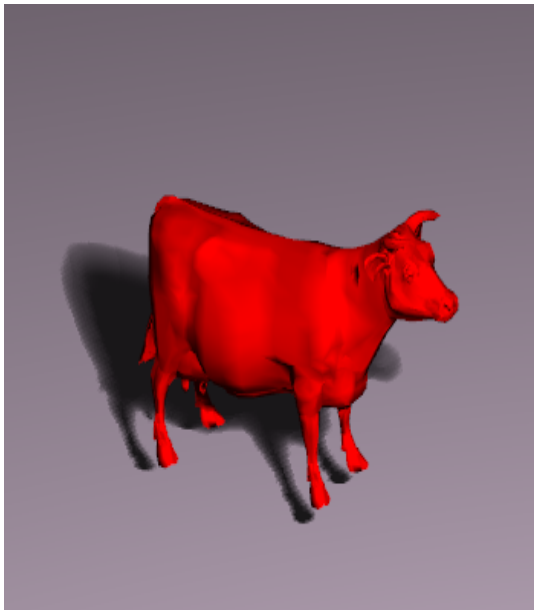

Figure 25: PCSS Lamp shadow result 1

Figure 26: PCSS Lamp shadow result 2

Figure 27: PCSS Lamp shadow result 3

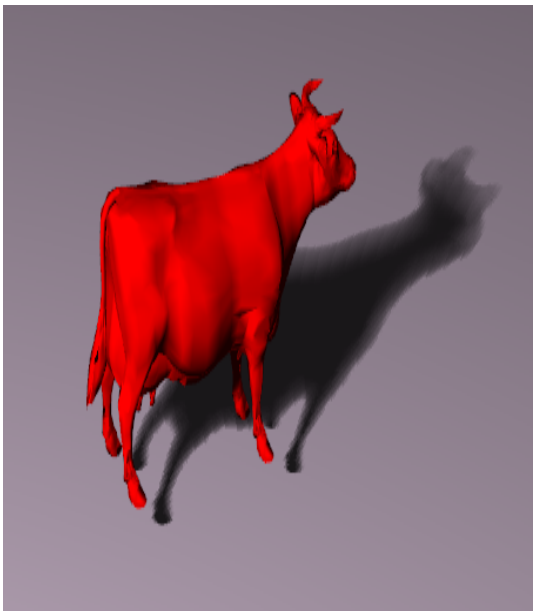

Figure 28: PCSS Lamp shadow result 4

5.3 Performance

One of the efficient way of analysing performance of any graphics project is FPS. FPS is frames per second. The measurement of successive frames rendered on the screen can be shown by frames per second. The excessive value of FPS leads to best graphics work. If the FPS value is not that large it indicates bad graphics experience. Though shadow can make

a scene more realistic but with using shadows there are higher chances of FPS will decrease which not a good sign. To execute this entire application NVIDIA GEFORCE GTX graphics processor is used. The PCSS method roughly runs at 82 fps with the NVDIA GPU. The sample count used for this implementation is 8 an 16. With 16 it gives precise shadows as results but it will increase computational cost. Light size and the quality which desired plays a important role on deciding the sample count.

6 Conclusion

With the results of the PCSS shadow rendering technique it can be concluded that soft shadows which are closer to real world shadows can be generated. With the process of filtrating texture map value, texture maps required less storage space. This values will be filtered above some area from texture map. The depth of the surface which is rendered gets compared to the texture value to estimate if that part is in shadow or not. The outcome of this comparison process can return binary values because of which aliased edges in shadows are possible. This issue can be handled by PCSS as the sequence of comparison is altered by this method. First, for complete area a depth map is calculated and this will have z values. This process will change the depth map to a binary image and this image will go through the filtration process to estimate areas in shadow. This way there are high chances of getting antialiased edges in shadows. Generally, if shadows are suppose to render it will cost 40% to 50% additional computational cost as compare to not having shadows. As the number of light sources increases cost will be more but with the result of PCSS it is proved that soft shadows can be rendered efficiently even if there is more than one light source in scene. Storing depth maps significantly takes more space but the storage issue can be handled in a more efficient by PCSS. This method is not capable of rendering the shadows for motion blur images. Overall, if we neglect the high cost for computations required by PCSS it can offer realistic shadows for the scene.

References

[1] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François X. Sillion. A survey of Real-Time Soft Shadows Algorithms. *Computer Graphics Forum*, 22(4):753–774, December 2003. URL `https://hal.inria.fr/inria-00281388`.

[2] Pradeep Sen, Mike Cammarano, and Pat Hanrahan. Shadow silhouette maps. 22(3): 521–526, jul 2003. ISSN 0730-0301.

[3] Lance Williams. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, 12(3):270–274, aug 1978.

[4] Randima Fernando. Percentage-closer soft shadows. page 35–es, 2005. URL `https://doi.org/10.1145/1187112.1187153`.

[5] Nan Liu and Ming-Yong Pang. Shadow mapping algorithms: A complete survey. pages 1–5, 2009.

[6] Franklin C. Crow. Shadow algorithms for computer graphics. 11(2):242–248, jul 1977. ISSN 0097-8930. URL `https://doi.org/10.1145/965141.563901`.

[7] Lu Liu and Shuangjiu Xiao. Real-time soft shadows for large-scale virtual environments. pages 5464–5467, 2011. doi: 10.1109/ICMT.2011.6002281.

[8] Maneesh Agrawala, Ravi Ramamoorthi, Alan Heirich, and Laurent Moll. Efficient image-based methods for rendering soft shadows. page 375–384, 2000.

[9] William Donnelly and Andrew Lauritzen. Variance shadow maps. pages 161–165, 2006.

[10] Ian Mallett, Larry Seiler, and Cem Yuksel. Patch textures: Hardware support for mesh colors. *IEEE Transactions on Visualization and Computer Graphics*, 28(7):2710–2721, 2022.

[11] Eric Chan and Frédo Durand. Rendering fake soft shadows with smoothies. page 208–218, 2003.

[12] Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. Large steps in inverse rendering of geometry. *ACM Trans. Graph.*, 40(6), dec 2021. ISSN 0730-0301. URL `https://doi.org/10.1145/3478513.3480501`.

[13] T.A. Davis and E.W. Davis. Exploiting frame coherence with the temporal depth buffer in a distributed computing environment. pages 29–116, 1999.

[14] Ana Mihut, Richard Davison, Gary Ushaw, and Graham Morgan. Lighting and shadow techniques for realistic 3d synthetic object compositing in images. page 70–79, 2018. URL https://doi.org/10.1145/3193025.3193060.

[15] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 283–291, New York, NY, USA, 1987. Association for Computing Machinery. ISBN 0897912276.

[16] Fan Zhang, Hanqiu Sun, Leilei Xu, and Lee Kit Lun. Parallel-split shadow maps for large-scale virtual environments. page 311–318, 2006. URL https://doi.org/10.1145/1128923.1128975.

[17] Fan Zhang, Hanqiu Sun, and Oskari Nyman. Parallel-split shadow maps on programmable gpus. *GPU gems*, 3:203–237, 2007.

[18] Edwin Earl Catmull. *A subdivision algorithm for computer display of curved surfaces*. The University of Utah, 1974.

[19] Tran Anh Tuan, Min Gyu Song, and Jin Young Kim. A robust shadow and light region detection using within-class variance in face images. pages 455–460, 2011. doi: 10.1109/ISSPIT.2011.6151605.

[20] Bingshu Wang, C.L. Philip Chen, Yuyuan Li, and Yong Zhao. Hard shadows removal using an approximate illumination invariant. pages 1628–1632, 2018. doi: 10.1109/ICASSP.2018.8461695.

[21] Michael D. McCool. Shadow volume reconstruction from depth maps. 19(1):1–26, jan 2000. ISSN 0730-0301.

[22] Fabio Pellacini Michael Bunnell. Chapter 8. summed-area variance shadow maps. *GPU gems*, 2007. URL https://hal.inria.fr/inria-00281388.

[23] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1): 51–72, jan 1986. ISSN 0730-0301. URL https://doi.org/10.1145/7529.8927.

[24] Fabio Pellacini Michael Bunnell. Chapter 11. shadow map antialiasing. *GPU gems*, 2007. URL https://developer.nvidia.com/gpugems/gpugems/part-ii-lighting-and-shadows.

[25] About opengl - percentage closer soft shadow rendering. URL https://github.com/RaphaelK12/PCSS-With-OpenGL.

[26] Gregory S. Johnson, Warren A. Hunt, Allen Hux, William R. Mark, Christopher A. Burns, and Stephen Junkins. Soft irregular shadow mapping: Fast, high-quality, and robust soft shadows. page 57–66, 2009. URL https://doi.org/10.1145/1507149.1507159.

[27] Ankit Mohan, Jack Tumblin, and Prasun Choudhury. Editing soft shadows in a digital photograph. *IEEE Computer Graphics and Applications*, 27(2):23–31, 2007. doi: 10.1109/MCG.2007.30.

[28] Marc Stamminger and George Drettakis. Perspective shadow maps. page 557–562, 2002. doi: 10.1145/566570.566616. URL https://doi.org/10.1145/566570.566616.

[29] J. Konrad and P. Agniel. Subsampling models and anti-alias filters for 3-d automultiscopic displays. *IEEE Transactions on Image Processing*, 15(1):128–140, 2006. doi: 10.1109/TIP.2005.860339.

[30] Jaakko Lehtinen, Timo Aila, Jiawen Chen, Samuli Laine, and Frédo Durand. Temporal light field reconstruction for rendering distribution effects. 2011. URL https://doi.org/10.1145/1964921.1964950.

[31] Per H. Christensen, Julian Fong, David M. Laur, and Dana Batali. Ray tracing for the movie 'cars'. pages 1–6, 2006. doi: 10.1109/RT.2006.280208.