Universidade do Minho
Escola de Engenharia

João Abrantes de Oliveira

Analysis and Proposal of a Shadow Mapping Remix Approach

Outubro de 2009

Universidade do Minho
Escola de Engenharia

João Abrantes de Oliveira

Analysis and Proposal of a Shadow Mapping Remix Approach

Mestrado em Informática

Trabalho efectuado sob a orientação do
Professor Doutor António Ramires Fernandes

Outubro de 2009

DECLARAÇÃO

Nome ____João Abrantes de Oliveira____

Endereço electrónico: _joao.abrantes.80@gmail.com_ Telefone: _966071667_ / ______________

Número do Bilhete de Identidade: __11807813_________

Título dissertação ?/tese ?

_Analysis Analysis and Proposal of a Shadow Mapping Remix Approach_______

____

____

Orientador(es):

___António Ramires Fernandes____

____ Ano de conclusão: __2009________

Designação do Mestrado ou do Ramo de Conhecimento do Doutoramento:

_____Mestrado em Informática____

Nos exemplares das teses de doutoramento ou de mestrado ou de outros trabalhos entregues para prestação de provas públicas nas universidades ou outros estabelecimentos de ensino, e dos quais é obrigatoriamente enviado um exemplar para depósito legal na Biblioteca Nacional e, pelo menos outro para a biblioteca da universidade respectiva, deve constar uma das seguintes declarações:

1. É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA TESE/TRABALHO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

2. É AUTORIZADA A REPRODUÇÃO PARCIAL DESTA TESE/TRABALHO (indicar, caso tal seja necessário, nº máximo de páginas, ilustrações, gráficos, etc.), APENAS PARA EFEITOS DE INVESTIGAÇÃO, , MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

3. DE ACORDO COM A LEGISLAÇÃO EM VIGOR, NÃO É PERMITIDA A REPRODUÇÃO DE QUALQUER PARTE DESTA TESE/TRABALHO

Universidade do Minho, ___/___/______

Assinatura: ____

Analysis and Proposal of a Shadow Mapping Remix Approach

Abstract

Virtual Environments creation is becoming increasingly demanding, trying to portrait all the aspects of reality with great detail. Fast hardware innovation turns possible the implementation of some real-time techniques that were not feasible only a few years ago using a single, ordinary computer.

Shadow effects are one of the contributions for the realism improvement in virtual environments. One of the most successful high-quality real-time shadow effect techniques used is shadow mapping. This technique is efficient and simple to implement, but its standard implementation has its drawbacks and limitations, so it is necessary to improve this technique according to the type of scene we want to show.

This work starts by analyzing the standard implementation of the shadow mapping algorithm, pointing out its main issues. Then, the most relevant contributions to solve/alleviate those issues are presented. For each contribution it is shown which issues are tackled, how they are tackled, and, where appropriate, the new issues that have arisen from the contribution itself.

Based on this analysis, the most relevant sections of each contribution are highlighted and a new algorithm, built with pieces from the presented contributions is proposed. This new algorithm, named Shadow Mapping Remix Approach, attempts to combine the strengths of each contribution without incurring into new issues, providing a more complete solution to the shadow mapping problem.

The results show that, although as expected the proposed approach has a lower performance than each individual contribution, its results are far superior to any of the previous contributions.

Análise e proposta de uma abordagem recombinada de algoritmos de Shadow Mapping

Resumo

A criação de ambientes virtuais está-se a tornar cada vez mais exigente, tentando assemelhar-se da realidade com grande detalhe. A rápida inovação do hardware torna agora possível a implementação de certas técnicas em tempo real.Os efeitos de sombras são uma das contribuições para a melhoria no realismo em ambientes virtuais. Uma das mais bem sucedidas técnicas de geração de sombras em tempo real de alta qualidade é o shadow mapping. Para além de ser eficiente, é também simples de implementar. Porém a sua implementação básica apresenta muitos inconvenientes e limitações.

O nosso trabalho começa por analisar a implementação básica do algoritmo de shadow mapping, apontando os seus principais problemas. Seguidamente, apresentamos as contribuições mais relevantes que resolvem/atenuam os problemas apresentados. Para cada contribuição, são descritos os problemas abordados e de que forma são resolvidos, sendo também apresentados, em alguns casos, problemas que derivam dessa mesma contribuição. Tendo em conta a análise efectuada, serão realçadas as secções mais importantes de cada contribuição e um novo algoritmo, baseado nessas mesmas secções, será proposto.

Este novo algoritmo, denominado por Shadow Mapping Remix Approach, combina assim os pontos mais fortes de cada um dos algoritmos propostos tentando não introduzir novos problemas, providenciando uma solução mais completa para a problemática do shadow mapping.

Os resultados mostram que, apesar de ser esperado um decréscimo na performance relativamente ao algoritmo apresentado, os nossos resultados visuais são muito superiores a qualquer uma das contribuições apresentadas.

Contents

1. Introduction 1

   1.1. Motivation 6

   1.2. Research Methodologies 6

   1.3. Overview 8

2. Fundamentals 11

   2.1. Basic Algorithm 11

   2.2. Shadow Mapping Issues 12

      2.2.1. Perspective Aliasing 13

      2.2.2. Projection Aliasing 15

      2.2.3. Texture Resolution Constraints 16

      2.2.4. Self-Shadowing 17

      2.2.5. Adaptive Light Frustum 22

      2.2.6. Hard Shadows Binary Status 23

   2.3. Percentage Closer Filtering Algorithm 24

   2.4. Conclusion 26

3. State of the Art 27

   3.1. Perspective Shadow Mapping 30

      3.1.1. Implementation – General Case 32

      3.1.2. Special Cases 33

   3.2. Trapezoidal Shadow Mapping 37

3.2.1. Reducing the resolution problem 38

3.2.2. Implementation 41

3.2.3. Solving the Polygon Offset Problem 49

3.2.4. Special Cases 51

3.3. Parallel-Split Shadow Mapping 51

3.3.1. Split Scheme Discussion 53

3.3.2. Light frustum split 55

3.3.3. Implementation 57

3.3.4. Discussion 57

3.4. Variance Shadow Mapping 59

3.4.1. Implementation 60

3.4.2. Discussion 61

3.5. Conclusion 63

4. Analysis and Proposal of a Shadow Mapping Remix Approach 69

4.1. Analysis - Contributors' Strongest Features 69

4.2. Remix Approach Proposal 71

4.2.1. The Detailed Algorithm 72

4.2.2. Artifacts Analysis 74

4.3. Discussion 75

4.3.1. Visualizing the implementation 76

4.3.2. Performance estimation 82

4.4. Conclusion 83

5. Results 85
  5.1. The Setup 85
  5.2. Visual Results 86
    TreeLine 86
    SparseCity 93
5.3. Performance Results 100
    Application Speed 100
  5.4. Conclusion 101
6. Conclusions 103
  6.1. Future Work 104
Appendix A 105
  Trapezoid side lines computation 105
References 107
Web References 109
Models and Utilities 110

List of Abbreviations

SSM – Standard Shadow Mapping

PCF – Percentage Closer Filtering

PSM – Perspective Shadow Mapping

LiSPSM – Light Space Perspective Shadow Mapping

TSM – Trapezoidal Shadow Mapping

VSM – Variance Shadow Mapping

CSM – Convolution Shadow Mapping

ESM – Exponential Shadow Mapping

SMRA – Shadow Mapping Remix Approach

List of Figures

Figure 1: Hard Shadow (left) vs. Soft Shadow (right).. 2

Figure 2: A 2D diagram showing shadow volume geometry.............................. 3

Figure 3: Shadow Mapping depth comparison: *z* is the occluder's distance to the light. *d* is the distance of the surface to the light. *p* will be shadowed, since $z < d$.. 4

Figure 4: Aliased shadow due to insufficient shadow map resolution 4

Figure 5: Hybrid Shadow Rendering Algorithm interaction. 5

Figure 6: Shadow created with a shadow map. Bottom-left square shows a shadow map created from the light point of view...................................... 12

Figure 7: Aliased shadow map, mainly due to insufficient resolution. The red square, representing a texel in the shadow map, isn't detailed enough to represent the shadow of the woman's shoulder. 13

Figure 8: While distant shadows will be detailed, nearby shadows will suffer of insufficient resolution. This problem is caused by perspective aliasing..... 14

Figure 9: The result of projection aliasing is insufficient shadow map resolution for shadows in surfaces close to parallel to light-view direction................ 15

Figure 10: The projection aliasing artifacts (dark stripes at building walls) in the eye view (left image) are caused by too few samples of the cubes sides as seen from the light view (right image).. 16

Figure 11: A single 4096x4096 shadow map has not enough resolution (right) to produce detail for a large-scale virtual environment with 8 square kms (left). 17

Figure 12: Sampling differences between light point-of-view and camera point-of-view can lead to incorrect self-shadowing.. 19

Figure 13: Depth quantization due to shadow map finite precision also leads to incorrect self-shadowing 20

Figure 14: The moiré patterns are caused by sampling conflicts and depth quantization problems 21

Figure 15: Continuity problem due to adaptation of the light frustum to the scene as viewed from the camera: The top picture shows a scene containing a palm tree; as the camera moves, a big building enters the scene, introducing drastic changes to the shadow map resolution. 23

Figure 16: Hard shadow (left) vs. filtered shadow map (right)......................... 24

Figure 17: Standard Shadow Maps process (top) vs. Shadow Maps with Percentage Closer Filtering process (bottom). 25

Figure 18: Standard shadow mapping (Top Left) and the respective shadow map (Top Right) generated as if the scene was rendered from the light's point of view. Perspective Shadow mapping (Bottom Left) reduces aliasing by applying a perspective transformation while generating the shadow map (Bottom Right)... 31

Figure 19: World View (left) is transformed to Post-perspective space (right) in order to perform perspective shadow mapping. The shadow map pixels, once projected on the scene surface will be evenly distributed, according to the camera-view.. 32

Figure 20: Mapping of lights in world space to post perspective space. Directional lights in world space become point lights in post perspective space (Top). Also, lights from behind the camera become inverted. Point lights in world space remain point lights in post perspective space, except for boundary cases (Bottom). 34

Figure 21: Computation of the scene's (world-space) region of interest for shadow map generation. 36

Figure 22: Case where the region of interest for the shadow map generation needs to be extended in order to avoid scene disfiguration introduced by the post-perspective transformation. 37

Figure 23: The most similar to shape to a view-frustum as seen from the top is noticeably a trapezoid. Using a square for the approximation, some non-interest areas of the scene would be included in the transformation......... 39

Figure 24: Trapezoidal approximation vs. Bounding-box approximation.......... 40

Figure 25: Side lines of the trapezoid (left) are calculated according to the 80% rule (right), so that the stretching of trapezoid does not introduce sampling artifacts (center) 42

Figure 26: Trapezoidal transformation: mapping trapezoidal corners to square edge corners 43

Figure 27: TSM application of T1 44

Figure 28: TSM application of R 44

Figure 29: TSM application of T2 45

Figure 30: TSM application of H 45

Figure 31: TSM application of S1 46

Figure 32: TSM application of N 46

Figure 33: TSM application of T3 47

Figure 34: TSM application of S2 47

Figure 35: TSM application of NT 48

Figure 36: Scene shadowed with TSM using standard polygon offset problem (large bias). 49

Figure 37: Scene shadowed with TSM after polygon offset problem tackling. . 50

Figure 38: The view frustum is split into three parts, each of them will be rendered to a different shadow map with the same resolution 52

Figure 39: Although PSSM assumes directional light sources, point light sources can be converted once in light's post-perspective space............. 52

Figure 40: Split planes distribution along the view frustum z axis 53

Figure 41: Split schemes presented by Parallel-Split Shadow Maps 55

Figure 42: Light’s frustum split (W*i*) computation according to the respective view split frustum V*i* and its bounding box B*i*... 56

Figure 43: Split discontinuity problem.. 58

Figure 44: Scene rendered with a 512 shadow map: (left) PCF 4x4 vs. (right) VSM.. 62

Figure 45: Scene shadow rendered with VSM suffers of light bleeding (left). The same scene shadow is rendered using Standard Shadow Mapping (right). Note that both images have their contrast levels increased so that artifacts are more perceivable. 62

Figure 46: A scene with projection aliasing, right before the projection aliasing removal technique is applied (top) and after the technique is applied (bottom). 75

Figure 47: Scene rendered without shadows 76

Figure 48: Scene rendered with SSM 512x512 77

Figure 49: Trapezoidal approximation of the view frustum, as seen from the light 77

Figure 50: Shadow Map generated using SSM (left) and TSM (right).............. 78

Figure 51: Scene rendered with TSM 512x512 79

Figure 52: PSSM scene splits, as seen from the camera 79

Figure 53: Scene rendered with PSSM 3x512x512 80

Figure 54: Scene rendered with PSSM and TSM 3x512x512 80

Figure 55: Scene rendered using VSM 512x512 81

Figure 56: Scene rendered using SMRA 3x512x512 81

Figure 57: TreeLine rendered with SMRA 86

Figure 58: TreeLine scene being rendered with SSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom) 87

Figure 59: TreeLine scene being rendered with TSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom) 89

Figure 60: TreeLine scene being rendered with PSSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom) 90

Figure 61: TreeLine scene being rendered with VSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom) 91

Figure 62: TreeLine scene being rendered with SMRA using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom) 92

Figure 63: SparseCity rendered with SMRA 93

Figure 64: SparseCity scene being rendered with SSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom) 94

Figure 65: SparseCity scene being rendered with TSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom) 96

Figure 66: SparseCity scene being rendered with PSSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom) 97

Figure 67: SparseCity scene being rendered with VSSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom) 98

Figure 68: SparseCity scene being rendered with SMRA using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom) 99

Figure 69: A 1D homogenous perspective projection problem to compute q ..105

List of Tables

Table 1: Problems present (red) vs. problems tackled (green) for each of the presented techniques. Note: (N.A.) means the current algorithm does not suffer from the referred artifact 66

Table 2: Rendering speed (in frames per second) for the most relevant algorithms. 100

1. Introduction

Although graphics hardware do not provide shadows by default, shadows are a precious element for 3D perception of a scene – they provide visual cues that help to perceive the geometric relationship between objects, clarifying objects' position, size and geometry. Even though we don't stop to think about what shadows really mean for our comprehension of the surroundings, we immediately notice that a 3D scene without shadows lacks life and realism. This is why calculation of shadows is a very important component when representing virtual environments.

At first, the main Graphics' APIs (OpenGL, Direct3D) did not support mechanisms for shadows generation, so their implementation was made 100% by software.

With the increasingly growth of functionality of GPUs, these APIs enabled the possibility of real-time shadowing. The hardware performance improvement has been encouraging the usage of these mechanisms, so the research around this subject has grown.

Naively, a shadow is a binary status: a point is either «in shadow» or not – there is no penumbra. This corresponds to hard shadows.

Hard Shadows suffer of a problem which is the binary status described above. This binary status produces very noticeable shadow edges, being easily noticed. In light to this problem arises a technique called soft shadows: this technique applies a soft transition between the shadowed and unshadowed areas, creating a smooth transition between lit and unlit surfaces (see Figure 1).

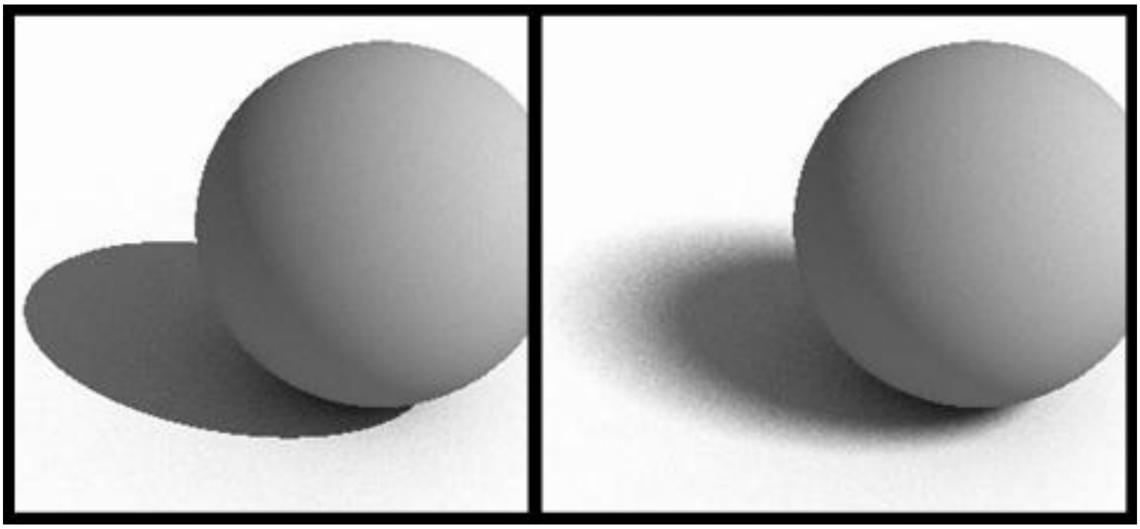

Figure 1: Hard Shadow (left) vs. Soft Shadow (right)

Although there are some filtering techniques for applying to the hard-shadows, these rely on a fixed-size penumbra basis, while soft shadows enable a variable-size penumbra, which may be dependent on light and occluder distance to the projected shadow surface. For a more thorough discussion about this topic, consult [SRTSSA03] and [ASSMT08].

As to Hard-Shadows the most popular real-time generation techniques that have emerged over time are Shadow Volumes and Shadow Maps.

Shadow volumes approach [SACG77] uses the geometry to produce the exact volume of the shadow, rendering boundaries without aliasing. This technique works in object space, being required a full connectivity information of all polygons in order to produce the silhouette of each shadow caster (see Figure 2). Therefore, the scene complexity directly influences the application performance.

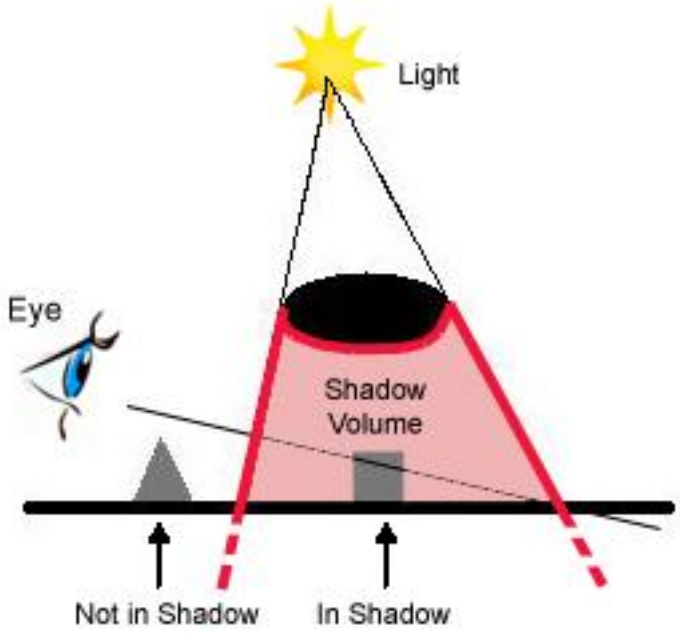

Figure 2: A 2D diagram showing shadow volume geometry

Shadow Maps approach [CCSCS78] distinguishes itself amongst other algorithms thanks to its implementation simplicity. This algorithm works partially in image space, so the performance hit is less dependent on scene complexity and the possibility of being taken from arbitrary locations with no performance hits turns this algorithm very popular. Shadow mapping is basically a two-pass algorithm. In the first pass, the scene is rendered from the light's point of view, the depth buffer stores the distance of the scene to the light (creating the so called shadow map). In the second pass, the scene is rendered from the eye's point of view, and each pixel is again transformed into the light's view space, performing a comparison between its distance to the light with the value stored in the shadow map: If the distance value of the pixel is larger than the depth value stored in the shadow map this pixel is shadowed, otherwise, it is not shadowed (see Figure 3).

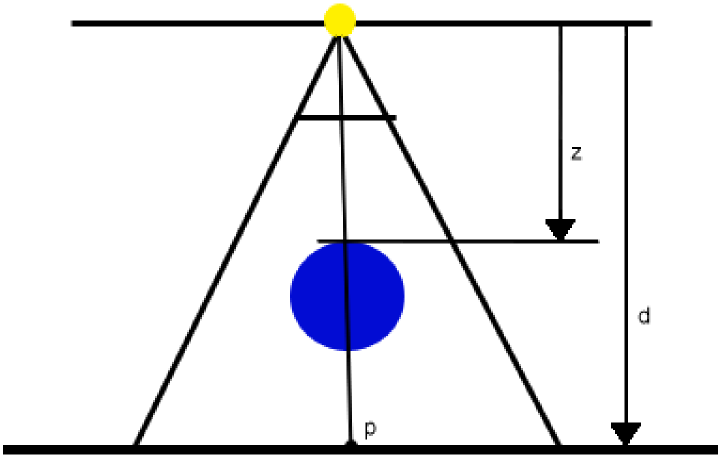

Figure 3: Shadow Mapping depth comparison: *z* is the occluder's distance to the light. *d* is the distance of the surface to the light. *p* will be shadowed, since *z* < *d*.

Since the algorithm relies on two different perspectives it suffers from aliasing problems: the shadow map resolution may be insufficient for the scene being rendered and stretching the shadow map to the needed space will introduce pixelated shadows (see Figure 4).

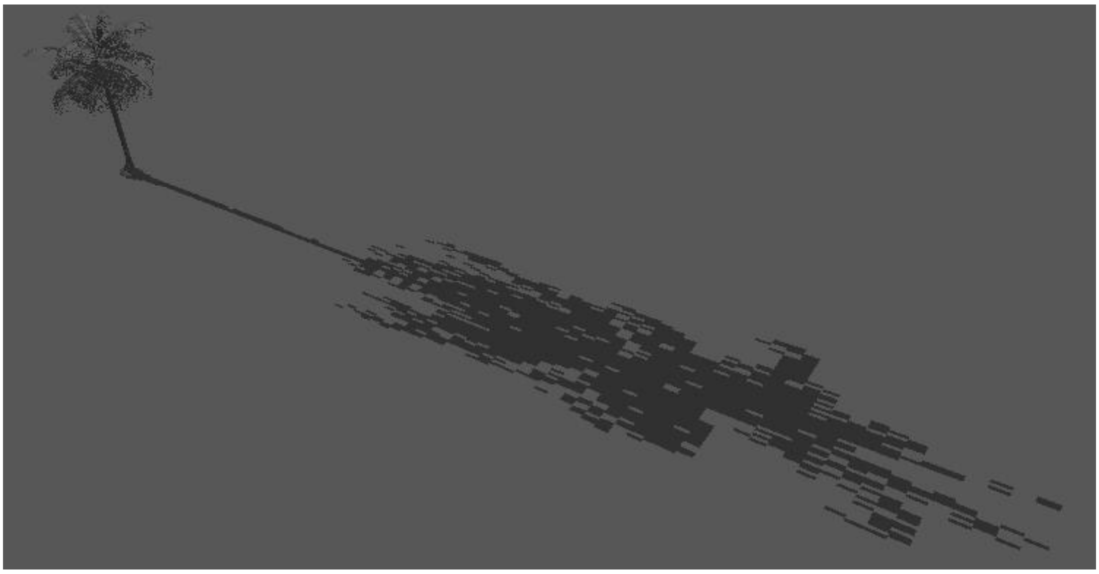

Figure 4: Aliased shadow due to insufficient shadow map resolution

There is also a hybrid algorithm for rendering hard shadows [EHSRA04], combining the strengths of shadow maps for identifying the pixels in the image that lie near shadow discontinuities and shadow volumes for ensuring accurate shadow edges at these discontinuities. This approach simultaneously avoids the edge aliasing artifacts of standard shadow maps and avoids the high fill rate consumption of standard shadow volumes.

Figure 5 shows this hybrid algorithm at work. Red and blue pixels are non-silhouette pixels while black and green pixels are shadow silhouette pixels. Hybrid Shadow Rendering Algorithm uses Shadow Maps for determining which pixels are in shadow (red) and which ones are not (blue) and uses Shadow Volumes for determining which ones are in shadow (black) and which ones are not (green).

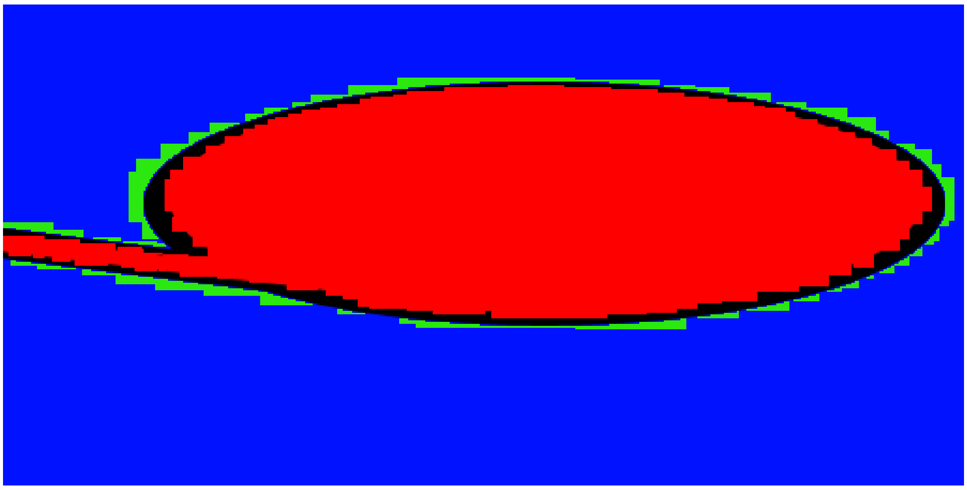

Figure 5: Hybrid Shadow Rendering Algorithm interaction.

This thesis focuses on Shadow Maps, presenting the general algorithm and some of the most relevant proposals that try solve its artifacts, improving the quality of the generated shadows. Based on these algorithms, a new proposal will be presented, where some components of the presented algorithms will be used and recombined in order to achieve a better quality of the projected shadows at the best possible performance. As to soft shadows, its implementation is complementary to the hard shadows generation and is out of the scope of this investigation.

1.1. Motivation

As stated before, although Shadow Maps are a very straight-forward technique, there are issues that result directly from the algorithm's design. Summarily, we can point out the following: perspective aliasing, projection aliasing, surface self-shadowing, texture resolution constraints, shadows discontinuity, binary shadow status representation.

Over time, many algorithms have been proposed, each one of them presenting different (and sometimes complementary) solutions to some of these problems. In a general way, we can identify the area of action for each one of these algorithms as one of the following:

- Perspective Correction;
- Texture Filtering;
- Scene Partitioning.

Although there are notorious improvements in the presented algorithms, there are still limitations associated to each one of them: they focus on a specific shadow mapping artifact, not covering the whole problematic of shadow maps limitations.

Our work is to identify each of the shadow maps algorithm limitations, present algorithms based on shadow maps that solve some of these limitations (comparing different solutions for the same problem) and finally propose a solution based on parts of each of the best complementary approaches. This remix approach will be able to render a large scene in real-time, providing high resolution shadows for all perceivable parts of the scene, as viewed from the camera.

1.2. Research Methodologies

This research will start with a bibliographic research in order to evaluate the state-of-the-art in Shadow Mapping techniques. This investigation will be complemented with a study of some terms and techniques required for the implementation of shadow maps, as matrix transformations, blending, texture mapping, texture filtering, frame buffers, amongst others.

Based on this investigation and we will implement an OpenGL based application capable of comparing all the shadow mapping techniques presented in this thesis, including the one we propose – this application will be able to provide both performance and visual results in real-time for the same scene, by swapping between algorithms, shadow maps resolutions, cameras, light position, and logging the results.

Given this, we will start with a first version, where a scene will be displayed with standard shadow mapping and where all the theoretically identified artifacts are present.

In a second step, we will evaluate which algorithms are more suitable for correcting each class of the artifacts and implement them, comparing the results with the standard algorithm (also called uniform shadow mapping).

After implementing all the relevant algorithms and comparing them to the standard shadow mapping, we will compare the algorithms between themselves, in an attempt to fully grasp, together with the theoretical basis, what are their strengths and limitations.

In a final phase, we will implement our algorithm based on the strong aspects of all the studied algorithms and compare it with the previously implemented approaches.

According to this, the research methodology that better seems to suit to our plans is the one designated as Action-Research. This methodology belongs to a family of methodologies action (or changing) and research (or understanding) at the same time.

Initially it will be made a cyclic process (or in spiral) that alternates between action and critical analysis, later it will happen an understanding of the methods and data, being continuously improved.

This is an emerging process that acquires its shape with the knowledge growth and is also an iterative process that converges for a better perception of what actually happens.

The Action-Research is carried out as follows:

- Problem identification;
- Problem analysis – find ways to solve it;
- Implementation of improvement proposals;
- Evaluation of the implementation of improvement proposals;
- Modification of the learning methods;
- Re-definition of the Action-Research methods.

Thus, with this thesis, we are looking for an algorithm that answers the following challenges:

- Solve severe aliasing present in Standard Shadow Mapping approach;
- Filter shadow maps in order to smooth the shadow silhouettes;
- Achieve the above results with the maximum performance possible, having in mind that shadow mapping is used for real-time rendering.

1.3. Overview

The remaining of this thesis is organized as follows. Chapter 2 presents the fundamentals for understanding the following chapters. It will be given a description of the Shadow Mapping algorithm, stating all its problems/limitations. It will also be given an introduction to shadow map filtering, referring to Percentage-close Filtering.

Chapter 3 presents the state of the art relatively to Shadow Mapping algorithms. For each algorithm, we explain the approach, discuss their contribution to standard shadow mapping, identifying the class of artifacts it reduces, and present its limitations.

Chapter 4 presents our algorithm’s implementation: “Shadow Maps Remix Approach”. We will explain how the pieces of some of the algorithms present in Chapter 3 are reconnected, giving birth to a new, more complete algorithm, capable of presenting improved shadow quality for large-scale environments.

Chapter 5 shows the results of each implementation, comparing the results with the other algorithms, including Standard Shadow Maps as worst case scenario and Shadow Maps Remix Approach as the proof of concept.

Chapter 6 is the conclusion, where we summarize all our efforts and present future research directions regarding this investigation.

2. Fundamentals

In this chapter we will describe the standard shadow maps algorithm, presenting its artifacts. It will also be given a brief description of the percentage-close filtering technique, which in addition to the current hardware support for this feature, is the basis for all shadow mapping filtering techniques presented in this thesis.

2.1. Basic Algorithm

Shadow mapping (SSM) was introduced by Lance Williams [CCSCS78] in 1978 and has been extensively used since then, in both offline and real-time rendering.

Shadow mapping is a two-pass algorithm: in a first pass the scene is rendered from the light point of view, being gathered the depth values in the so called shadow map (see Figure 6). These depth values represent the distance from each visible point (according to the light point of view) to the light source;

Afterwards, in the second pass the scene is rendered from the camera point of view, each pixel being transformed into light space and its distance to the light compared to the value stored in the shadow map. The comparison is made for shadow determination: If the stored value is smaller than the distance to the light, the pixel is shadowed; otherwise, the pixel is not in shadow.

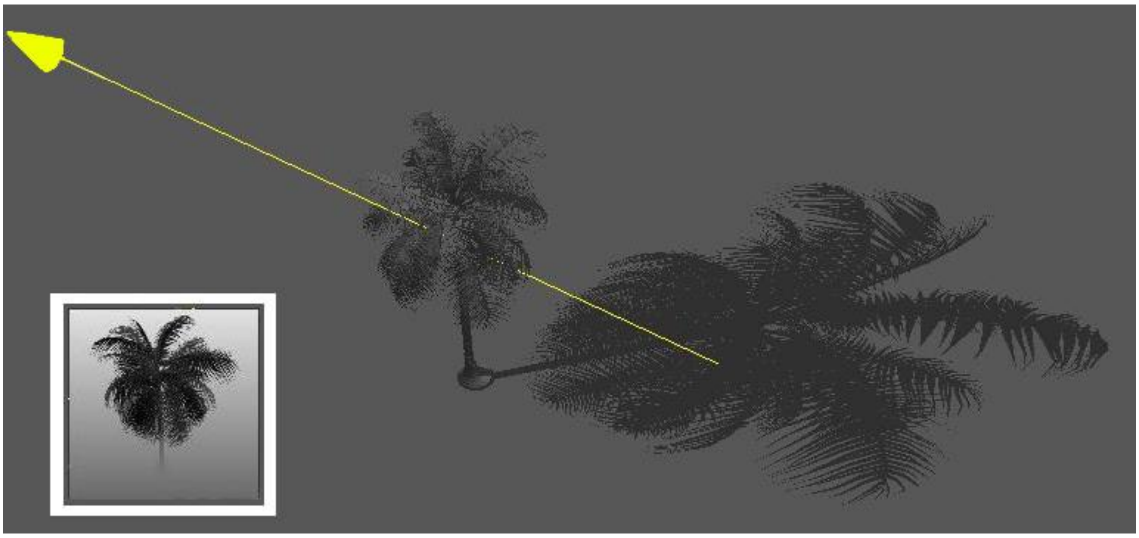

Figure 6: Shadow created with a shadow map. Bottom-left square shows a shadow map created from the light point of view.

2.2. Shadow Mapping Issues

Shadow mapping suffers of many problems, mostly due to aliasing issues relative to the nature of the algorithm itself. These aliasing errors are mainly caused by the under-sampling (insufficient shadow map resolution). Such aliasing errors can be further classified into perspective aliasing and projection aliasing [PSM02].

In Figure 7 the aliasing of shadow maps can be seen. It is noticeable the lack of resolution as square blocks refer to one texel of the shadow map, magnified in eye-space. It is also clear from the image that the resolution problem is more noticeable near the point-of-view of the camera because the shadow map texels in far-away from the camera, when viewed from the camera's point of view, are small enough to provide sufficient shadow resolution for far away objects.

Below, we present a list of issues related to shadow mapping limitations. Some are applicable to every virtual environment, while others are noticed especially in large-scale or outdoor environments:

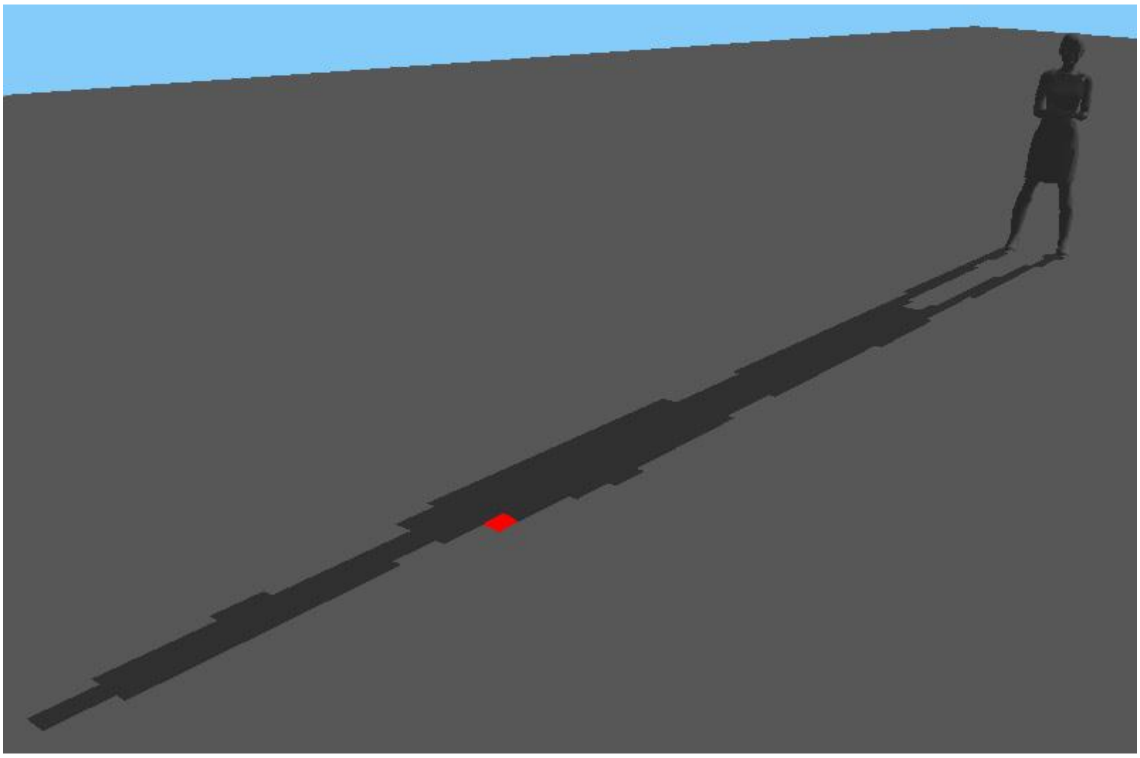

Figure 7: Aliased shadow map, mainly due to insufficient resolution. The red square, representing a texel in the shadow map, isn't detailed enough to represent the shadow of the woman's shoulder.

2.2.1. Perspective Aliasing

Perspective aliasing is a common problem with standard shadow maps considering a perspective eye view. A perspective view shows nearby objects larger than distant objects, where distance is relative to the camera. The light space, in which the standard shadow map is computed, is usually taken from a different point of view - it does not incorporate this information.

Considering a directional light, an object is stored with a fixed resolution in the shadow map, regardless of the distance to the eye. The outcome is a shadow resolution that does not match the resolution of the perspective camera; hence it is clearly insufficient for nearby objects (relative to the camera). For distant objects in eye space, the problem is not so noticeable. In Figure 8, the shadow map is represented by a grid (in the top); the space between the grid strips represents a texel. We assume the light direction as straight top-down. The camera is observing this scene from the side. The shadow resolution applied to the shadow map created in light-space is not equally distributed in eye-space, which leads to potentially over-sampling (wastage of resolution) for far away shadows and under-sampling (insufficient resolution) for the shadows near the camera.

As to the different light configurations, we can have point lights and directional lights. For directional lights, this mismatch has a maximum if the light direction is perpendicular to the view direction, while the minimum mismatch occurs if the light direction and the camera are facing in the same direction (are parallel to each other). For point lights the problem becomes more pronounced as the light's direction moves away from the camera viewing direction, having its most pronounced mismatch when the light and camera are facing each other. This is known as the dueling frusta problem.

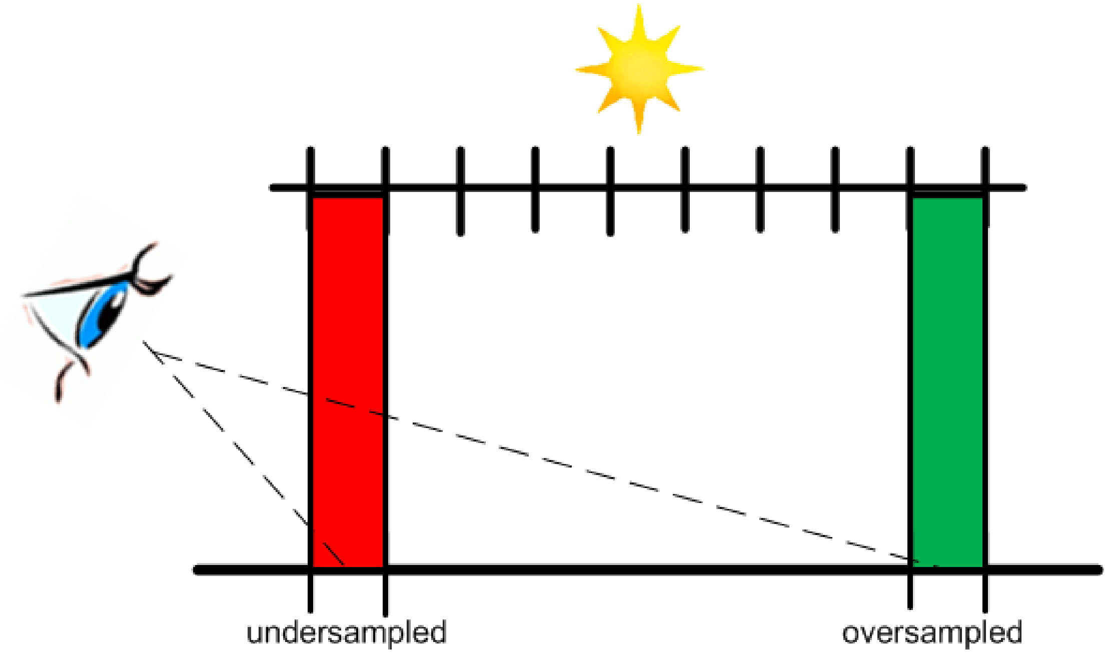

Figure 8: While distant shadows will be detailed, nearby shadows will suffer of insufficient resolution. This problem is caused by perspective aliasing.

2.2.2. Projection Aliasing

When a surface is roughly parallel to the light direction, its depth information will be sampled sparsely in the shadow map, resulting in the so called projection aliasing error. In Figure 9, the shadow map is represented by a grid (in the top), being a texel represented by the space between the grid strips; we also assume the light direction as straight top-down. As the object's side is close to parallel to the light's view direction, a simple shadow map texel will gather the depth information for almost the whole side surface of the object. Thus, at eye-view, we will see incorrect shadows due to the projection of too many pixels to a single shadow texel.

Another issue regarding to projection aliasing is related to its large temporal discontinuity on camera movement. This will lead to severe shadow flickering while moving around in a scene, disturbing the intended effect of shadows.

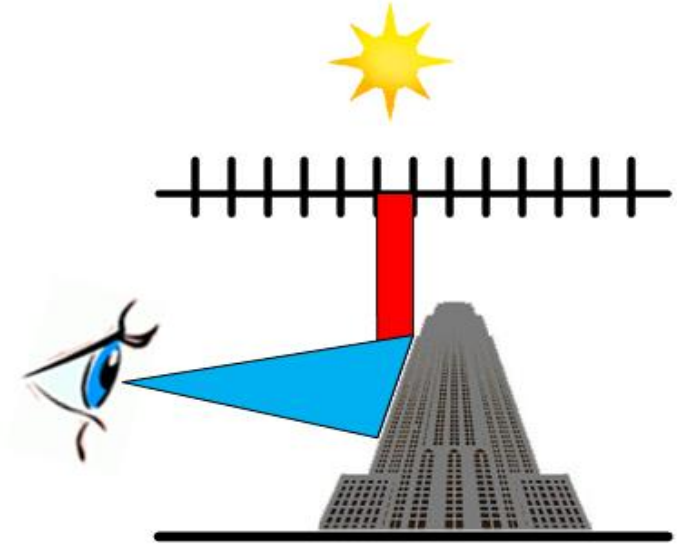

Figure 9: The result of projection aliasing is insufficient shadow map resolution for shadows in surfaces close to parallel to light-view direction.

Projection aliasing is only dependent on the angle between the light direction and the surface normal (as opposed to perspective aliasing, where the angle between camera and light direction dictates the amount of aliasing).

Thus, projection aliasing will be largest if the surface normal is perpendicular to the light direction – in this case the surface projected into the shadow map has no area, leading to lack of depth information stored, resulting in arbitrary shadowing results and infinite aliasing projection error. If light direction is parallel to the surface normal, then the opposite case happens, leading to a maximum of depth information in the shadow map, avoiding projection aliasing.

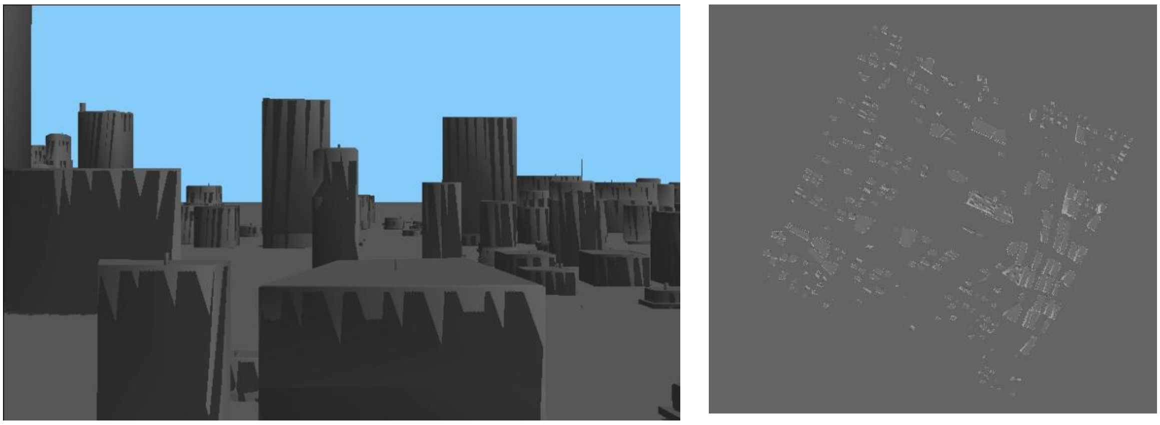

Figure 10: The projection aliasing artifacts (dark stripes at building walls) in the eye view (left image) are caused by too few samples of the cubes sides as seen from the light view (right image).

As seen in Figure 10, light direction is straight top-down and buildings' walls are perpendicular to the light direction. This is the extreme case of projection aliasing mentioned above, as the lack of depth information for buildings walls leads to arbitrary shadowing. The shadow should cover all the buildings walls, but instead, dark stripes appear over the walls. Also, while moving the camera, these stripes shape will change randomly, causing unwanted flickering effects.

2.2.3. Texture Resolution Constraints

In order to produce high-quality shadowing effects for large-scale virtual environments, a very large shadow map resolution is required. Current hardware still imposes a limitation to the maximum texture resolution. Hence it is insufficient to use a single huge shadow map. Consider for instance an outdoor scene with an area of 1 square km. If we select a shadow map resolution of 1024x1024, and considering a directional light perpendicular to the ground, each texel would occupy an area of roughly 1 square meter. Even considering the most up to date hardware, which allows for 8192x8192 as the maximum resolution for a texture, a single texel would occupy a square with more than 12 centimeters on the side.

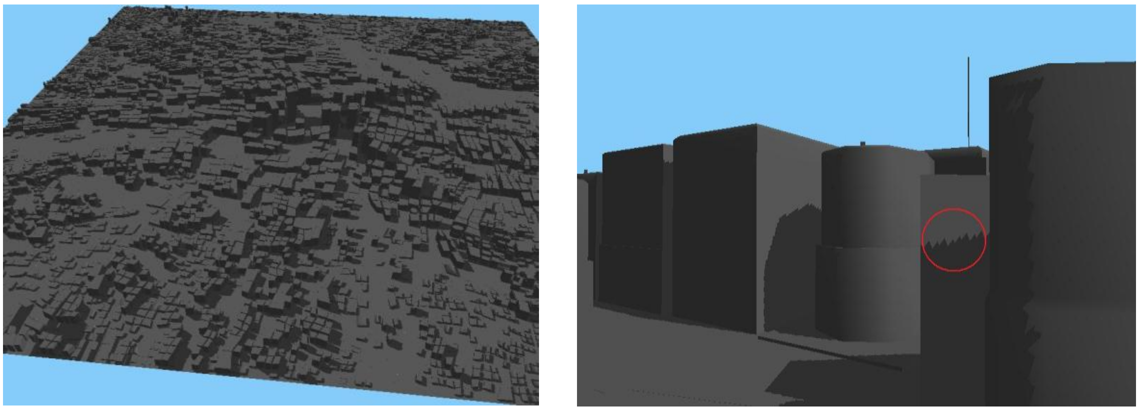

Figure 11: A single 4096x4096 shadow map has not enough resolution (right) to produce detail for a large-scale virtual environment with 8 square kms (left).

As shown in Figure 11 (above), although a large scene (8 square kms) can be rendered from far-away with satisfactory shadows (left picture), when closing up the camera (right picture) it is noticeable the lack of resolution of a large shadow map (4096x4096).

2.2.4. Self-Shadowing

Shadow mapping process has two instants where sampling takes place:

First when the shadow map is created viewing the scene from the light point-of-view. In this case, the depth information is stored in a regular grid.

Second, when the scene is rendered through the eye-point-of-view and the fragments are calculated, having to fit the output resolution. In many cases, these two processes lead to different samplings of the scene data.

In Figure 12, the shadow map is represented by a grid (in the top); the space between the grid strips represents a texel and again, we assume the light direction as straight top-down. The sampling is represented through the vertical lines starting in the middle of the corresponding texels of the shadow map. The camera is located at the side. As described in the picture, the distance from eye point-of-view should be equal to the depth stored in the shadow map, but in this case, observer’s distance is calculated as greater than the shadow depth sampled from the light point-of-view. The result will be the polygon erroneously shadowed by itself, which is obviously wrong. This artifact happens often in shadow mapping process and is known as self-shadowing, also called “shadow acne”.

Depth quantization problems also lead to self-shadowing. The shadow map is discretized, hence the information that is stores in the shadow map is not the exact depth, but a floating point (sometimes an integer) representation of limited precision. This representation often leads to conflicts. When a texel receives the depth information, it may not be represented exactly, being always rounded to the nearest depth available for the current precision.

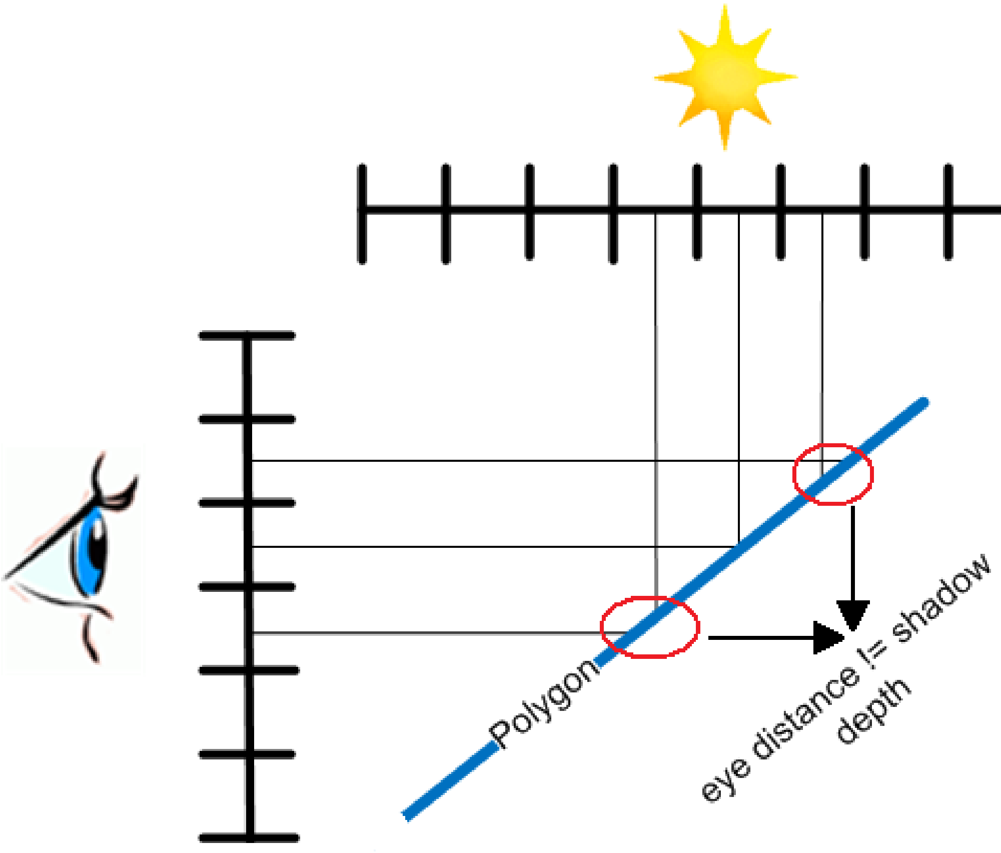

Figure 12: Sampling differences between light point-of-view and camera point-of-view can lead to incorrect self-shadowing.

In Figure 13, the shadow map is again represented by a grid (in the top); the space between the grid strips represents a texel and we assume the light direction as straight top-down. The vertical lines coming from the light source may not assume their exact length, but just certain lengths that are limited to the precision of the shadow map.

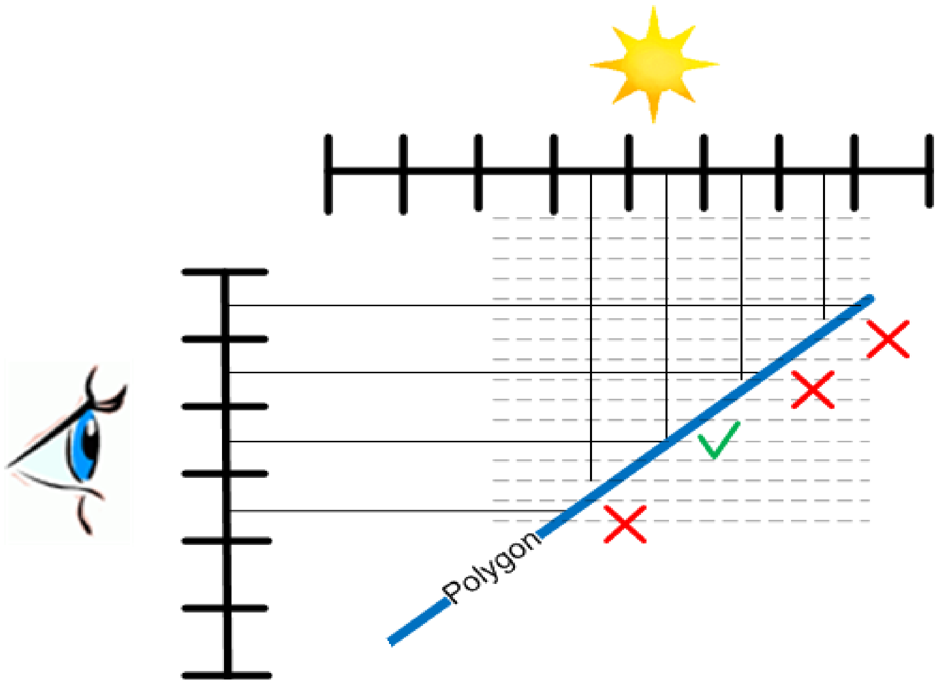

Figure 13: Depth quantization due to shadow map finite precision also leads to incorrect self-shadowing

The light green horizontal lines represent the depth values able to be stored in the shadow map. Therefore, the depth that is captured from the shadow map for the depth comparison is not the exact depth of the geometry.

Figure 14 shows the results of incorrect self-shadowing in a large scene. Although the expected shadows are well represented, we can notice visible patterns of shadowed/unshadowed regions in places that should be mostly unshadowed (e.g.: on the ground).

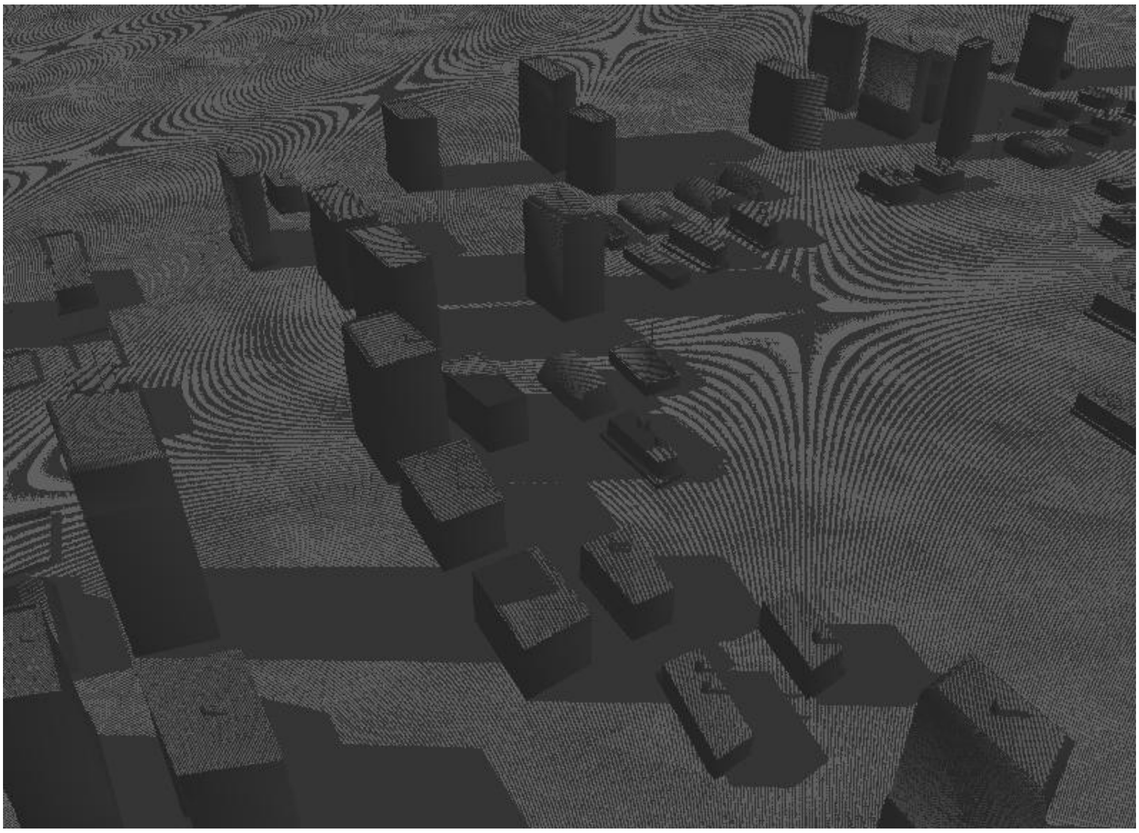

Figure 14: The moiré patterns are caused by sampling conflicts and depth quantization problems

In light to this problem, a technique that is usually adopted is to create the shadow map with front face culling enabled and then render the scene with back face culling enabled, so that these conflicts do not take place anymore. Unfortunately, most of the times, 3D models are not as solid as they appear to be (not including some back faces that are supposed to exist in a real model), leading to light leaking artifacts, in the boundaries between the model and the ground.

Thus, using the face culling technique for avoiding self-shadowing is not enough to remove this artifact.

2.2.5. Adaptive Light Frustum

The adaptive light frustum procedure increases the effective shadow map resolution by focusing as much as possible the shadow map on the objects within the view frustum as seen from the light. This is achieved by adapting the frustum from the camera that is capturing the shadow map so that it is as tight as possible around the visible geometry. Nevertheless, in large scenes the empty gaps between objects may still represent a waste of great amount of resolution.

An issue that arises from the adaptive light frustum is the sudden change in shadow resolution, also known as the continuity problem, resulting in the flickering of shadows as the camera moves around. This problem may be noticeable as large shapes enter the view frustum due to camera movements, or when occluders not present in the view frustum come into play. The frustum of the light may be significantly altered to include all the new geometry and its increase in size may result in a loss of shadow map or depth resolution.

Figure 15 represents the continuity problem explained above, when a drastic change in the scene as viewed from the camera point-of-view (a big object is added or removed from the scene) changes the resolution of the shadow map due to the adaptive light frustum procedure.

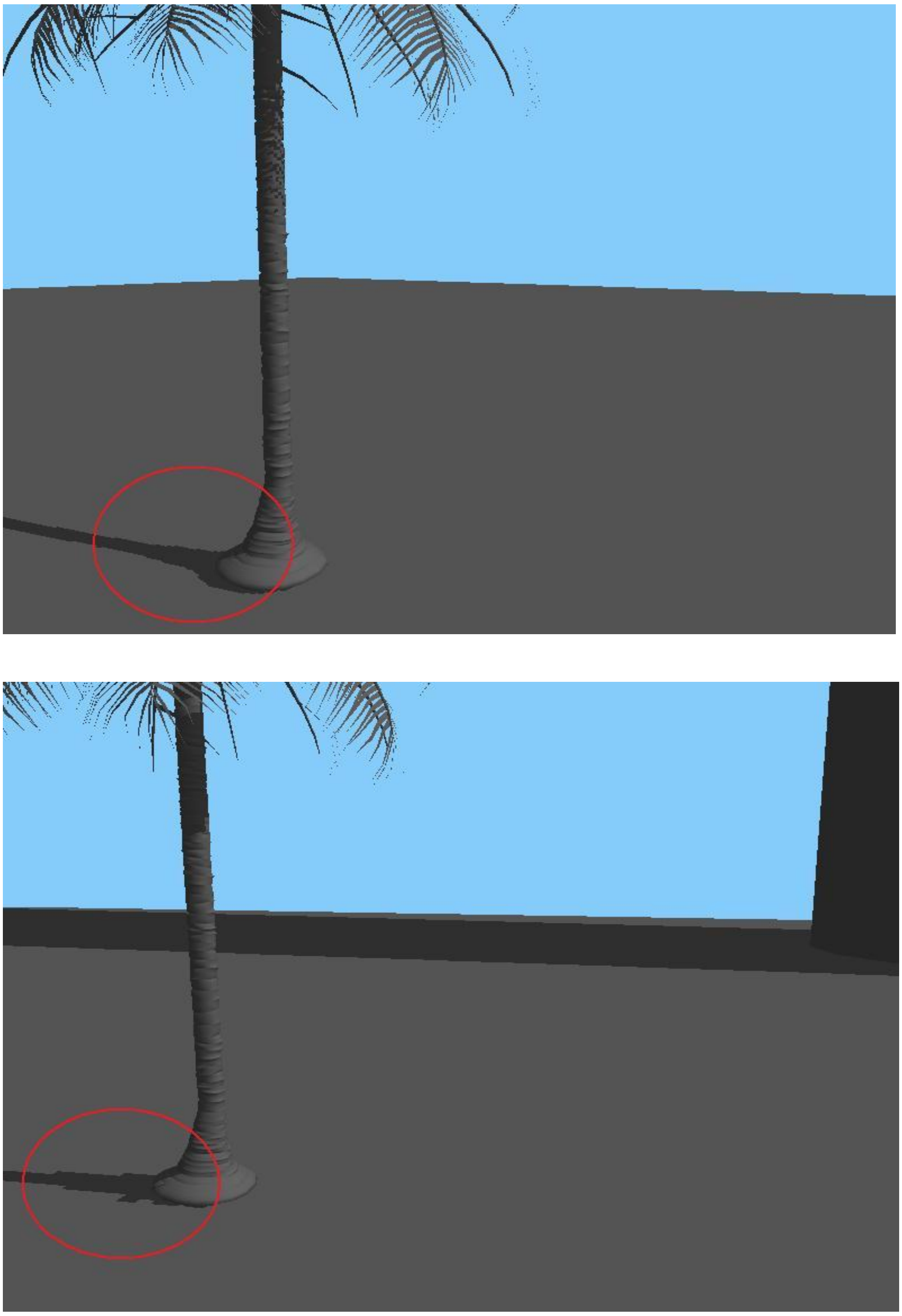

Figure 15: Continuity problem due to adaptation of the light frustum to the scene as viewed from the camera: The top picture shows a scene containing a palm tree; as the camera moves, a big building enters the scene, introducing drastic changes to the shadow map resolution.

2.2.6. Hard Shadows Binary Status

As said before, the crispy boundaries of the hard shadows make them unrealistic due to the total absence of penumbra. In the limit, it can lead to erroneous perception of the scenario, mistakenly perceiving the dark shadow as an object instead of what a shadow is: absence of direct light projected on a surface. Figure 16 shows the difference between a shadow mapping process with no filtering and a shadow mapping with filtering applied (which is different from soft-shadowing in the sense of umbra/penumbra generation).

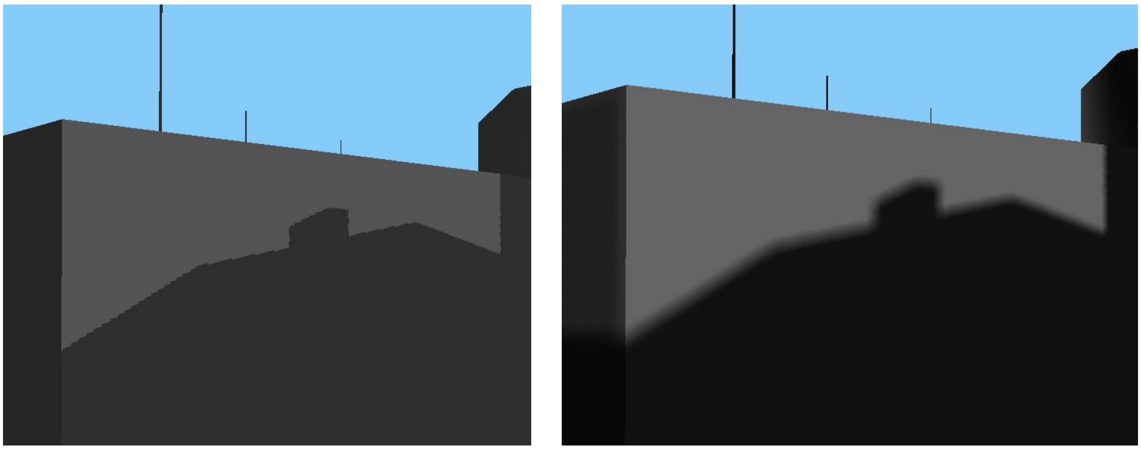

Figure 16: Hard shadow (left) vs. filtered shadow map (right).

2.3. Percentage Closer Filtering Algorithm

Percentage Closer Filtering was proposed by W. Reeves [PCF87] in 1987.

This technique calculates the percentage of the surface that is closer to the light and therefore not in shadow, avoiding the already mentioned binary status of hard shadows. This is done by performing multiple shadow map comparisons per pixel and averaging the results, outputting the percentage visibility value as a shadow indicator.

Figure 17 shows how percentage closer filtering works, as opposed to standard shadow mapping with no filtering process. While rendering the scene, for each pixel it will be made a comparison whether it is in shadow or not. In case of PCF, it will rely on a kernel surrounding the pixel, being stored in each element of the grid the boolean result of the depth comparison between the surface distance and the shadow map (as with standard shadow mapping). After all the elements of the kernel are calculated, the intensity of the shadow will be the percentage of shadowed elements in the kernel. That being said, it will be easy to understand that, at shadow boundaries, the binary status will disappear, giving place to a smoother transition between shadowed surface and lit surface (smoothness will depend on the kernel size).

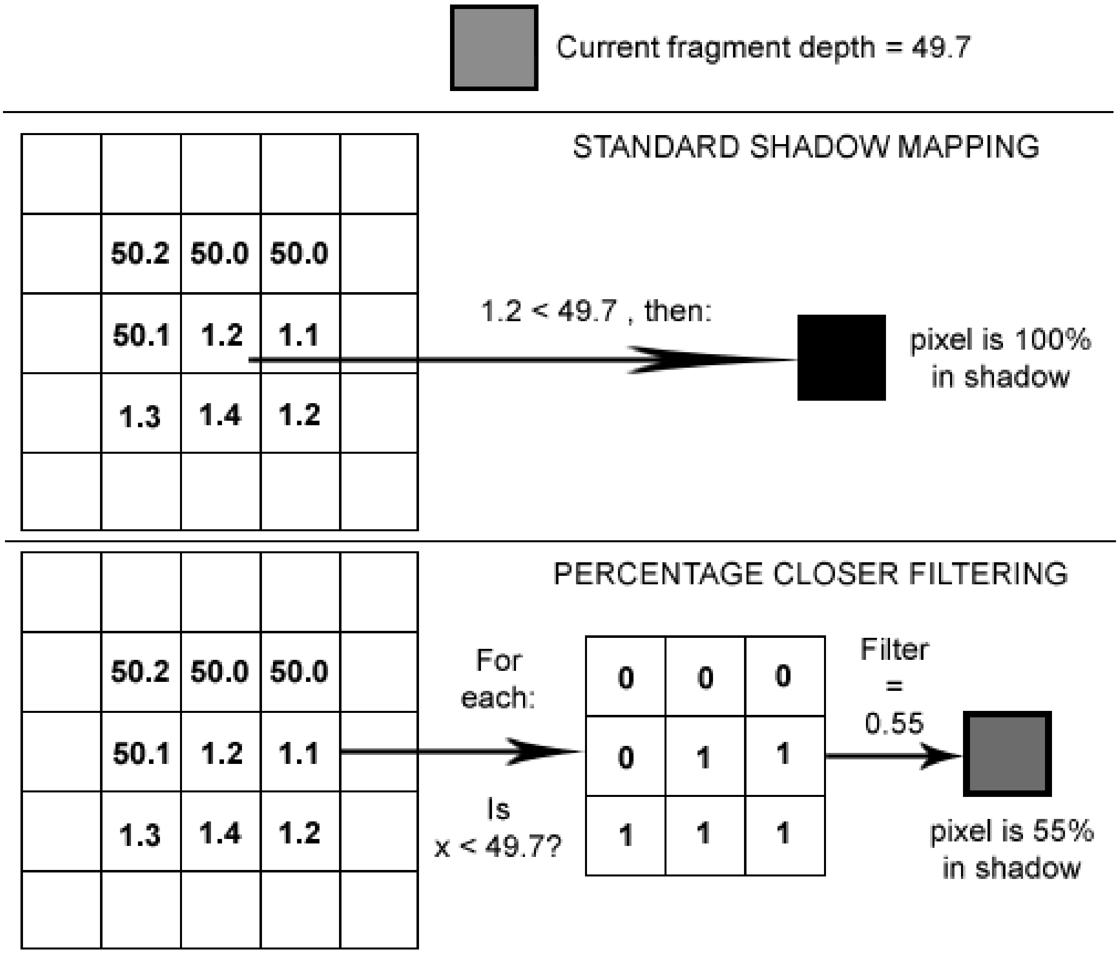

Figure 17: Standard Shadow Maps process (top) vs. Shadow Maps with Percentage Closer Filtering process (bottom).

For some years now, most graphics cards support PCF, which greatly enhances shadow quality when using shadow mapping in real-time rendering.

2.4. Conclusion

In this chapter we introduced shadow mapping as an important technique to enhance realism of virtual environments. We have also discussed the various problems associated to shadow mapping: perspective aliasing, projection aliasing, texture resolution, self-shadowing, adaptive light frustum and hard shadows binary status. Finally, we also introduced the basic filtering technique Percentage Closer Filtering, which is also supported by most graphics cards.

New algorithms have been proposed, solving distinct issues (as described earlier). All approaches are intended to be real-time rendering, based on standard shadow mapping, updating shadow maps in every pass. These techniques will be presented in the next chapter

3. State of the Art

In this section we start with a brief approach over the most important contributions for the field of shadow mapping. Then, we will choose the most relevant approaches for our study and analyze them more thoroughly. We finalize by presenting the main advantages and limitations of the analyzed algorithms.

As said before, Shadow Mapping suffers of some problems, mainly aliasing issues. This can be improved by the addition of other techniques to the basic shadow mapping algorithm. Solutions range from generating the shadow map in post perspective space, constructing hierarchical and adaptive shadow maps to increase the resolution where needed, to scene partitioning into multiple shadow maps and finally shadow map filtering.

All the algorithms mentioned below will work as improvements to the basic shadow mapping algorithm. Some are complementary, while others are different approaches for the same problem.

A number of papers have tried to solve perspective aliasing using perspective transformations. Perspective re-parameterization has been first proposed in Perspective Shadow Maps [PSM02], in which both the scene and the light source are transformed to the post-perspective space, which is the space after perspective transformation (also known as normalized device coordinate space). The shadow map is generated in this space by rendering a view from the transformed light source to the transformed view frustum. Since the shadow map sees the scene after perspective projection, perspective aliasing can be significantly decreased. Despite its drawbacks, this paper has inspired and opened the door to more general shadow map re-parameterization approaches.

Light Space Perspective Shadow Maps [LiSPSM04] is a work that avoids some of the PSM inconveniences and leverages the aliasing distribution over the whole depth range. Trapezoidal Shadow Maps [TSM04] is another perspective re-parameterization and very similar in concept to PSM and LiSPSM. The essential difference between TSM and prior perspective parameterizations is that a different perspective warping transform is used in TSM, such that the user-focused portion at the front of the frustum is mapped to the 80% line on the shadow plane.

Adaptive Shadow Maps [ASM01] reduces aliasing by storing the shadow map as a hierarchical grid structure. By evaluating the contributions of shadow map pixels to the overall image quality, it is refined to create higher resolution at regions that contain shadow boundaries. Artifacts at those critical regions are thus greatly reduced without requiring a flat shadow map of huge resolution. However, the traversal and refinement operations require many rendering passes and aren't feasible for real-time applications for current graphics hardware.

Practical Shadow Mapping [PraSM02] proposes bounding box calculation for the view frustum. The tight fitting frustum makes the shadow map focusing to the visible part of the scene, enabling an available shadow map resolution increase.

Plural Sunlight Depth Buffers Shadow Mapping [PluSM01] approach uses a dynamic texture array comprised of multiple shadow maps with varying resolutions. It divides the view frustum into several parts to approximate the continuously varying of the resolution along the distance from the view point.

Parallel-Split Shadow Mapping [PSSM06] proposes to divide the viewing frustum into multiple splits, generating a shadow map for each one, improving the overall scene shadows resolution. The main differences to [PluSM01] are the uniform resolution distribution along the shadow map and the split scheme, which is efficiently calculated according to a pre-determined set of rules, instead of the complicated and time-consuming computations for optimal lengths of split parts of [PluSM01] (which are generated using recursive searching procedures).

Percentage Closer Filtering [PCF87] is the first approach to alleviate projection problems by filtering shadow map texture. Extending the concept of classical bilinear filtering used in texture sampling, it enables anti-aliasing around shadow boundaries. PCF determines the coverage of a camera pixel in light space and applies the shadow test to a number of samples distributed over this region to get a filtered result.

Variance Shadow Mapping [VSM06] is a probabilistic approach that supports pre-filtering, and additional convolutions. When the shadow map is calculated, the z and z-squared values are stored, being used during the render to estimate the probability of a point being lit or not. It may produce noticeable high frequency light leaking artifacts for scenes with a high depth complexity.

Convolution Shadow Mapping [CSM07] achieves anti-aliased shadows by approximating the shadow test with a Fourier series expansion. Depending on the truncation order, z-values are converted into several basis textures. In the final rendering, pre-filtered texture samples are fetched to reconstruct a smoother shadow. CSM have the same desirable properties as VSM, but do not exhibit such severe light leaking artifacts. However, a reliable shadow test requires a high truncation order, which in turn increases memory consumption and filtering as well as reconstruction effort. This makes CSM less attractive for practical and real-time applications.

Finally, Exponential Shadow Mapping [ESM08] presents an algorithm that allows efficient pre-filtering, being inspired by CSM, but using a single-term approximation, while CSM uses typically 16 terms. Besides ESM not suffering from light leaking of VSM and being much faster than CSM, it can also exploit latest texture filtering modes supported by today’s graphics hardware (such as anisotropic filtering).

For a more thorough analysis of part of the existing algorithms, please refer to [SSA90] and [STIRTA04].

3.1. Perspective Shadow Mapping

Perspective Shadow Mapping (PSM) is the first algorithm suggesting shadow map post-perspective transformations resulting on a non-uniform distribution of the shadow map. This distribution enables high resolution shadows near the camera, and lower resolution as the shadows move away from the viewpoint.

A standard shadow mapping produces aliased shadows, and this effect is more perceivable near the camera due to the uniform distribution of the depth. Perspective shadow mapping reduces aliasing by applying a perspective transformation while generating the shadow map, such that nearby objects have increased shadow resolution, while faraway objects have reduced shadow resolution, which is less perceivable because of the distance to the camera.

Perspective shadow maps are computed in post-perspective space of the camera. This perspective is obtained by transforming the world to a perspective-distorted space where proximity produces objects enlargement and distance enables objects shrinking (see Figure 18).

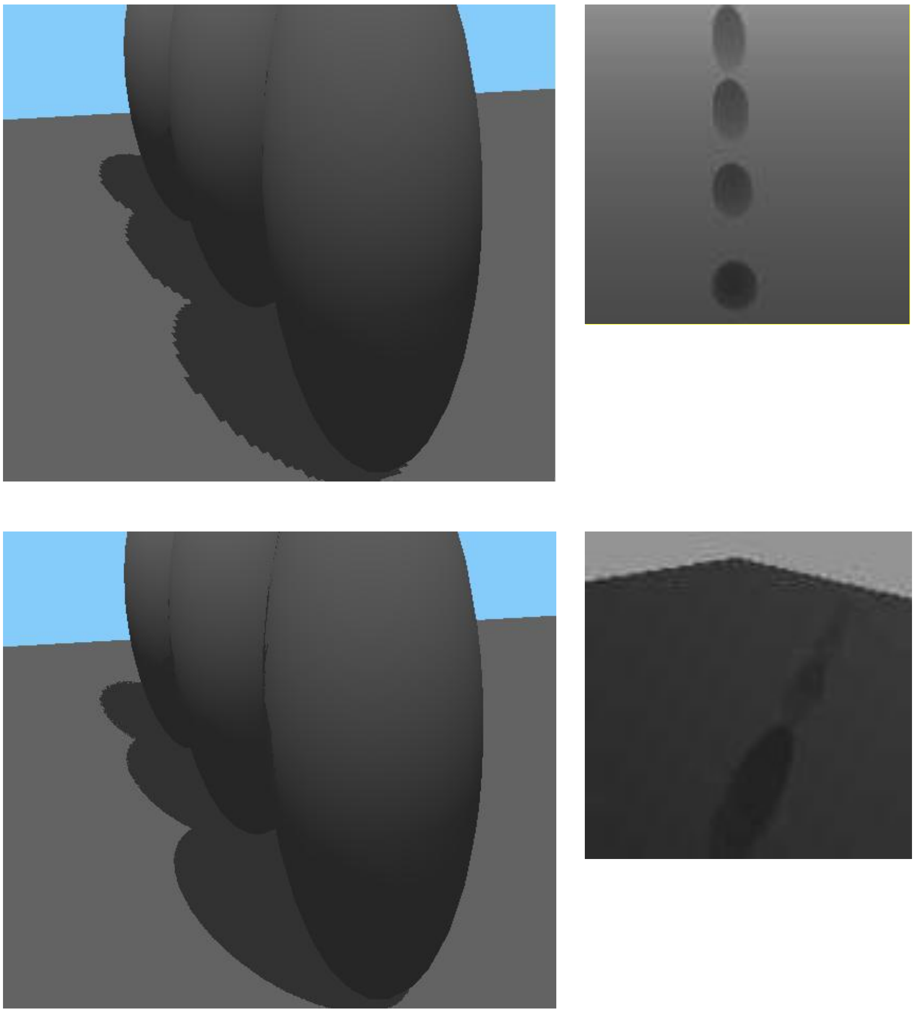

Figure 18: Standard shadow mapping (Top Left) and the respective shadow map (Top Right) generated as if the scene was rendered from the light’s point of view. Perspective Shadow mapping (Bottom Left) reduces aliasing by applying a perspective transformation while generating the shadow map (Bottom Right).

PSM distinguishes itself from SSM mostly by using a 4x4 matrix which represents the shadow map projection with homogeneous coordinates.

It projects the camera view frustum to a unit cube, being the final image generated by a parallel projection of this cube along z. Thus, the first step is to map the scene to post-perspective space and generate a shadow map in this space. This is done by rendering a view from the similarly transformed light source to the unit cube (Figure 19).

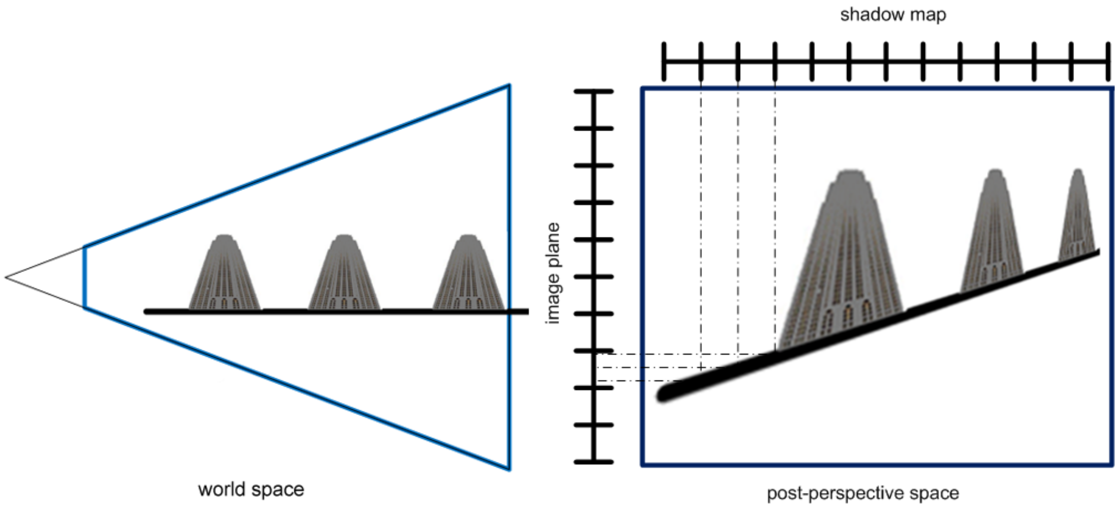

Figure 19: World View (left) is transformed to Post-perspective space (right) in order to perform perspective shadow mapping. The shadow map pixels, once projected on the scene surface will be evenly distributed, according to the camera-view.

3.1.1. Implementation – General Case

Although PSM relies on a relatively simple transformation to the standard shadow mapping algorithm, this technique has many particular cases with non-trivial resolutions where the general case in not enough to produce detailed, reliable shadows. These special cases will be briefly addressed later in this section.

As for the general case, this algorithm is composed of two steps, which are described below:

1st step - Shadow Map Generation

In order to create the shadow map, it if first needed to apply a post-perspective transform to the camera view. Let MV and MP be the model-view and projection matrices for the eye-view, respectively. The post-perspective transform M for the eye-view is:

$M = MV * MP$

Also, the light source should be transformed similarly.

Let LV and LP be the model-view and projection matrices for the light view, respectively. Then, the post-perspective transform L for the light-view is:

$L = LV * LP$

Having M and L, it is now possible to calculate the transform matrix, which is:

$L' = M * L$

Using L' for the perspective transform, the shadow map is now created.

2nd step – Rendering the scene

While rendering the scene, each fragment of the world should be multiplied by the inverse of L' for getting the texture coordinate of the shadow map. Then, the standard shadow map comparison for shadow determination is used.

3.1.2. Special Cases

In order to produce reliable shadows, some scenarios that are not covered by the general case have to be considered. These cases relate to the light source type and its position relative to the view frustum, also taking into account all objects casting shadows into the view frustum, even if they are not present in that view. A short overview will now be presented, for a more thorough analysis please refer to [PSM02].

Directional/Point Light Sources

Directional light sources can be considered as point lights at infinity and the perspective mapping moves these sources to a finite position. The settings for all possible cases are shown in Figure 20, where the top cases are related to transformation of directional light sources, and the bottom cases refer to transformation of point light sources.

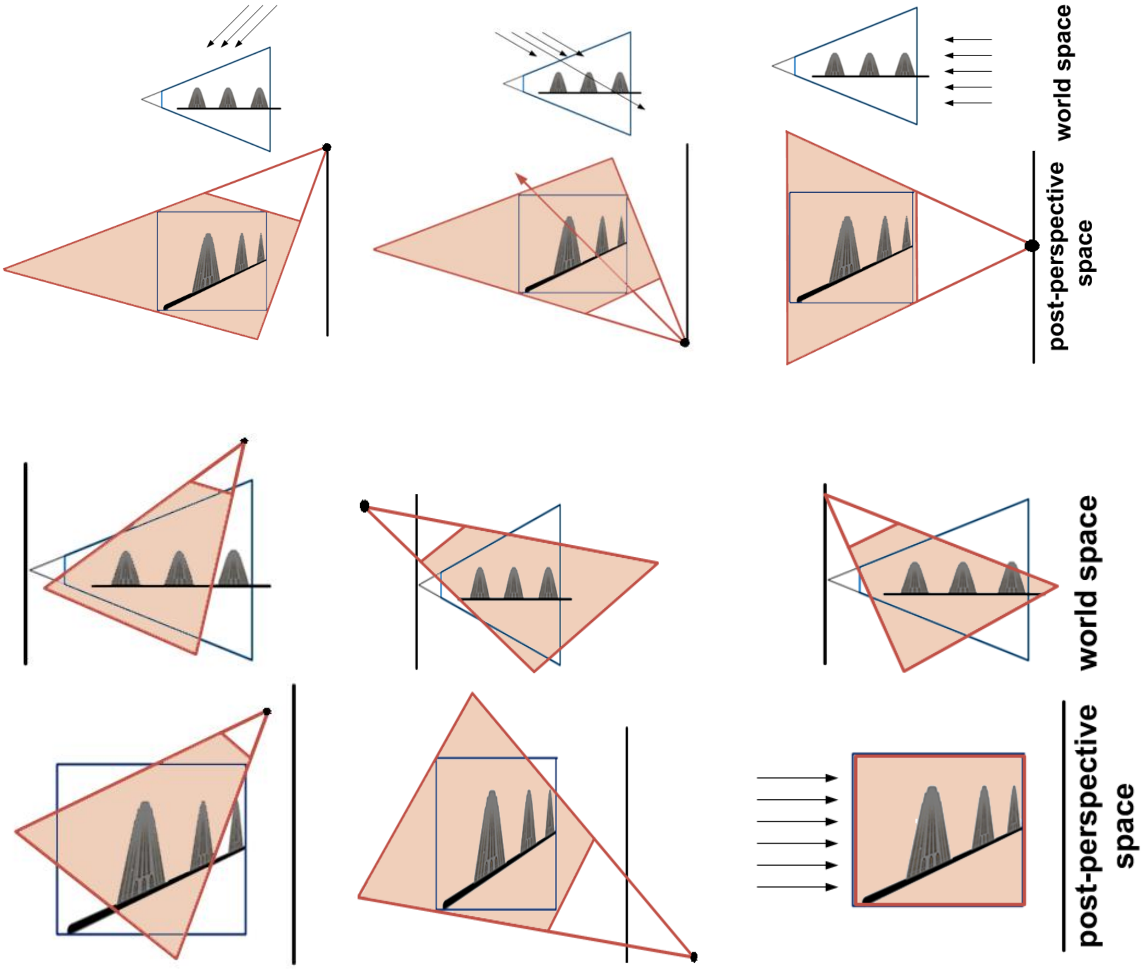

Figure 20: Mapping of lights in world space to post perspective space. Directional lights in world space become point lights in post perspective space (Top). Also, lights from behind the camera become inverted. Point lights in world space remain point lights in post perspective space, except for boundary cases (Bottom).

PSM results will depend greatly on how lights are positioned and oriented relative to the camera. For directional light sources, the success is proportional to the light direction perpendicularity to the camera direction. As the smallest angle formed between these two directions decreases, also the advantages of the PSM approach decrease (if the angle is 0, then PSM converges into Standard Shadow Maps).

As for point light sources, the algorithm effectiveness is also proportional to the distance between the point light and the view frustum. The further away from the view frustum the more similar a point light is to a directional light.

Including all Objects Casting Shadows

In addition to objects within the view frustum, a shadow map must contain all potential occluders outside the frustum that cast shadows onto the visible part of the scene.

Consider the scene present in Figure 21 (below). Let S be the bounding shape of all objects present in the scene, V be the view frustum as seen from the camera and L be the view frustum as seen from the light source. The light source is positioned at position l.

In order to compute the (world-space) region of interest H for shadow map generation, it is first needed to generate a convex hull M of all rays emanated from l to V. Then, H will be the result of the intersection of M with the scene bounding shape S and light frustum L:

$H = M \cap S \cap L$, where $M = \text{convex\_hull}(l \cup V)$

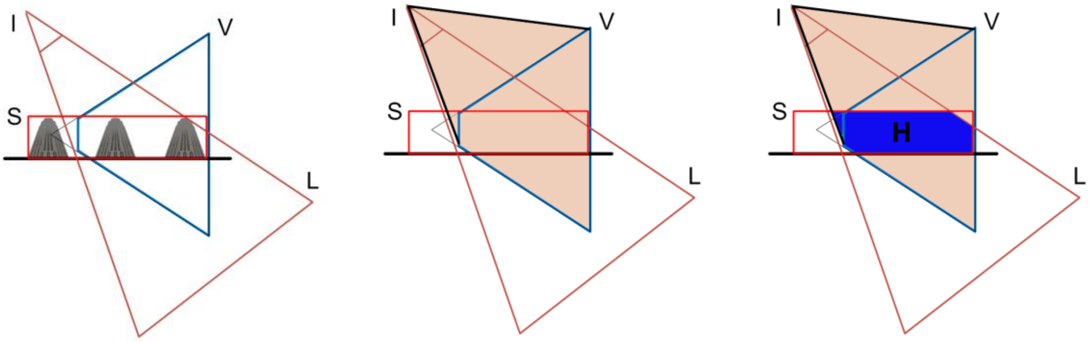

Figure 21: Computation of the scene's (world-space) region of interest for shadow map generation.

At this point of the process, the algorithm is still in world-space, so it is needed to move into post-perspective space before generating the shadow map.

While transforming lines from world to post-perspective space, points in the line may change their relative order: imagine a line intersecting the camera plane; the point of intersection is mapped to infinity, so these points behind the camera plane are projected to beyond the infinity plane.

Hence, converting H to post-perspective space imposes some concerns:

If H is completely in front of the viewer, it can be transformed to post-perspective space immediately;

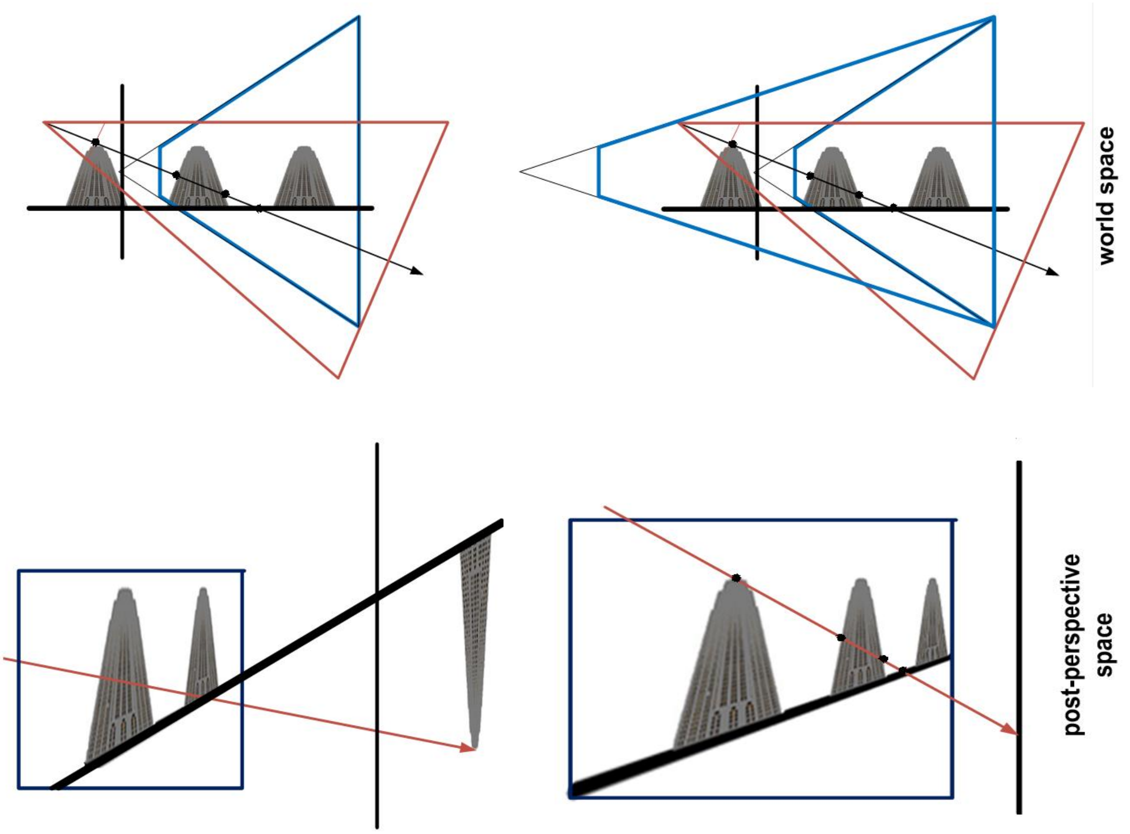

Figure 22: Case where the region of interest for the shadow map generation needs to be extended in order to avoid scene disfiguration introduced by the post-perspective transformation.

Else, if H has some objects casting shadows to the scene that are positioned behind the camera, a virtual shift to the camera is required so that the view frustum includes all objects casting shadows to generate the shadow map. If the camera is shifted back to infinity, then it would become an orthogonal camera, and post perspective space would be equivalent to world space (see Figure 22). In this case the benefits of the post perspective transformation are null, with the penalty of the geometry approximations and post-perspective transformations. In practice only a small shift is required for most cases.

3.2. Trapezoidal Shadow Mapping

Trapezoidal Shadow Maps (TSM) is proposed after Perspective Shadow Mapping, suggesting a transformation that is the result of trapezoidal approximations of the eye's frusta as seen from the light. Similarly to PSM, TSM produces a shadow map with enhanced distribution enabling high resolution shadows near the camera, and lower resolution as the shadows move away from the viewpoint.

This algorithm deals with the resolution problem of shadow maps, while treating the continuity problem, which is present in the majority of shadow mapping algorithms that try to fix the shadow aliasing problem using greedy adaptive frustum (e.g.: PSM, which introduces severe continuity problems due to its adaptive light frustum algorithm). Greedy adaptive frustum implies that the frustum usage is maximized for the projection used. It builds a light frustum that contains only the visible geometry and the occluders. As mentioned before, the light frustum may change significantly when new geometry enters the view frustum. TSM proposes a non-greedy optimization of the usage of the shadow map; hence the light frustum changes smoothly from frame to frame.

3.2.1. Reducing the resolution problem

One of the main challenges while addressing the resolution problem is to better utilize the shadow map, adapting the light frustum to the area within eye’s frustum.

Obviously, the resolution of a shadow map is inversely proportional to the size of the area of interest, so the tighter we can put the region of interest of a determined scene, the better resolution we get. Practical Shadow Mapping [PraSM02], in addition to uniformly spaced depth values, proposed a tight fitting frustum recurring to the smallest bounding box of the interest area.

TSM approach goes beyond the idea of Practical Shadow Mapping, claiming that the shadow map could be further optimized. When at light post-perspective space, the camera view frustum looks like a polygon with up to six edges

(Figure 23). A rectangular based frustum is clearly not the best approximation to the camera frustum. Large areas of the shadow map become useless under this approximation.

The main goal behind TSM is to find a better approximation, avoiding the non-visible areas from eye-view. In light of this problem, TSM suggests the trapezoidal shape, as opposed to the squared-box. A trapezoid is very similar to the view-frustum as seen from the top.

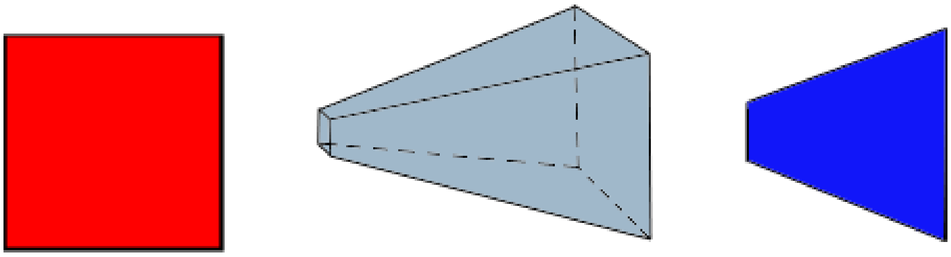

Figure 23: The most similar to shape to a view-frustum as seen from the top is noticeably a trapezoid. Using a square for the approximation, some non-interest areas of the scene would be included in the transformation.

Figure 24 refers to the wastage of the bounding-box approach proposed by Practical Shadow Mapping as opposed to the trapezoid approach proposed by TSM. Due to view-frustum shape, the bounding box approach will minimize less non-interest parts of the scene, when compared to the trapezoid approach. In addition to that, the trapezoid can stretch its base line so that it reaches the shape of a square, increasing its resolution for near-camera objects.

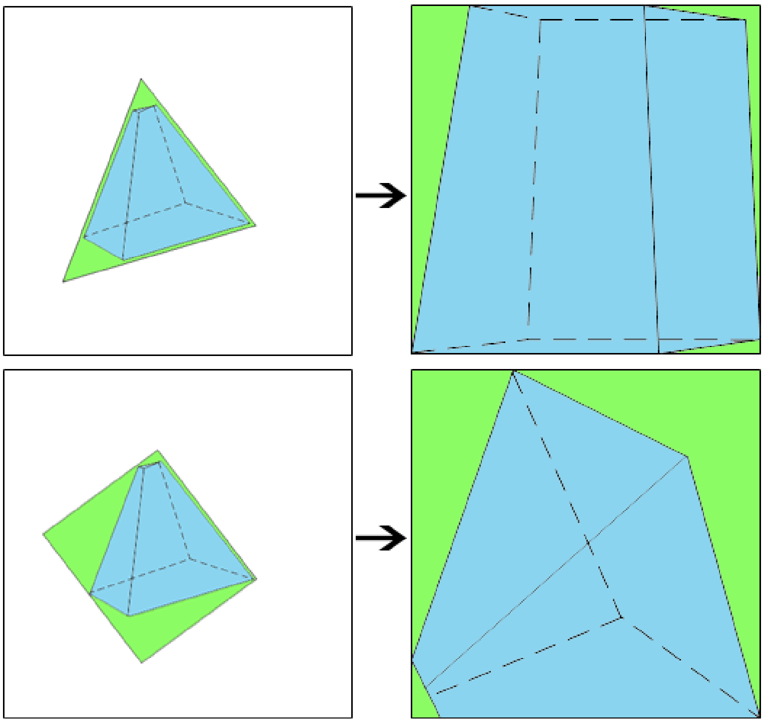

Figure 24: Trapezoidal approximation vs. Bounding-box approximation

The top and base parallel lines of the trapezoid enable a powerful mechanism for controlling the shadow map resolution for each frame, solving also the continuity problem associated to the greedy adaptive light frustum algorithms suggested by many shadow mapping techniques.

The increase in gained resolution in the ideal case, where the light direction and the eye vector are perpendicular, is mostly in areas near the view camera; hence we may end up with relative under-sampling for objects further away from the camera.

Associated to the trapezoid side lines, TSM presents a mechanism to better spread the available resolution to objects within a specified focus region. While the bounding box approach cannot stretch its shape to the unit cube, the trapezoid can be warped according to a pre-determined rule (described later in this section) for avoiding over-sampling in objects near the camera and under-sampling as distance to the camera increases.

3.2.2. Implementation

Similarly to standard shadow mapping, TSM first creates the shadow map and then – while rendering the scene – for each fragment, it is determined whether it is shadowed or not. Despite the similarities, a comprehensive and sometimes computationally expensive calculation needs to be made before the shadow map generation: the so called normalization matrix N needs to be calculated in order to (right before shadow map generation) transform the post-perspective space of the light to a N-space, where N refers to the trapezoidal space described above.

Thus, instead of the standard 2 steps, TSM is composed of 3 main steps, which are described below:

1st step - Normalization matrix calculation

The normalization matrix NT results of the mapping of the four corners of the trapezoidal T into a unit square (described later, in this section).

In order to calculate T, one needs to obtain the base and top lines of the trapezoid (referring to the approximation of the far and near planes respectively). Then, the side lines of the trapezoid are computed using the bottom and top lines. These steps are done as follows:

Trapezoidal base and top lines determination

Let E be the area within the eye's frustum as seen from the light. The base and top lines refer to the approximation of the far and near planes of E respectively.

The algorithm for calculating these lines is as follows:

- Transform eye's frustum into post-perspective space of the light (E is obtained);

- Compute the center line *l*. This line connects the centers of the near plane and far plane of E;
- Generate the 2D convex hull H of E;
- Compute the top line *lt*, which is perpendicular to *l* and touches the boundary of H: *lt* intersects *l* at a point closer to the center of the near plane of E than that of the far plane;
- Compute the base line *lb* which is parallel to *lt* (not being the same), also touching the boundary of H.

Trapezoidal side lines determination

The base line of the trapezoid (which is related to the far plane) should always be wider than the top line (which is related to the near plane). As the trapezoid is stretched to fit the unit square, the top line is stretched until it has the same length as the base line. With this process, the objects close to the near plane will have over-sampled shadows, while the objects close to the far plane will have under-sampled shadows.

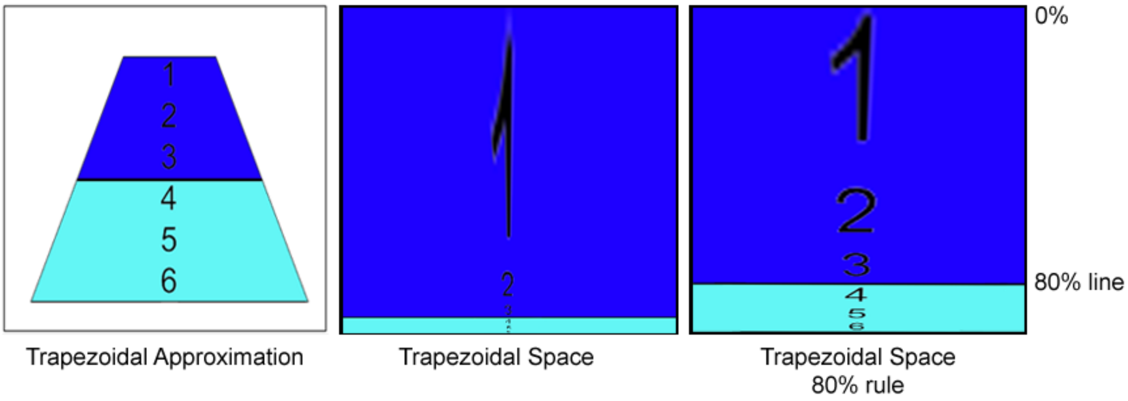

Figure 25: Side lines of the trapezoid (left) are calculated according to the 80% rule (right), so that the stretching of trapezoid does not introduce sampling artifacts (center)

To avoid this, TSM proposes an algorithm for achieving a balanced shadow mapping, reducing both under-sampling and over-sampling (see Figure 25 –

above). This algorithm relies on the side lines of the trapezoid, which dictates the stretching effect provided by the transform matrix NT.

For more details about the determination of trapezoidal side lines, please consult Appendix A.

Trapezoidal-to-square transformation steps

One last step for calculating the transformation matrix NT is needed: The four corners (t0, t1, t2, t3) of the trapezoid previously generated need to be mapped to a square that refers to the front side of the unit cube (Figure 26).

Thus, NT is generated according to the following equations:

$$T_{(-1.-1))} = NT * t0$$
$$T_{(+1.-1))} = NT * t1$$
$$T_{(+1.+1))} = NT * t2$$
$$T_{(-1.+1))} = NT * t3$$

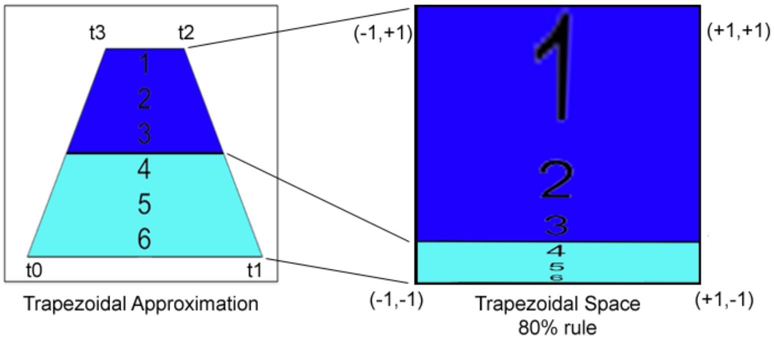

Figure 26: Trapezoidal transformation: mapping trapezoidal corners to square edge corners

These equations establish the connection between the trapezoid and the square, being NT used to move each of the corners t0, t1 ,t2 ,t3 to the respective corner of the cube.

In order to achieve this, TSM propose a sequence of steps formed of rotation (R), translation (T1, T2, T3), shearing (H), scaling (S1, S2) and normalizing (N) operations, each of them resulting in a matrix. At the end, these matrices are combined, returning the matrix normalization matrix NT.

This process will now be described:

1. Calculate T1

It is first needed to center the top edge of the trapezoid to the origin of the light post perspective view:

$$u = \frac{t2 + t3}{2}$$

$$T1 = \begin{bmatrix} 1 & 0 & 0 & -u_x \\ 0 & 1 & 0 & -u_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

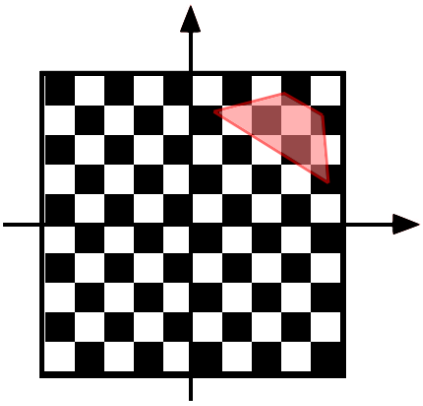

Figure 27: TSM application of T1

2. Calculate R

Then, the trapezoid needs to be rotated, so that its top edge is collinear with the x-axis:

$$u = \frac{t2 - t3}{\left| t2 - t3 \right|}$$

$$R = \begin{bmatrix} u_x & u_y & 0 & 0 \\ u_y & -u_x & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

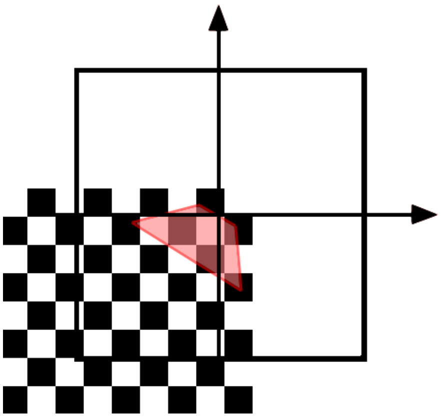

Figure 28: TSM application of R

3. Calculate T2

After step 2, the point *i* resultant of the intersection of the side lines is translated to the origin:

$$u = R * T1 * i$$

$$T2 = \begin{bmatrix} 1 & u_y & 0 & -u_x \\ u_y & 1 & 0 & -u_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

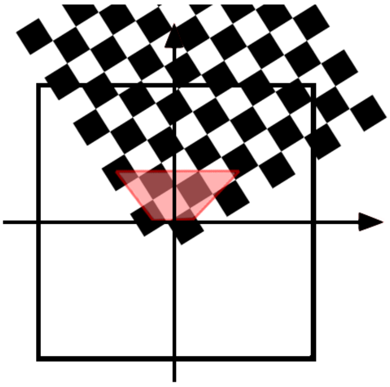

Figure 29: TSM application of T2

4. Calculate H

Shearing needs to be applied to the current trapezoid, so that it becomes symmetrical along the y-axis:

$$u = \frac{T2 * R * T1 * (t2 + t3)}{2}$$

$$H = \begin{bmatrix} 1 & \frac{-u_x}{u_y} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

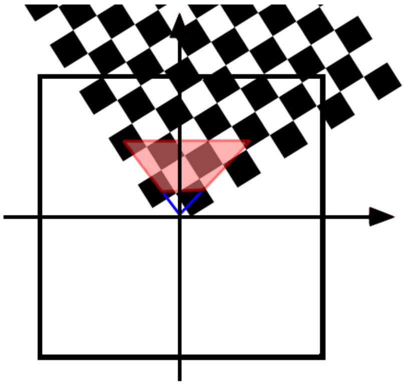

Figure 30: TSM application of H

5. Calculate S1

Scale the trapezoid, so that the angle formed between the side edges is 90º and also the distance between the top edge and the x-axis is 1:

$$u = H * T2 * R * T1 * t2$$

$$S1 = \begin{bmatrix} \frac{1}{u_x} & 0 & 0 & 0 \\ 0 & \frac{1}{u_y} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

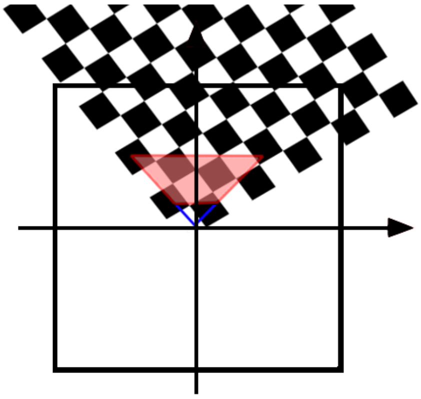

Figure 31: TSM application of S1

6. Calculate N

Now, transform the trapezoid into a rectangle:

$$N = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

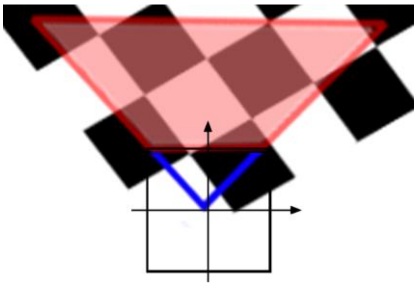

Figure 32: TSM application of N

7. Calculate T3

Then translate the rectangle to the origin, so that both center of rectangle and origin should be coincident:

$$u = N * S1 * H * T2 * R * T1 * t0$$

$$v = N * S1 * H * T2 * R * T1 * t2$$

$$w = -\frac{\frac{u_y}{u_w} + \frac{v_y}{v_w}}{2}$$

$$T3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & w \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

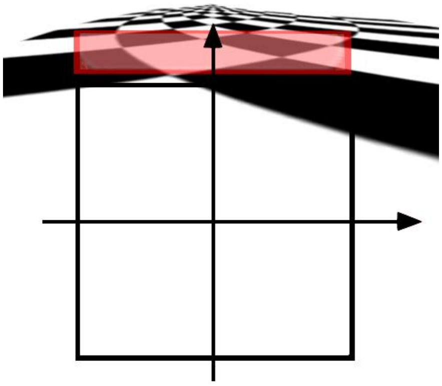

Figure 33: TSM application of T3

8. Calculate S2

Finally, scale the rectangle along the y-axis, so that it acquires the form of the square that covers the front side of the unit cube:

$$u = T3 * N * S1 * H * T2 * R * T1 * t0$$

$$S2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{u_w}{u_y} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

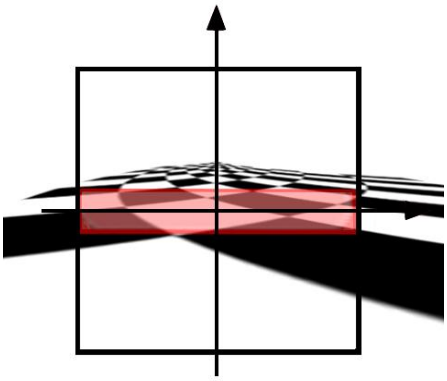

Figure 34: TSM application of S2

9. Calculate NT

Now that all required matrices are set, the transform matrix can be generated as follows:

$$NT = S2 * T3 * N * S1 * H * T2 * R * T1$$

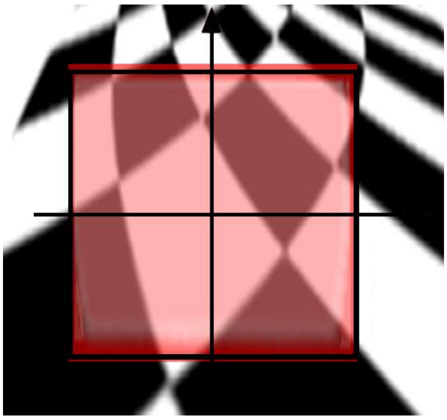

Figure 35: TSM application of NT

2^{nd} step - Shadow Map Generation

In order to use the normalization matrix calculated in the first step, it is needed to create the post-perspective matrix of the light view:

Let LV and LP be the model-view and projection matrices for the light view, respectively. Then, the post-perspective transform L for the light-view is:

L = LV * LP

Having NT (from the previous step) and L, it is now time to calculate the transform matrix, which is:

T’ = NT * L

Using T’ for the N-space, the shadow map is now created.

3^{rd} step – Rendering the scene

While rendering the scene, each vertex should be transformed by the N-space transformation matrix NT and use the resultant texture coordinates as the shadow map position, comparing the fragment z-value with the one present in the shadow map for shadow determination.

3.2.3. Solving the Polygon Offset Problem

Although the transformation matrix NT greatly improves the resolution problem and continuity problems, it worsens the polygon offset problem due to a non-uniform distribution of the z-values. This happens because, affecting the x and y coordinates values of each vertex inside the trapezoid, NT also affects the z coordinate value accordingly. This introduces severe polygon offset artifacts to generated shadows, even with the standard polygon offset resolution enabled (if bias is too big, nearby areas' shadows will disappear; else, if bias is too small surface-acne will appear in distant areas).

Figure 36 shows an example of a scene shadowed with Trapezoidal Shadow Mapping algorithm:

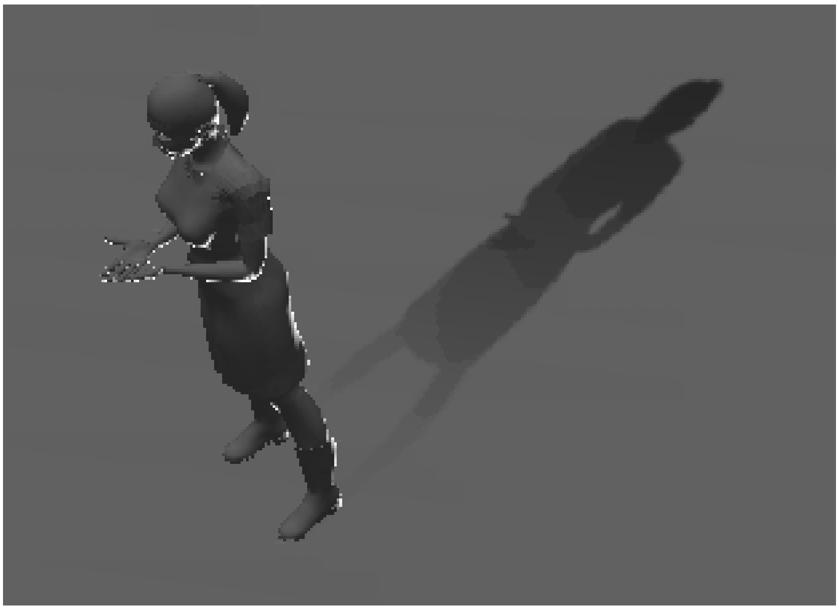

Figure 36: Scene shadowed with TSM using standard polygon offset problem (large bias).

A way to solve this is to transform only the x and y vertex values as follows:

In the shadow map generation stage, the vertex shader transforms each vertex v to VT = (XT,YT,ZT,WT) and assigns VL = (XL,YL,ZL,WL) as its texture coordinate. Then, the fragment stage replaces the fragment depth by ZL/WL added by an offset.

In the shadow determination step, at vertex shader each vertex is transformed to the post-perspective space of the camera, being computed VT = (XT,YT,ZT,WT) and assigned VL = (XL,YL,ZL,WL). At fragment shader, each fragment shadow status is tested by comparing ZL/WL with the stored shadow map, indexed by XT/WT and YT/WT.

Figure 37 shows the scene presented in Figure 36, now using the offset problem resolution proposed by TSM.

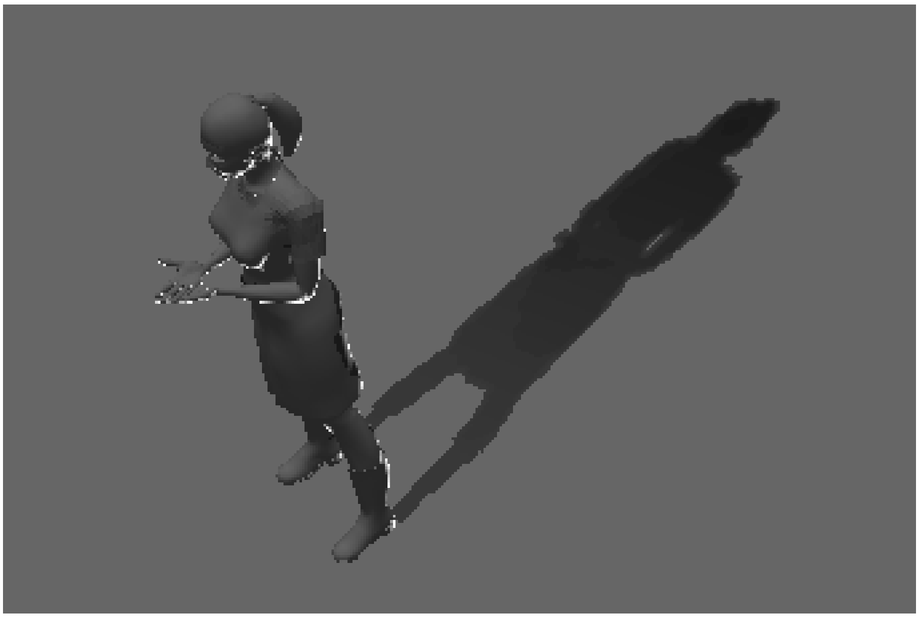

Figure 37: Scene shadowed with TSM after polygon offset problem tackling.

3.2.4. Special Cases

The algorithm described so far assumes that the light frustum contains the view frustum. This is the case in outdoor scenes using sun light for instance. However, when considering short range point lights this may not be the case.

In light for this problem, TSM needs to adapt its algorithm for cases where the camera-view frustum does not lie completely within light-view frustum. Hence the algorithm described above needs to be altered so that it only transforms the portion of the camera-view frustum that is inside of the light-view frustum, since the remains are not illuminated.

Thus, TSM needs to process the area *A* that is the intersection of both light and camera view frustums and compute the center point *c* of *A*. Hence the line *l* mentioned above is now the line passing through the camera point of view and the point c. The rest of the algorithm should be recomputed accordingly.

For more details on the re-computation of the trapezoid for cases where camera-view frustum is not inside light-view frustum, please refer to [TSM04].

3.3. Parallel-Split Shadow Mapping

According to Parallel-Split Shadow Maps (PSSM), it is impractical to achieve sufficient sampling densities for a large scene with a single shadow map, even with a high resolution texture.

PSSM as the above discussed techniques, lies in the fact that it is needed to produce different samplings densities based on the distance of points to the viewer. However, instead of proposing post-perspective transformations to the light view, this approach splits the view frustum into multiple discrete layers through split planes disposed along the view plane. Each split is thus rendered to an independent shadow map (see Figure 38).

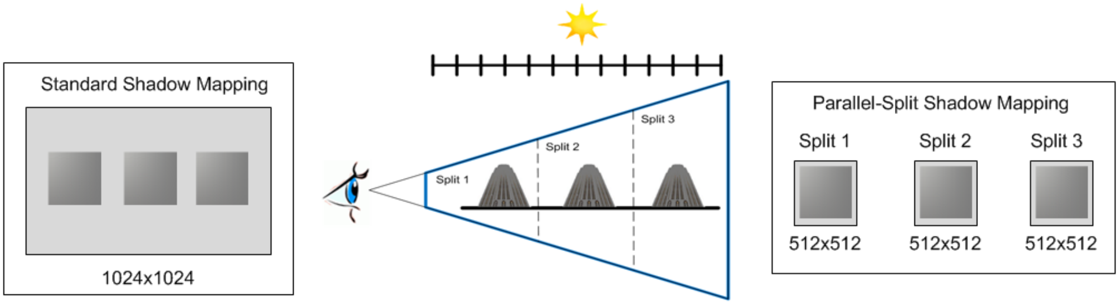

Figure 38: The view frustum is split into three parts, each of them will be rendered to a different shadow map with the same resolution

Hence each rendered shadow map will be the result of different parameterizations applied for each layer. A single shadow map with large size will be replaced by multiple, smaller ones, which requires less memory than the required for covering the same area with the standard approach and at the same time providing improved shadow resolution.

This algorithm assumes lights as directional, but since the shadow maps are rendered in the light's post-perspective space, all objects need to be transformed, including point light sources, which can at this point be converted to directional light sources (see Figure 39).

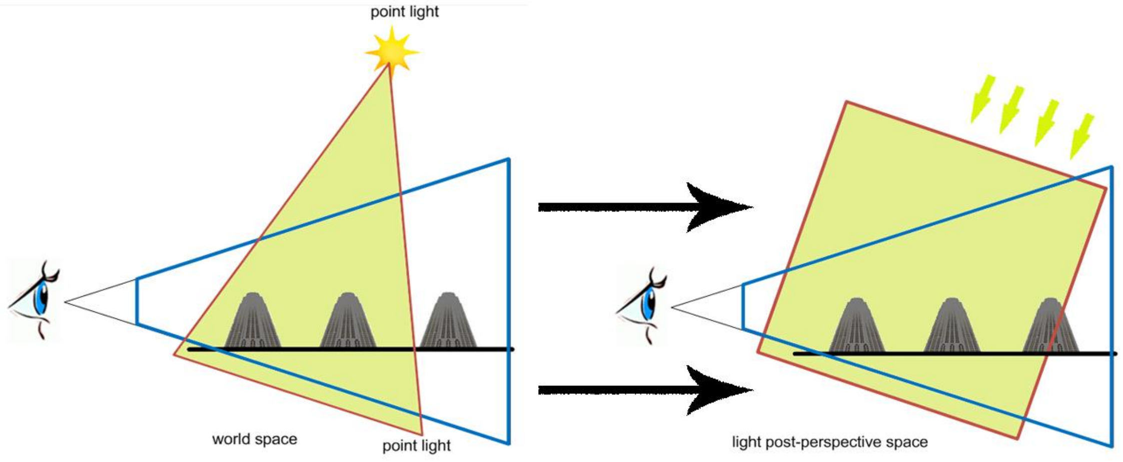

Figure 39: Although PSSM assumes directional light sources, point light sources can be converted once in light's post-perspective space

3.3.1. Split Scheme Discussion

Based on the observation that for different depths layers need different texture resolutions, PSSM suggests the partition of the view-frustum in various splits. Although the shadow maps associated with each split can have the same resolution, each split will have a different depth, according to the distance to the camera.

Consider C*i* as the depth of the i^{th} split plane in camera space and

$1 \leq i \leq m-1$, where *m* is the number of split parts. Let *n* and *f* be the near and far planes of the view frustum.

Figure 40 shows how the split planes are distributed along the z axis.

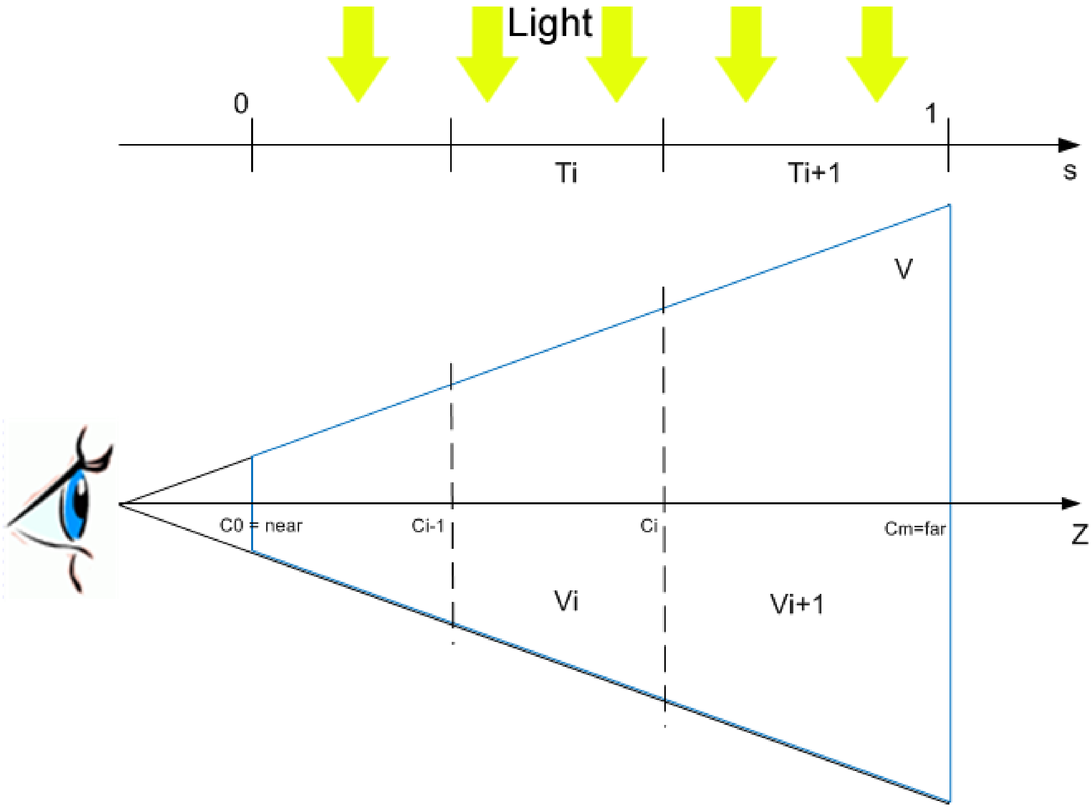

Figure 40: Split planes distribution along the view frustum z axis

In order to estipulate a rule for this distribution, PSSM present three split-schemes: uniform split scheme, logarithmic split scheme and practical split scheme.

Uniform split scheme proposes the following formula for the depth of split planes:

$$C_i^{uniform} = n + \frac{(f - n)i}{m}$$

This is the simplest split scheme, placing the split planes uniformly along the z axis. The aliasing distribution in this split scheme is same to that of standard shadow maps. Below, Figure 41 (left) shows the under-sampling for near points and over sampling for distant points. As a view-driven resolution, points near the camera need to have a more dense distribution, which turns this split scheme impractical.

Logarithmic split scheme proposes the following formula for the depth of split planes:

$$C_i^{\log} = n\left(\frac{f}{n}\right)^{\frac{i}{m}}$$

It produces a logarithmic distribution of perspective aliasing errors. In practice, this scheme is not suitable due to the scene’s potential intrinsic complexity. Figure 41 (middle) shows the over-sampling for near points and under-sampling for further points which is supposed to occur using this split scheme. Also, since this split scheme produces small lengths for split parts near the viewer, few objects can be included in these split parts.

Logarithmic split scheme behaves inversely to the uniform split scheme; a balanced algorithm would take into account both of them, proposing a moderate sampling of the points in the shadow map with a better aliasing distribution.

In light to this requirement, PSSM presents the practical split scheme, which is expressed as the following formula:

$$C_i = \frac{C_i^{\log} + C_i^{uniform}}{2} + b\,,$$

with *b* being a non-negative bias for adjusting the clip positions according to each application requirement.

Practical split scheme is then given by changing the variables:

$$C_i = \frac{n\left(\frac{f}{n}\right)^{\frac{i}{m}} + n + \frac{(f-n)i}{m}}{2} + b, \forall 0 \le i \le m$$

According to PSSM, unlike the other split schemes, this one satisfies the requirements of most practical applications (see Figure 41 – right).

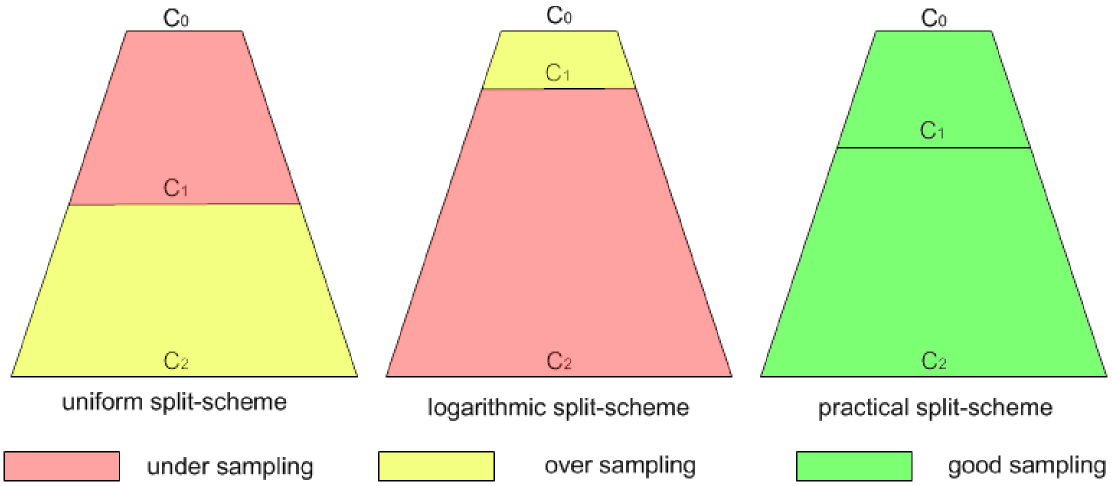

Figure 41: Split schemes presented by Parallel-Split Shadow Maps

3.3.2. Light frustum split

Since view-frustum is split into planes disposed along the distance to the camera, also light (as mentioned before: directional light sources) will be split accordingly.

Let V and W be the global view and light frustum respectively. Let also Vi and Wi be the ith split part of V and W respectively, with $(1 \le i \le m)$.

PSSM splits W into Wi smaller light frustums, each one being constructed with the help of a bounding box that intersects W with the current split. Thus, for each split:

$Wi = Vi \cap W$

Where W*i* should cover V*i* and objects casting shadows into V*i*.

Figure 42 demonstrates the process: B and B*i* refer to the bounding box of V and V*i* respectively. In order to maximize shadow maps resolution, W and W*i* will focus in the region of B*i* and B respectively.

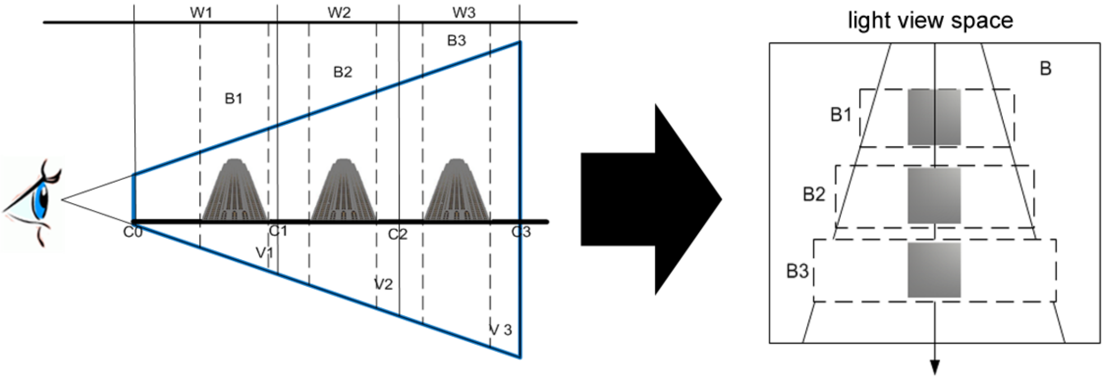

Figure 42: Light’s frustum split (W*i*) computation according to the respective view split frustum V*i* and its bounding box B*i*

This “greedy” adaptive light frustum proposed by PSSM makes the shadow map focusing only on the relevant object for each split, as opposed to a global adaptive light frustum where the focus would be whole scene V. As there may be non-interesting areas between Vi in V, implementing this adaptive light frustum per split will gain resolution for each split’s shadow map. However, this greedy adaptive frustum approach reintroduces the continuity problem.

3.3.3. Implementation

As a Standard Shadow Maps based algorithm, Parallel-Split Shadow Maps is also processed in 2 steps: the first is the shadow mapping generation while the second one consists of rendering the scene with shadow map comparison.

Thus, in order to implement PSSM, it is needed to:

- Split the view frustum into multiple depth parts, according to the practical split-scheme described above;
- Split the light frustum into multiple smaller ones, according to the light frustum split process described above;
- For each split V*i*, render it to the respective shadow map denoted as T*i*, in the space W*i* (Bi in case of using the PSSM adaptive light frustum technique);
- Render the scene and apply the correct shadow map for each pixel.

There are two options for rendering the scene: using the fixed-function pipeline or through a programmable pipeline approach. In case of using a fixed-function pipeline, for each split, execute the standard shadow map comparison. In case of using the programmable pipeline, while rendering the scene, current fragment's depth needs to be compared with the stored depth in the respective shadow map T*i*. In order to do so, one needs to pick the correct shadow map by comparing the pixel's depth with each of the splits ranges. Having the respective shadow map, the pixel should be transformed to its respective light space Wi (Bi in case of using the PSSM adaptive light frustum technique) and compared to the stored depth determining the shadow status.

3.3.4. Discussion

According to results presented by PSSM, with a number of splits less or equal to 4, the algorithm shows visual effects far better than with a single split of greater resolution.

As to the algorithm's performance, it will greatly depend on the hardware: the use of multiple shadow maps, requires multiple passes, (one per split), unless the GPU supports multiple rendering targets (MRT): using MRT, enables the generation of multiple shadow maps in a single pass.

As to the scene rendering, multiple rendering passes are also needed by default (one per split). If the GPU has a programmable pipeline, a single pass will also be possible by determining the appropriated shadow map for each rasterized fragment on-screen.

The majority of current hardware already provides GPU programmable pipelines, which makes this technique more appealing, also for being integrated with other complementary shadow mapping techniques.

Also, PSSM does not solve the noticeable discontinuity present in shadow maps' split transition. This discontinuity happens due to the different distribution of the depth between splits.

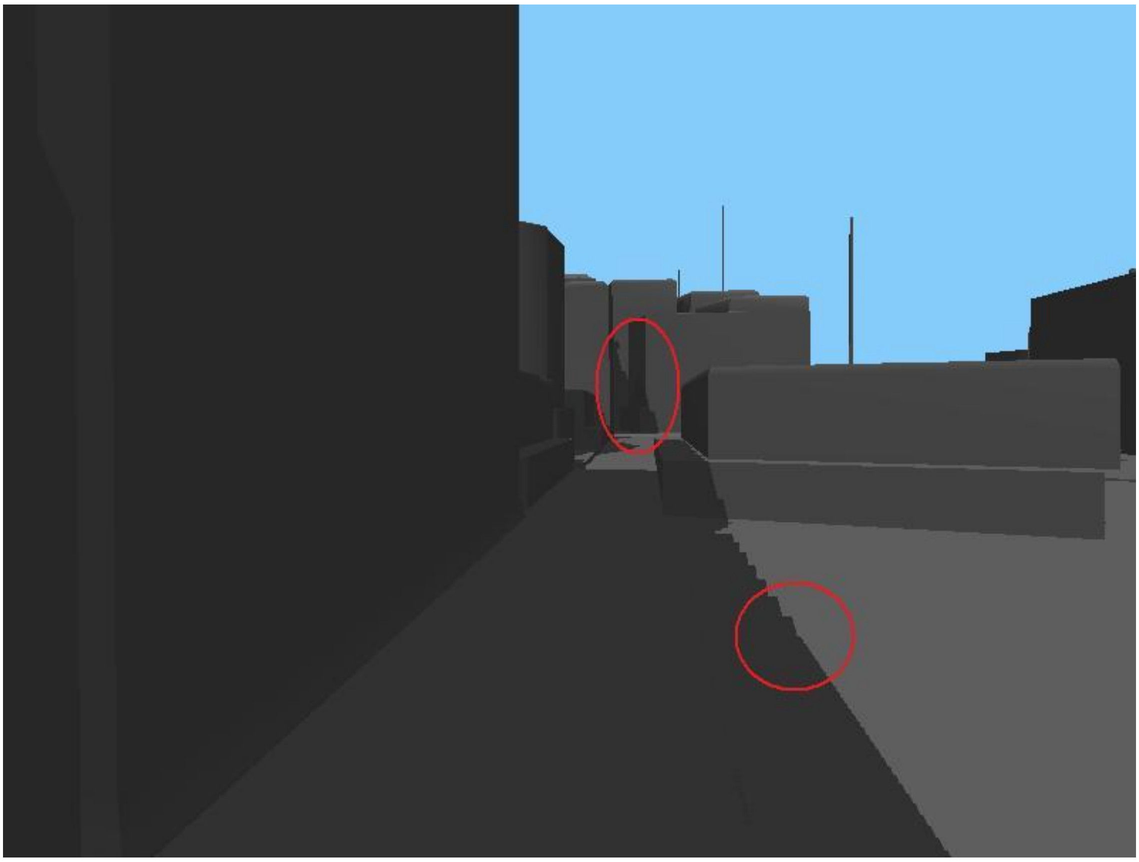

Figure 43: Split discontinuity problem

As Figure 43 shows, PSSM doesn’t handle well the transition between the splits, by not smoothing the areas where shadow maps are put together.

3.4. Variance Shadow Mapping

A shadow map is basically a texture and, as any texture, it may suffer from aliasing. Although we’ve been referring to algorithms that reduce aliasing through post-perspective transformations and scene partitioning, we now refer to a different, complementary approach that reduces aliasing by filtering the shadow map as a texture.

Unlike a normal texture, shadow maps cannot use the existing hardware built-in methods to reduce aliasing on color textures (e.g.: mipmapping, anisotropic filtering).

Also, shadow maps (depending on the distance between the query point and the light source) will change at each frame, turning pre-filtering for use of these techniques impossible.

Shadow map filtering approaches relied on nearest neighbor sampling or on taking multiple shadow samples and averaging them together (this former case is the one of PCF). Unfortunately, none of these approaches is sufficient to eliminate aliasing (even the PCF approach, despite being supported by current hardware, uses a small kernel for efficiency reasons).

Variance Shadow Mapping (VSM) addresses the limitations of PCF and proposes shadow map filtering using an upper bound approximation of the results given by the PCF algorithm.

Until here, each texel of a shadow map represented the depth of a single point. VSM presents a way of representing a distribution of depths for each shadow map texel. In order to do so, this approach stores the mean depth and mean squared depth, also known as the first and second moments respectively.

The moments are needed for computing the bound of the distribution being shaded. They also provide an approximation for the amount of light reaching the given surface.

As opposed to depths, moments can be interpolated. This means that, by using moments, hardware built-in methods such as mipmapping and anisotropic filtering can now be used for reducing aliasing, delivering improved quality shadows with low performance penalty.

3.4.1. Implementation

Being based on standard shadow mapping algorithm, VSM is composed of its two standard steps, plus some intermediate steps:

First, at shadow map generation step, the scene is transformed to post-perspective space of the light. Instead of generating a standard shadow map, this space is rendered to a two-channel buffer, where the first channel receives the depth (as in standard shadow mapping) and the second channel receives the square of that same depth.

Then, mipmaps generation is set, in order to facilitate filtering by hardware. The outcome of this operation is the moments M1 and M2.

Having M1 and M2, the mean μ and variance σ^2 are calculated as follows:

$$\mu = E(x) = M_1$$

$$\sigma^2 = E(x^2) - E(x)^2 = M_2 - M_1^2$$

where $E(x)$ and $E(x^2)$ derive from moments M1 and M2, which are the result of hardware filtering applied to the two-channel buffer carrying the depth and square depth respectively.

Finally, while rendering the scene from the camera point of view, for each fragment, its depth should be compared with μ :

If depth < μ m then the surface is unshadowed;

Otherwise, the surface is shadowed with intensity $p_{\max}$, which is the equation for Chebyshev's inequality:

$$p_{\max}(t) \equiv \frac{\sigma^2}{\sigma^2 + (t-\mu)^2}$$

3.4.2. Discussion

Light Leaking Artifacts Origin

Variance σ^2 is seen as a measure of the width of a distribution. Hence, it should place a bound on how much of the distribution can be concentrated far away from the mean. This bound is stated precisely in Chebyshev's Inequality, as presented by VSM:

Let x be a random variable drawn from a distribution with mean μ and variance σ^2. Then for t > μ :

$$P(x \geq t) \leq p_{\max}(t) \equiv \frac{\sigma^2}{\sigma^2 + (t-\mu)^2}$$

As presented in VSM, the quantity $P(x \geq t)$ in Chebyshev's Inequality is exactly the quantity needed in order to perform PCF, since it represents the fraction of pixels over a filter region that will fail the depth comparison with a fixed depth t. For more details on how this upper bound is enough to provide a good approximation to PCF, please refer to [VSM06].

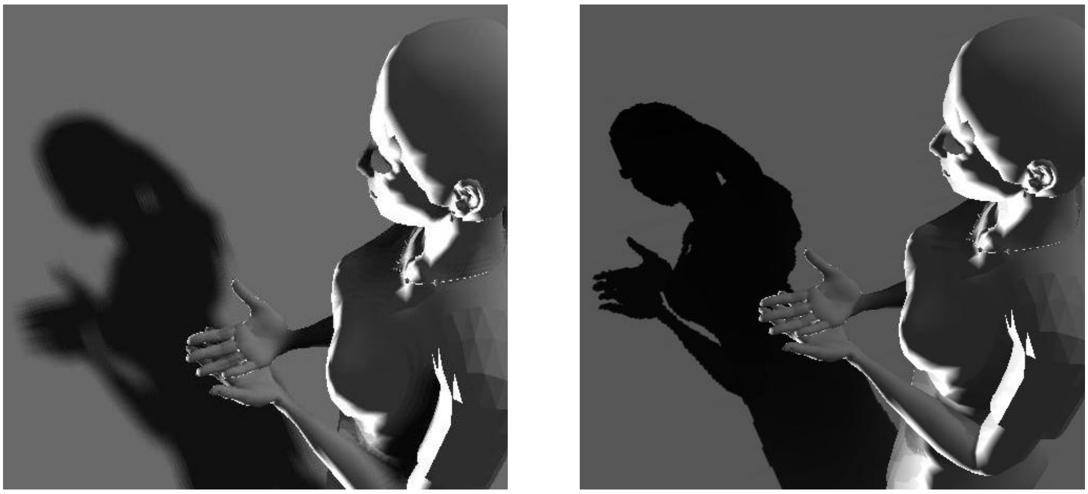

Figure 44: Scene rendered with a 512 shadow map: (left) PCF 4x4 vs. (right) VSM

Although being clearly superior to PCF (as shown in Figure 44), the VSM algorithm still suffers from light bleeding (also known as light leakage) artifacts: When variance σ^2 is close or equal to zero, these noticeable artifacts can appear over the shadow (as seen in Figure 45, left image).

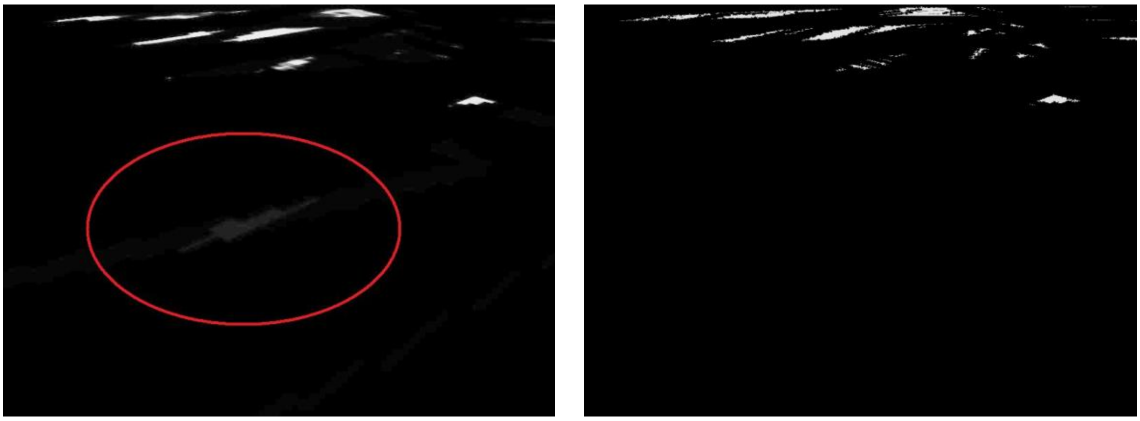

Figure 45: Scene shadow rendered with VSM suffers of light bleeding (left). The same scene shadow is rendered using Standard Shadow Mapping (right). Note that both images have their contrast levels increased so that artifacts are more perceivable.

In practice, it happens only for scenes with high depth complexity.

3.5. Conclusion

In this chapter we gave a brief overview of algorithms that contributed for alleviating shadow mapping artifacts. Then, we have presented the most relevant algorithms to our investigation:

Perspective Shadow Mapping is the first algorithm addressing the aliasing problem for objects near the camera by introducing a non-uniform parameterization, working in the post-perspective space. This approach improves resolution for some cases, depending on the position of the light relatively to the camera, but its implementation is rather non-trivial with many tradeoffs that are hard to optimize, compromising the use of this algorithm to interactive environments (where the user has the camera control and/or light changes dynamically). Also, the 3D convex hull needed to determine the region of interest requires a robust implementation, together with intersection and union operations and a successive set of approximations for tightening this same polygon.

Another unwanted characteristic of this approach is the continuity problem, due to the adaptive light frustum proposed by PSM. The convex hull can change suddenly, as the scene changes dynamically, causing noticeable changes in the resolution of the shadow map. Also, this approach may need to virtually move eye position (to avoid the inverted order of objects due to perspective projection), among other virtual modifications dependent of scene configuration. These abrupt changes on the convex hull, again will introduce severe continuity problems in shadows.

Polygon offset problem is worsened: since PSM transforms the post-perspective space in a non-linear way, depth values will also change differently, so the basic solution of adding a constant bias for removing the surface acne effects may not be enough.

Finally, although perspective correction enables a major improvement to shadow map resolution as seen from the camera, it may not be enough when representing large scenes. Although PSM has been very criticized due to its shortcomings, it was the first approach to decrease aliasing through perspective non-uniform parameterizations, being the basis for Light Space Perspective Shadow Mapping and the inspiration for Trapezoidal Shadow Mapping.

Trapezoidal Shadow Mapping proposes a different non-uniform parameterization, by warping a trapezoid to achieve the shadow map squared shape. This trapezoid is the post-perspective approximation of the view-frustum, as seen from the light. TSM reduces the shadow map resolution problem by using the transformation matrix generated by the trapezoid warping. This calculation may be computationally expensive for some scene configurations, but the algorithm handles all the cases more gracefully than PSM. Similarly to PSM, TSM also suffers from the polygon offset problem, but in this case, it is provided a very straight-forward method for solving the problem, by maintaining the depth values and transforming only the x and y coordinates of the polygons (avoiding the non-uniform parameterization of the depth).

TSM is an adaptive light frustum based technique, but it is not prone to the continuity problem, since it successfully avoids the problem while constructing the trapezoid (more specifically, the base and top lines of the trapezoid being parallel), assuring a smooth transition when the eye moves relative to the light from frame to frame.

Also, the problem of over-sampling near the camera and under-sampling at distance is solved, by the 80% rule introduced while calculating the trapezoid side lines, so that there is an effective use of the available shadow map resolution along the scene distance.

As to the dueling frusta case, both PSM and TSM cannot respond effectively, falling back to the standard shadow mapping algorithm.

Although TSM greatly improves the shadow map resolution, similarly to PSM it may not be enough to address the resolution problem for large scenes, especially with the fallbacks to standard shadow mapping (associated with, for instance, the dueling frusta case).

Parallel Split Shadow Mapping is especially useful for large scenes. The approach proposes the scene partitioning into several splits distributed along camera distance and for each split, the use of different shadow maps.

Each split will have a determined size and position, according to a practical split scheme that ensures smaller split sizes near the camera and bigger split sizes distant to the camera.

Having said that, it is easily noticed that PSSM addresses the same problems as PSM and TSM, but instead of a non-uniform parameterization at light post-perspective space, PSSM uses multiple discrete layers.

Large scenes shadow mapping is greatly improved by introducing multiple shadow maps instead of a large one. Also, the memory requirements are less demanding than a single shadow map of equivalent dimensions.

Also, PSSM does not propose a solution to the noticeable discontinuity present in shadow maps' split transition.

PSSM does not solve the perspective aliasing, focusing only on the texture resolution constraints associated to large scenes, being seen as a complementary approach that is often used for large scenes. Since each split is exclusively related to a shadow map, other shadow mapping techniques can be seamlessly integrated into this split scheme.

Variance Shadow Mapping is a simple and effective filtering approach for addressing the aliasing in shadow maps. As opposed to the previously presented algorithms, it does not introduce any parameterization, simply working at the shadow mapping filtering (as with PCF). This algorithm performs an upper bound on the result of the percentage closer filtering algorithm, providing a close approximation to it and at the same time taking full advantage of graphics' hardware built-in filtering techniques, leading to improved results in performance and shadows quality.

As with other methods, this technique only addresses a part of the limitations of the standard shadow maps, namely focusing only on reducing the binary status of hard-shadows through shadow map filtering. This can therefore be a complementary technique to other algorithms that address different problems of the standard shadow mapping algorithm.

Although being a very straight-forward technique, VSM is a probabilistic approach, and as any probability, it is prone to unsatisfactory results. In this case, since the probability is based on an upper bound of the estimation whether a point is in shadow or not (despite providing fast results to a technique that is time-consuming), it produces the referred light leaking effects in scenes with high depth complexity relative to the light source.

As a concluding remark, from the presented techniques, none of them solves all the problems associated with Shadow Mapping standard algorithm. Also, there are techniques that while addressing a specific problem introduce another. Table 1 shows a summary of the problems solved by each one of the presented algorithms.

Algorithm	Perspective Aliasing	Projection Aliasing	Texture Resolution	Self-Shadowing	Continuity Problem	Hard Shadows Binary Status
SSM	✗	✗	✗	✗	(N.A.)	✗
PSM	✓	✗	✗	✗	✗	✗
TSM	✓	✗	✗	✓	✓	✗
PSSM	✗	✗	✓	✓	✗	✗
VSM	✗	✗	✗	✓	✓	✓

Table 1: Problems present (red) vs. problems tackled (green) for each of the presented techniques. Note: (N.A.) means the current algorithm does not suffer from the referred artifact

Analyzing the table, we conclude that there is no perfect technique; instead there are complementary techniques which combined may form a more robust algorithm able to solve the majority of the problems inherent to the shadow mapping technique.

Also, projection aliasing is a problem that is not addressed by any of the mentioned techniques. Some approaches suggested solutions for alleviating the projection aliasing problem, namely LiSPSM suggests combining surface blurring and a phong lighting model for “disguising” projection aliasing. Also, Adaptive Shadow Maps suggests a view-dependent hierarchical grid structure able to improve determined regions shadow resolution as the ones with projection aliasing (which is a signal of lack of resolution). Nevertheless, this artifact hasn’t yet been addressed effectively by any of the shadow mapping analyzed solutions.

4. Analysis and Proposal of a Shadow Mapping Remix Approach

In this chapter we propose the *Shadow Mapping Remix Approach* (SMRA). We start by identifying the strengths of the algorithms presented in Chapter 3. Afterwards, we describe our algorithm's implementation where we start by outlining its setup, and then we present our approach implementation steps.

Then, we address to one artifact that is not addressed by any of the contributors to our new algorithm proposal and, for that reason, it needs to be described in more detail than all the other implementation steps.

Next, follows a discussion about minor changes to the contributors' algorithm techniques and also performance estimation for our approach.

Finally, a conclusion will summarize the findings of this chapter.

4.1. Analysis - Contributors' Strongest Features

Our work builds on the analysis of the most important contributions that improve standard shadow mapping and proposes a remix approach that merges the strongest points of each algorithm. In this section, we will focus on the most relevant features of each algorithm.

In Chapter 3 we have presented 4 algorithms which we consider as being the main contributions to our work. From a theoretical point of view, perhaps the most important contribution to the shadow mapping algorithm is Perspective Shadow Mapping, because it is the first to propose a transformation to tackle the aliasing problem. However, its implementation is rich in particular cases, which makes it less attractive than other newer approaches. That’s why we will not extract any specific feature from PSM for our implementation.

Trapezoidal Shadow Mapping, an algorithm following PSM algorithm, offers a solution for the same issues without the shortcomings of PSM. As mentioned in the previous chapter, TSM proposes a non-uniform parameterization, by warping a trapezoid to achieve the shadow map squared shape. This is a very straightforward technique for reducing two of the main shadow mapping artifacts: perspective aliasing and the continuity problem introduced by greedy adaptive light frustum techniques. Also, it proposes an efficient way to alleviate the polygon offset problem worsened due to the non-uniform parameterization.

Parallel-Split Shadow Mapping is another algorithm we find as a big contribution to shadow mapping, mostly for large scenes, where the shadow map resolution is a critical factor for the aliasing. In order to assure enough resolution for the scene, almost independently of its dimension, PSSM proposes scene partitioning into several splits distributed along camera distance. For each split a shadow map would capture its depth values and the same shadow map would later be used for shadow determination. Basically, it extends the standard shadow mapping to multiple instances, each of the being applied to a different part of the scene. Although it may be expensive in terms of performance, it helps to solve another shadow mapping artifact - the texture resolution constraints.

Finally, Variance Shadow Mapping is a shadow mapping variant for generating filtered shadows. It relies in the basic shadow mapping algorithm, not requiring post-perspective transformations or other mechanisms other than texture filtering. VSM performs an upper bound on the result of the percentage closer filtering algorithm, while taking full advantage of graphics’ hardware built-in filtering techniques. This will alleviate another shadow mapping artifact, which is the hard shadows binary status, with performance gains (when compared to the basic shadow filtering approach: PCF).

4.2. Remix Approach Proposal

Having highlighted the most relevant features for building our algorithm, we introduce our Shadow Mapping Remix Approach.

The main idea is to merge the strongest features of TSM with PSSM and VSM, which happen to be complementary.

PSSM is the only proposal that addresses the resolution problem in a way which is suitable both for indoor and outdoor large scale scenes. The frustum splitting strategy will be a component of the remix. The greedy adaptive frustum approach proposed in PSSM is going to be discarded because it introduces the continuity problem.

TSM proposal to transform the light frustum into a trapezoidal shape will be used for each split. The nature of this algorithm provides full usage of the shadow map resolution; hence it minimizes the resolution problem without the continuity problems of greedy adaptive frustum approaches such as the one used in the original PSSM approach. The transformation of the frustum proposed by TSM will be a component of the remix.

VSM offer the final touch by filtering the shadows, helping to alleviate the aliasing problem, both perspective and projective aliasing. In the remix, VSM will be used to compute the data for each split, after transformed into a trapezoidal shape, and perform the depth comparison.

As previous shadow mapping techniques, SMRA is a two pass approach. In the first step the following steps are executed:

- The view frustum is split according to the practical scheme proposed in PSSM;
- For each split, the trapezoidal transformation is computer (as proposed in TSM);
- For each pixel in the shadow map, the depth and square depth will be stored (according to the 1st step of VSM)

In the second pass, for each split, once again the TSM transformation is required to compute the appropriate texture coordinates to the shadow map. Then, VSM formula to compute the shadow status will be used.

4.2.1. The Detailed Algorithm

The algorithm is done according to the following steps:

1. First of all, the view frustum is split into multiple depth parts V*i*, according the practical split-scheme (refer to Section 3, Parallel Split Shadow Mapping);
2. Then, comes the generation of the shadow map: For each split V*i*:
    a. Approximate current split's eye frustum V*i* as seen from the light with a trapezoid to warp it onto a shadow map:
    b. Transform the eye's frustum 8 vertices into light point of view;
    c. Construct the trapezoidal approximation based on these 8 vertices;
        i. Obtain the base and top lines of the trapezoid (refer to Section 3, Trapezoidal Shadow Mapping);
        ii. Obtain the side lines, according to the 80% rule, (refer to Section 3, Trapezoidal Shadow Mapping);
        iii. Generate the trapezoidal transformation matrix, which maps the four corners of the constructed trapezoid to the side face of a unit cube - a square (refer to Section 3, Trapezoidal Shadow Mapping);

    d. Transform the scene split into post-perspective space of the light W*i* (refer to Section 3, Perspective Shadow Mapping);
    e. Assign current depth value to a temporary texture coordinate;
    f. Transform the light post-perspective space W*i* into the normalized N-space, by applying the trapezoidal transformation N*i*;
    g. In the fragment shader, replace the transformed fragment depth by the depth stored in the temporary texture coordinate;
    h. Render the transformed scene split N*i* to a 2-channel buffer T*i*, where the first channel receives the depth (as in standard shadow mapping) and the second channel receives the square of that same depth;

- Before the 2nd pass begins, mipmaps generation is enabled for facilitating shadow maps filtering by hardware;

3. Finally, in the 2nd pass, for each scene split, for each fragment p:
    a. At vertex shader, transform p to the post-perspective space of the camera p';
    b. Assign current depth value to a temporary texture coordinate;
    c. Transform p' into the N-space, generating p'';
    d. In fragment shader, replace the transformed fragment p'' depth by the one stored in the temporary texture coordinate (p' depth);
    e. Get the mean μ and variance σ^2 for the moments resultant from the interpolated, transformed shadow map;
    f. Compare the depth of p'' with μ:
        i. If depth < μ then the surface is unshadowed;
        ii. Otherwise, the surface is shadowed with intensity $p_{\max}$ (refer to Section 3, Variance Shadow Mapping).

4.2.2. Artifacts Analysis

From the analysis of our approach, we can identify one artifact that hasn't yet been addressed, which is the projection aliasing problem.

Improving Projection Aliasing

As mentioned before, projection aliasing results of insufficient detail about a determined surface depth. Projection aliasing is only dependent on the angle between the light vector and the surface normal. Thus, when this angle approaches 90º, the maximum projection aliasing occurs.

We propose a very simple heuristic to alleviate extreme projection aliasing: In the second shadow mapping pass (rendering the scene split), execute the following steps (integrated in the aforementioned algorithm):

- Calculate fragment normal;
- Calculate the angle α formed between the normal and light direction;
- Before comparing the depths, if the angle α is close to 90º (say, $85º \le \alpha \le 90º$), then we assume the scene is fully shadowed with half of maximum intensity, suppressing the steps 3.e and 3.f of the algorithm's 2nd pass.

Our approach therefore assumes that surfaces close to orthogonal to the light never gather enough light, resulting in a deliberate self-shadowing. It is a very lightweight technique, not involving any additional image processing; just a simple condition is introduced.

Figure 46 shows the projection aliasing removal technique effect. Most of the black stripes present in the buildings are transformed into a solid shadow.

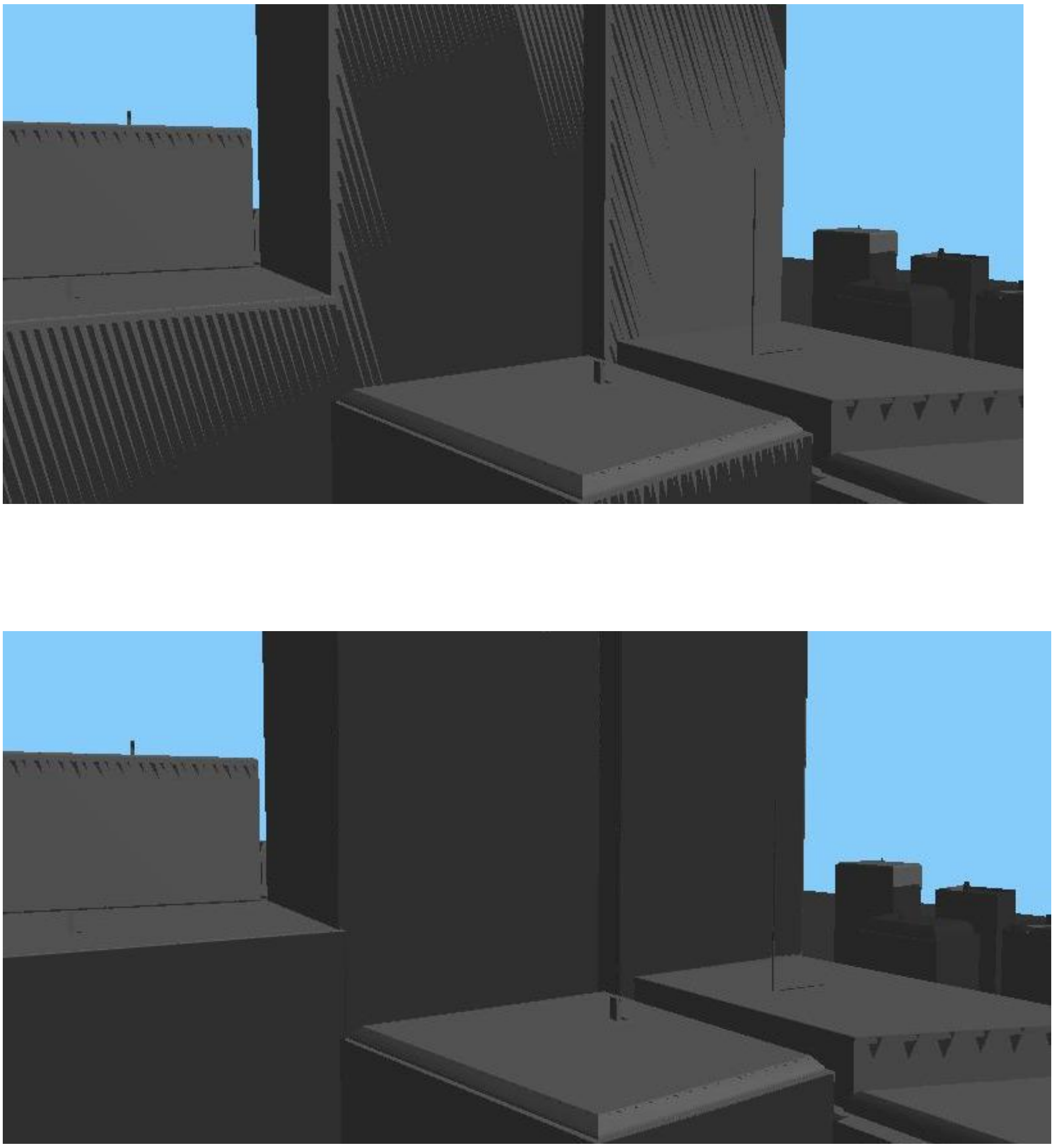

Figure 46: A scene with projection aliasing, right before the projection aliasing removal technique is applied (top) and after the technique is applied (bottom).

4.3. Discussion

After explaining the algorithm, follows a brief set of considerations.

4.3.1. Visualizing the implementation

As we claim, none of the contributions by itself is enough for addressing standard shadow mapping artifacts.

In this section we show the effect of all the contributions through renders of the same scene for each one of the contributions.

First, the scene is drawn with no shadows (Figure 47):

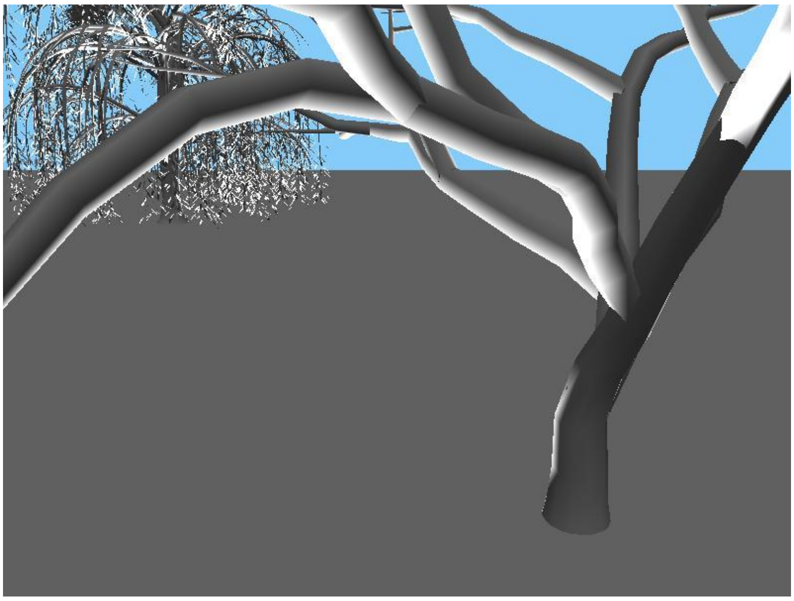

Figure 47: Scene rendered without shadows

As we stated before, a scene without shadows lack of realism – there is no perception of the geometric relationship between objects.

Using the standard shadow mapping algorithm for shadow generation, this geometric relationship between objects is now present (Figure 48):

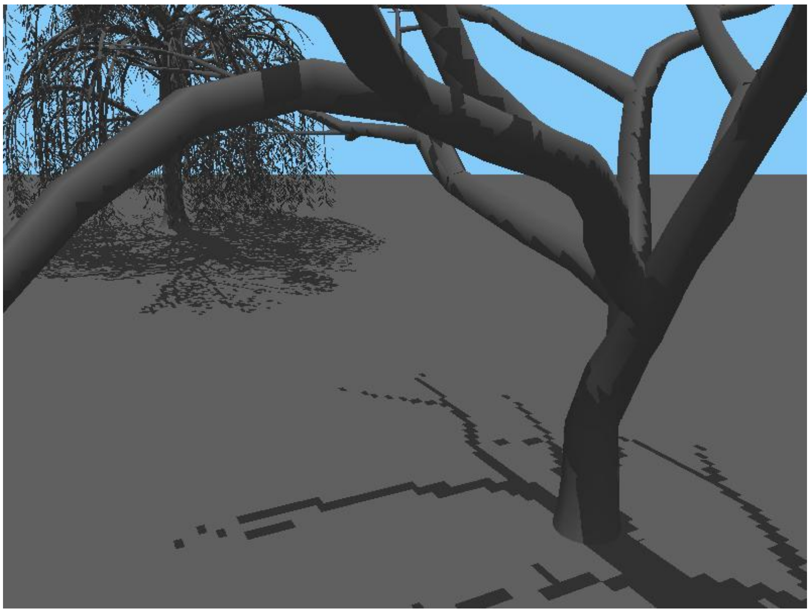

Figure 48: Scene rendered with SSM 512x512

Although, the shadow mapping algorithm suffers of too many problems (mainly texture resolution constraints and aliasing).

As we introduce TSM, the aliasing is greatly reduced thanks to the trapezoidal approximation of the view frustum (as seen in Figure 49).

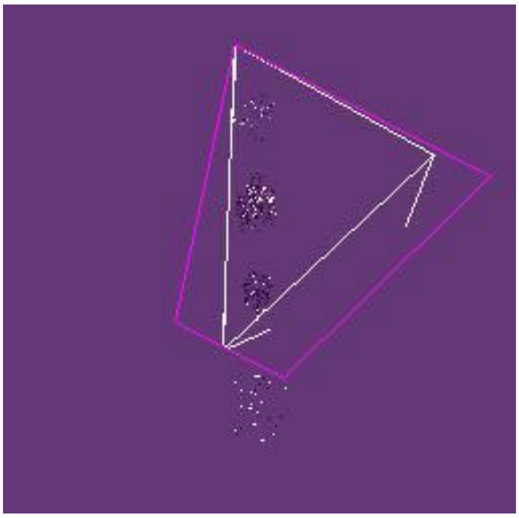

Figure 49: Trapezoidal approximation of the view frustum, as seen from the light

Figure 49 shows the scene rendered from the light point of view. It is noticeable the view-frustum (in white) and the trapezoidal approximation polygon (in violet).

This trapezoid is then warped so that it provides more detail for nearby objects and less detail for far-away objects. Figure 50 shows the difference between a shadow map generated using TSM (non-uniform parameterization) and the same shadow map generated using a uniform parameterization (as in SSM).

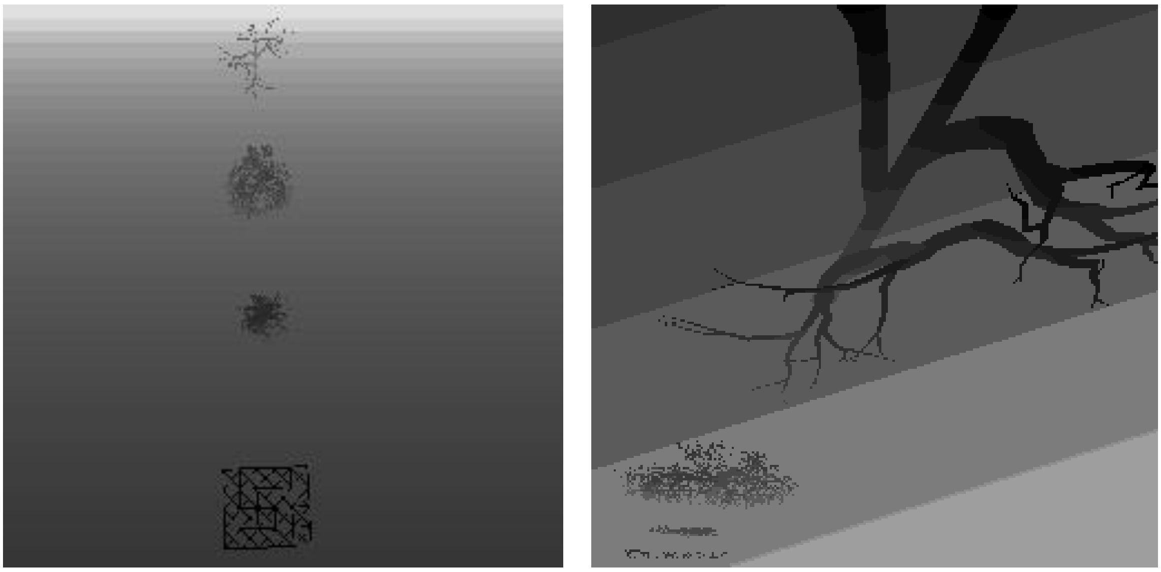

Figure 50: Shadow Map generated using SSM (left) and TSM (right)

As we can notice, the top of the TSM shadow map, which represents the nearby shadows, is enlarged and the bottom of the shadow map is shrunk, while the SSM shadow map is like a photo taken from the light position (the depth densities are uniformly distributed along the shadow map).

This trapezoidal transformation not only reduces perspective aliasing, but also avoids the continuity problem. However, this technique may not be enough, when addressing the shadow map resolution problem: large scenes may require more resolution, which leads to a fast degeneration of the shadow map resolution along the distance, when using only TSM (Figure 51). Although far better than the original SSM approach, there is aliasing in the shadows near to the camera.

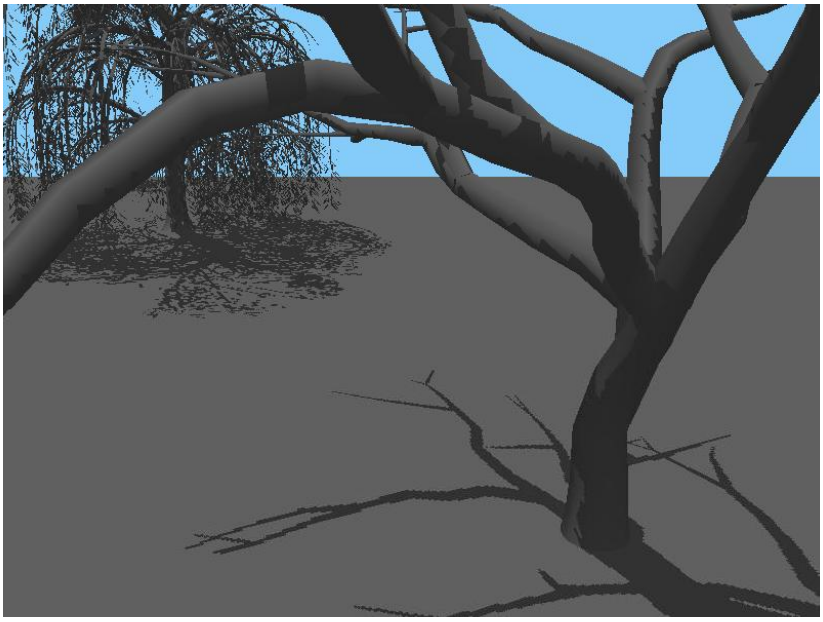

Figure 51: Scene rendered with TSM 512x512

In light of this problem, we use PSSM multiple shadow maps approach. For this scene, we render it using 3 splits (Figure 52), distributed along the distance to the camera. The splits are determined according to the practical split scheme described above.

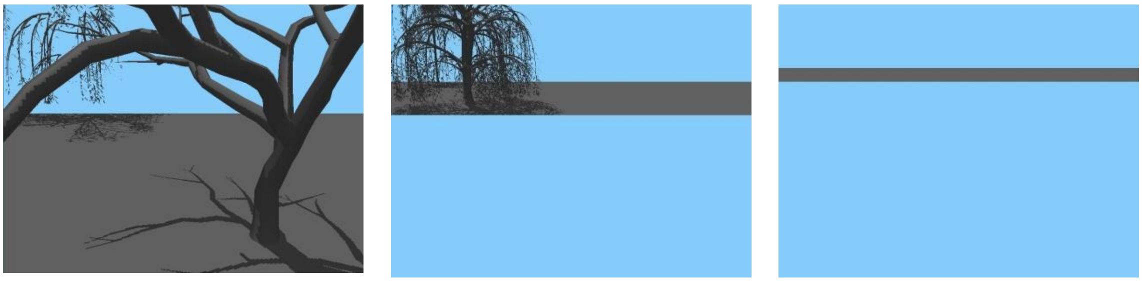

Figure 52: PSSM scene splits, as seen from the camera

The result of rendering the scene with PSSM is as follows (Figure 53):

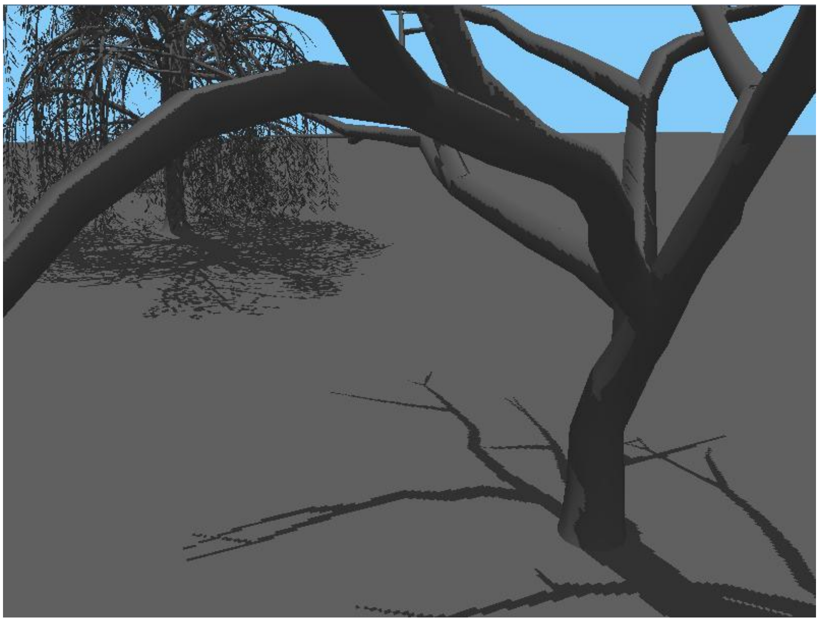

Figure 53: Scene rendered with PSSM 3x512x512

As it is noticeable, PSSM produces better shadows on the trunks compared to TSM, while having worse results on the floor. On the other hand it produces slightly better results with distant shadows. In order to improve the precision even more, we introduce TSM to each split of the PSSM. This way, not only we address aliasing and continuity problems (tackled by TSM), but also shadow map resolution problems (tackled by PSSM). Figure 54 shows the scene rendered with the mixture of PSSM and TSM:

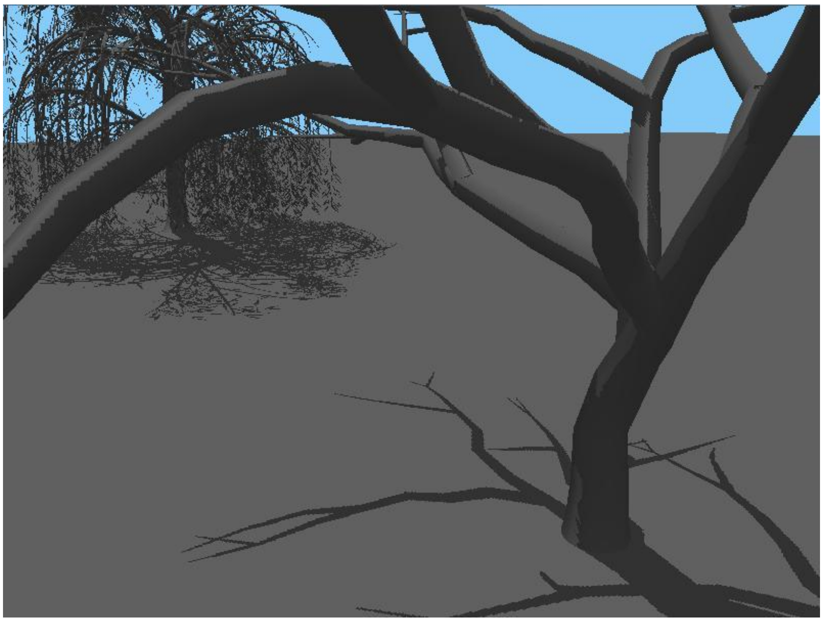

Figure 54: Scene rendered with PSSM and TSM 3x512x512

Finally, in order to produce smoother shadows, we use the filtering proposed by VSM. VSM by itself would only address the binary status of hard shadows, not addressing any of the above described artifacts (Figure 55):

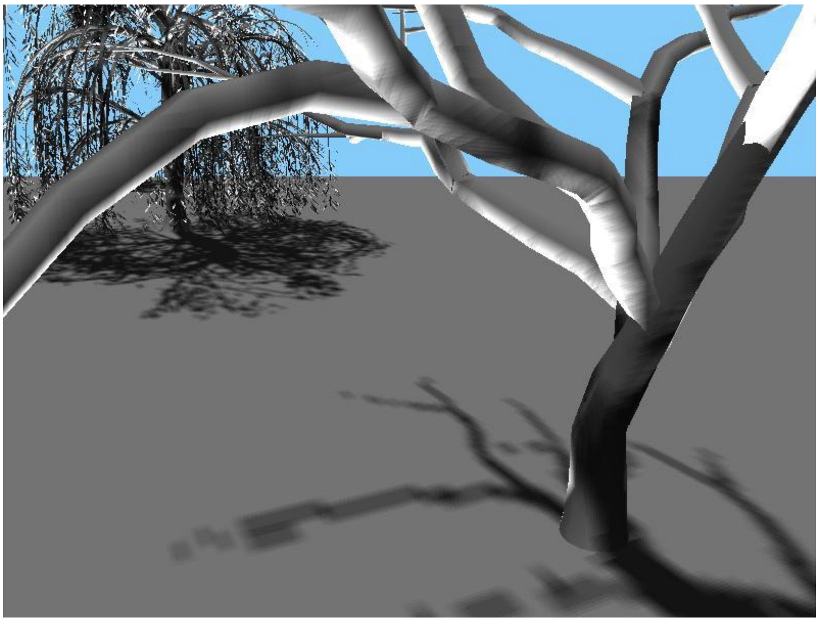

Figure 55: Scene rendered using VSM 512x512

This filtering technique makes all the sense when put together with TSM and PSSM, forming a complete and improved algorithm called Shadow Mapping Remix Approach (Figure 56):

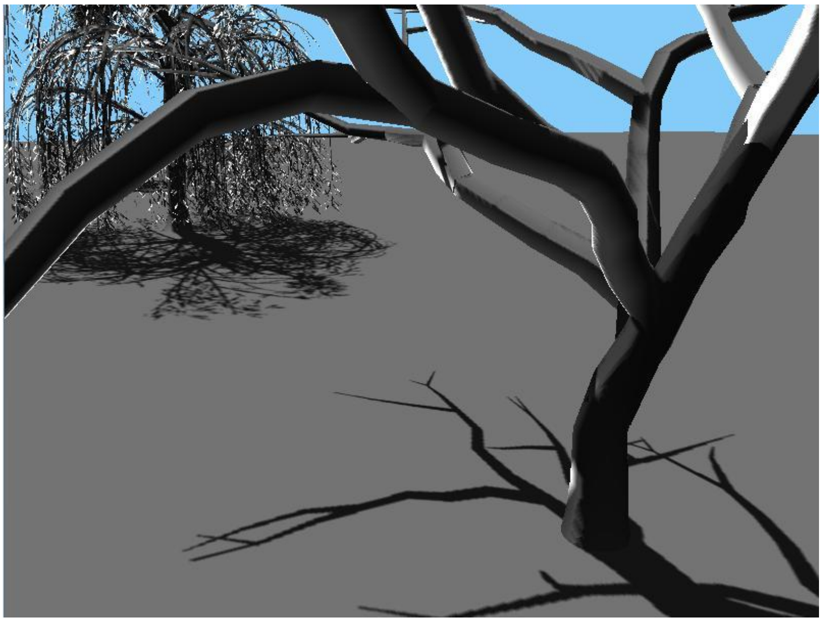

Figure 56: Scene rendered using SMRA 3x512x512

4.3.2. Performance estimation

Since SMRA is a mixture of many complementary techniques, its computational cost is higher than using each one of the contributors’ approaches by itself. However, as stated in next section, this speed penalty is compensated by the serious visual improvements, when compared with each of the presented techniques, still being perfectly applicable for real-time applications when using moderate shadow map resolutions.

4.4. Conclusion

In this section we have proposed an algorithm for improving visual results of shadow mapping. Shadow Mapping Remix Approach mixes different techniques proposed by three algorithms. Since these techniques are complementary, their interaction is simple to perform with minor changes to some of these techniques.

We have also presented a visualization of the implementation of SMRA, through the various steps of rendering a scene unshadowed, shadowed with standard shadow mapping, also with the main contributions' algorithms and finally with SMRA. The results show a clear visual improvement when combining the components.

5. Results

In this section we present the results of rendering shadows using the algorithms discussed above and also the presented our approach: Shadow Mapping Remix Approach. First, we will describe the configuration in which the tests were performed. Then, we present the visual results of each algorithm while varying the shadow maps resolutions. We also present and discuss the performance of each algorithm. Finally, we conclude by summarizing our findings.

5.1. The Setup

In order to test each one of the presented algorithms, we have created a platform called ShadowExplorer, capable of running each one of these approaches and change all shadows algorithms, including each algorithm specific properties, in real-time. The platform and each one of the algorithms were implemented using Visual C++ and OpenGL, under Microsoft Windows Vista.

We ran the tests using an Intel Core2Duo processor working at 1.8Ghz, with 3Gb RAM and a NVidia GeForce 8400M GT GPU with 128Mb of dedicated memory.

We used a viewport with resolution 800x600, varying shadow mapping resolutions. Camera and light positions were also user controlled, through the application.

We used two different scenes for gathering the results: the TreeLine (using 111.000 triangles) and the SparseCity (using 33.000 triangles).

Both scenes were rendered with a field of view of 45 degrees. As to the near and far plane, TreeLine had its camera view frustum set with 1 for near plane and 50 for far plane, while SparseCity configured camera view frustum near and far planes to 15 and 5000 respectively.

5.2. Visual Results

In this section, scenes will be rendered separately: First, TreeLine, which is the most demanding scene in terms of triangles; then, SparceCity which is the most demanding scene in terms of shadow resolution requirements.

Each scene will be rendered 3 times with different shadow map resolutions (512x512; 1024x1024; 2048x2048) per algorithm (SSM, TSM, PSSM, VSM and SMRA).

TreeLine

TreeLine is composed of 3 high-detailed trees disposed through a line, over the floor. This scene was designed for studying the detail capabilities of each shadow algorithm, more precisely on how shadows of leafs and small tree branches are projected to the ground, from a distant light).

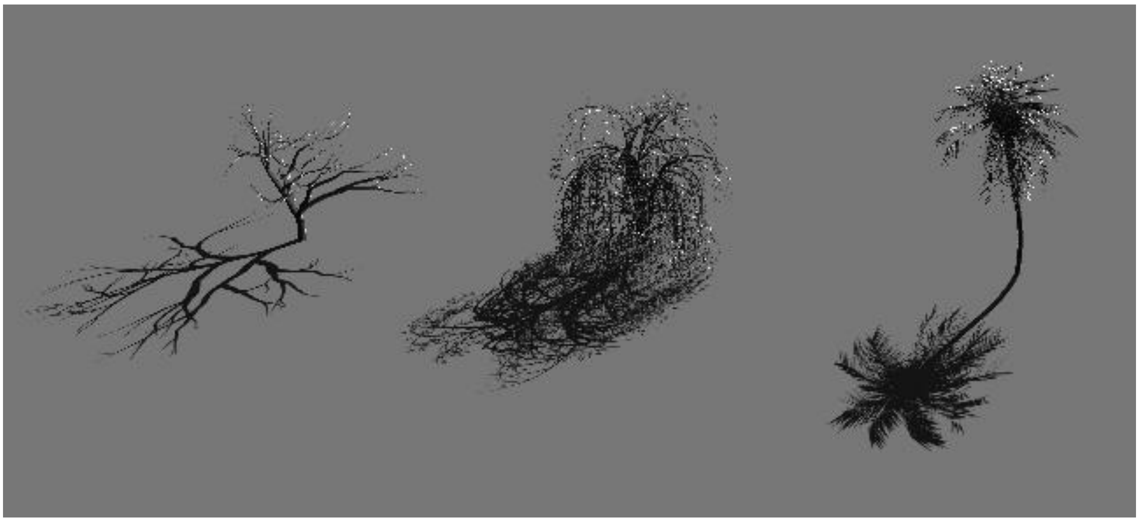

Figure 57: TreeLine rendered with SMRA

Standard Shadow Maps

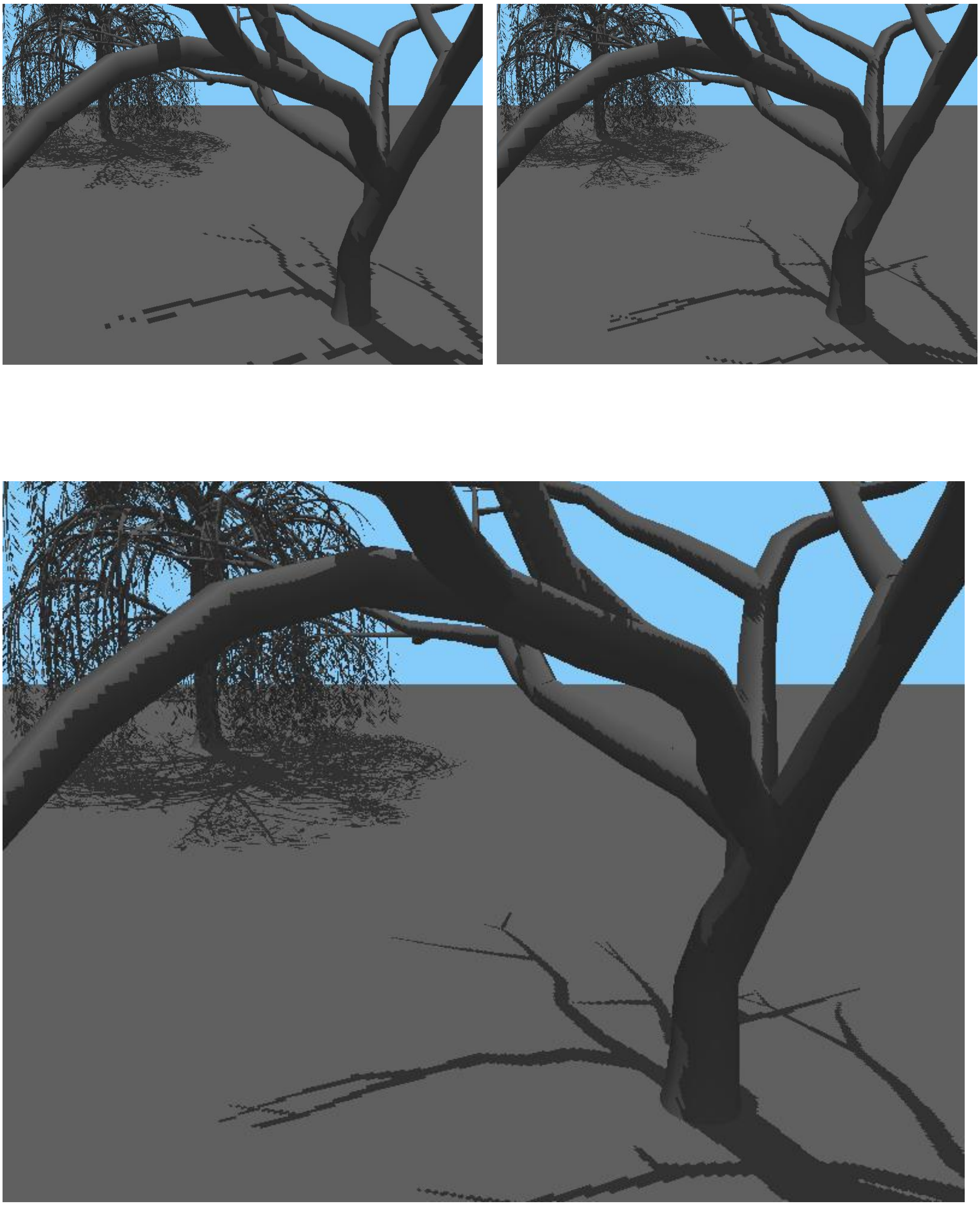

Figure 58: TreeLine scene being rendered with SSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom)

Figure 58 shows the scene rendered with standard shadow mapping. As it is noticeable, using low shadow map resolutions, the shadows become heavily pixelated. This is caused by insufficient shadow map resolution. In the bottom picture, the used shadow map has a 2048x2048 resolution and still, for a relatively small scene, pixelated areas are noticeable. This is due to the perspective aliasing problem already mentioned, where shadows become under sampled near the eye and over sampled as the distance to the camera increases (the far-away tree's shadow is over sampled, when compared with the shadows nearby the camera).

Figure 59 presents the result of the scene rendered with the TSM approach. This approach introduces a non-uniform depth distribution in the shadow map, alleviating the aliasing noticed in the previous approach. In this case, lack of shadow resolution is detected even for the largest small shadow map resolution in the trunks of the closest tree.

PSSM addresses the shadow map resolution constraints, but does not introduce any uneven distribution to the shadow map. Thus, if the resolution is still insufficient, then aliasing will be noticeable (it can be noticed in Figure 60, where the resolution is not sufficient for the current scene).

Figure 61 shows the scene rendered with VSM. This algorithm presents the same problems as in SSM, so when SSM fails massively, this algorithm will never succeed (since it only filters the shadow map, introducing a smooth effect to the shadow edges).

Trapezoidal Shadow Maps

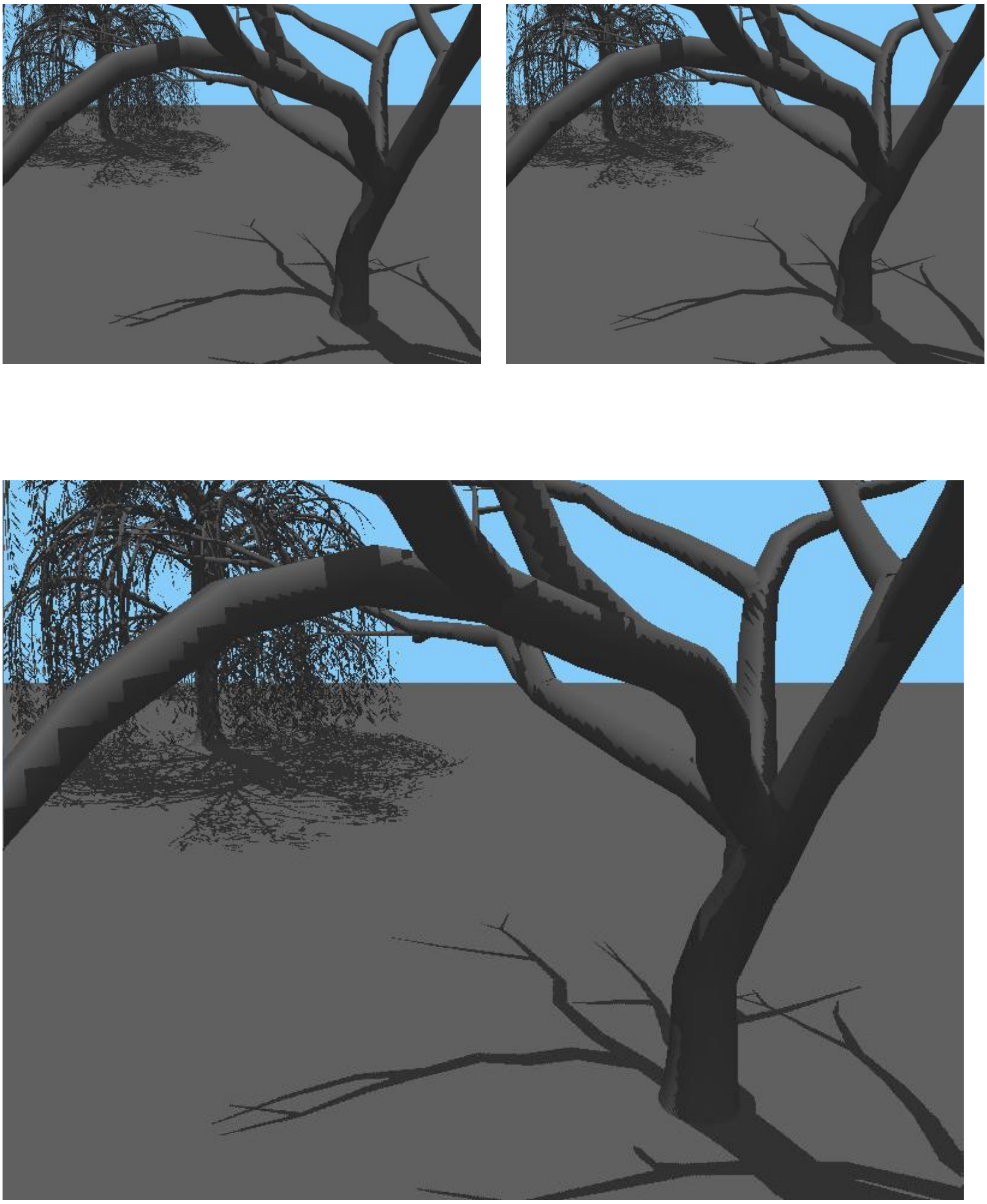

Figure 59: TreeLine scene being rendered with TSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom)

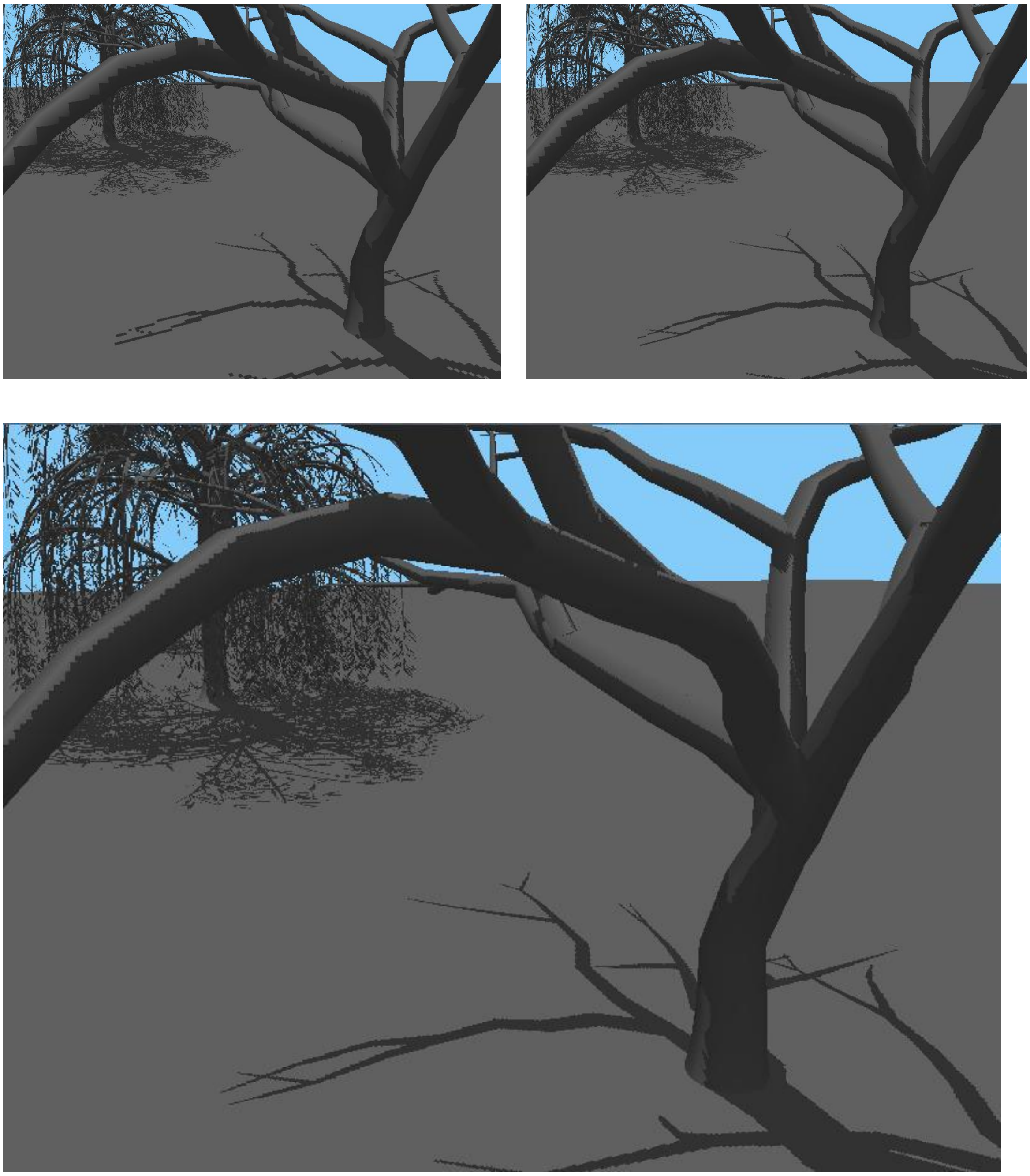

Figure 60: TreeLine scene being rendered with PSSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom)

Variance Shadow Maps

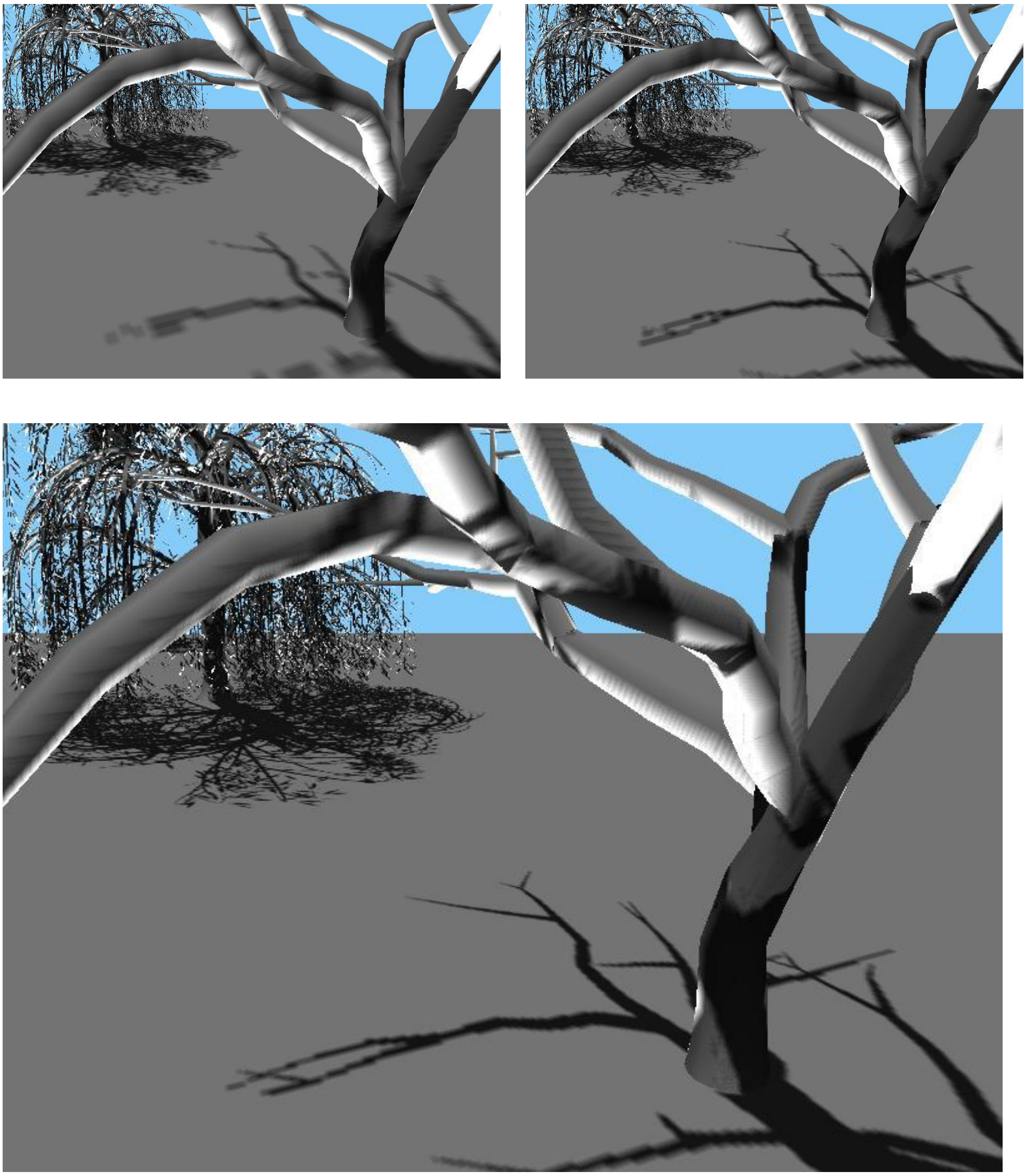

Figure 61: TreeLine scene being rendered with VSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom)

Shadow Mapping Remix Approach

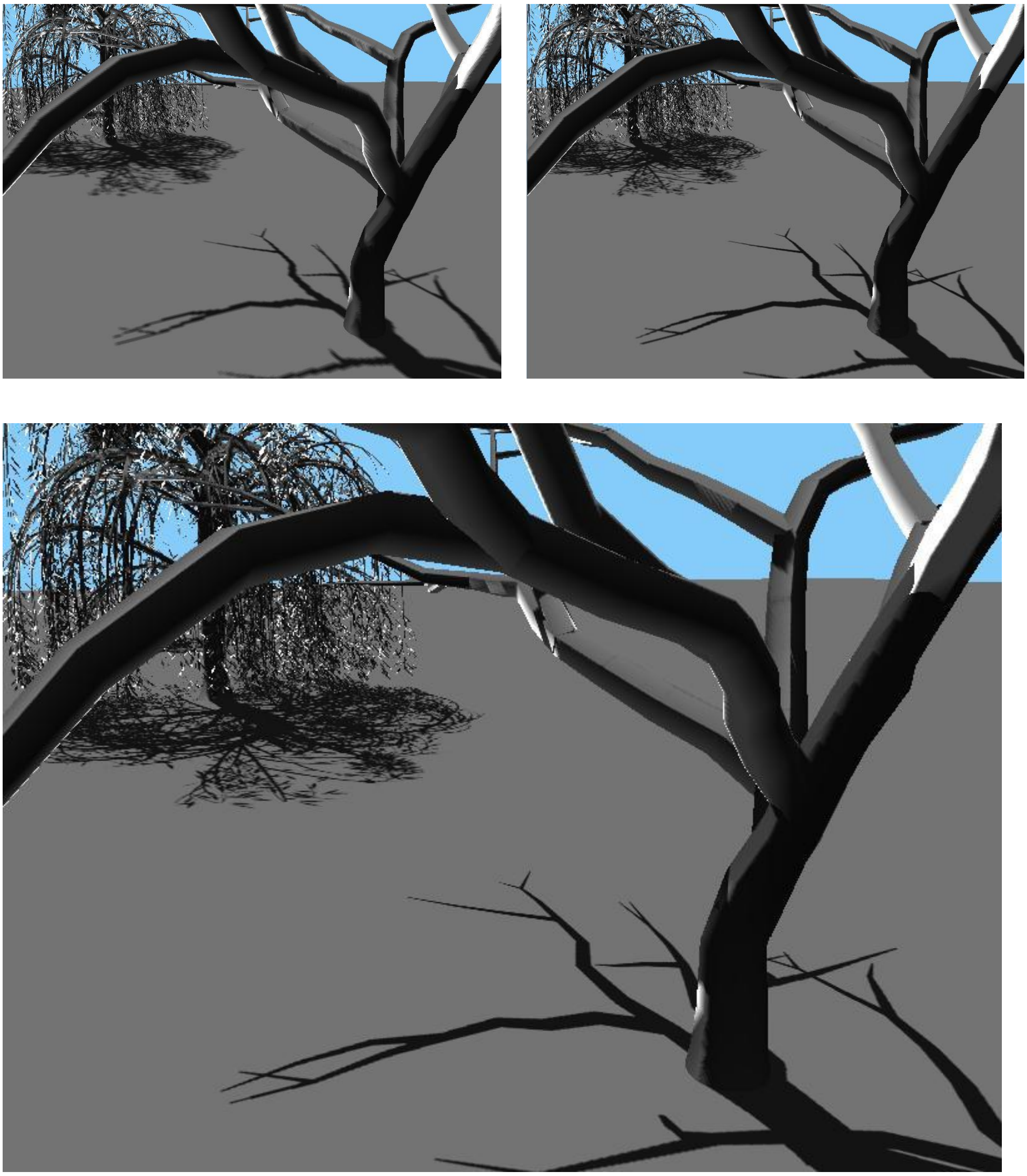

Figure 62: TreeLine scene being rendered with SMRA using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom)

Figure 62 renders the same scene, but in this case with our proposed algorithm: SMRA. The output is very positive, providing better defined shadows than any of the isolated methods. The VSM depth comparison introduces some light leakage as can be seen in Figure 62.

SparseCity

SparseCity is a big scene that is composed almost entirely of low detail buildings. Although the scene has a very big area, it is less complex in terms of geometry. The scene was designed for studying how each one of the algorithms behave in large environments, more precisely, by rendering a large scale area that includes some small, detailed entities inside (which are going to be focused).

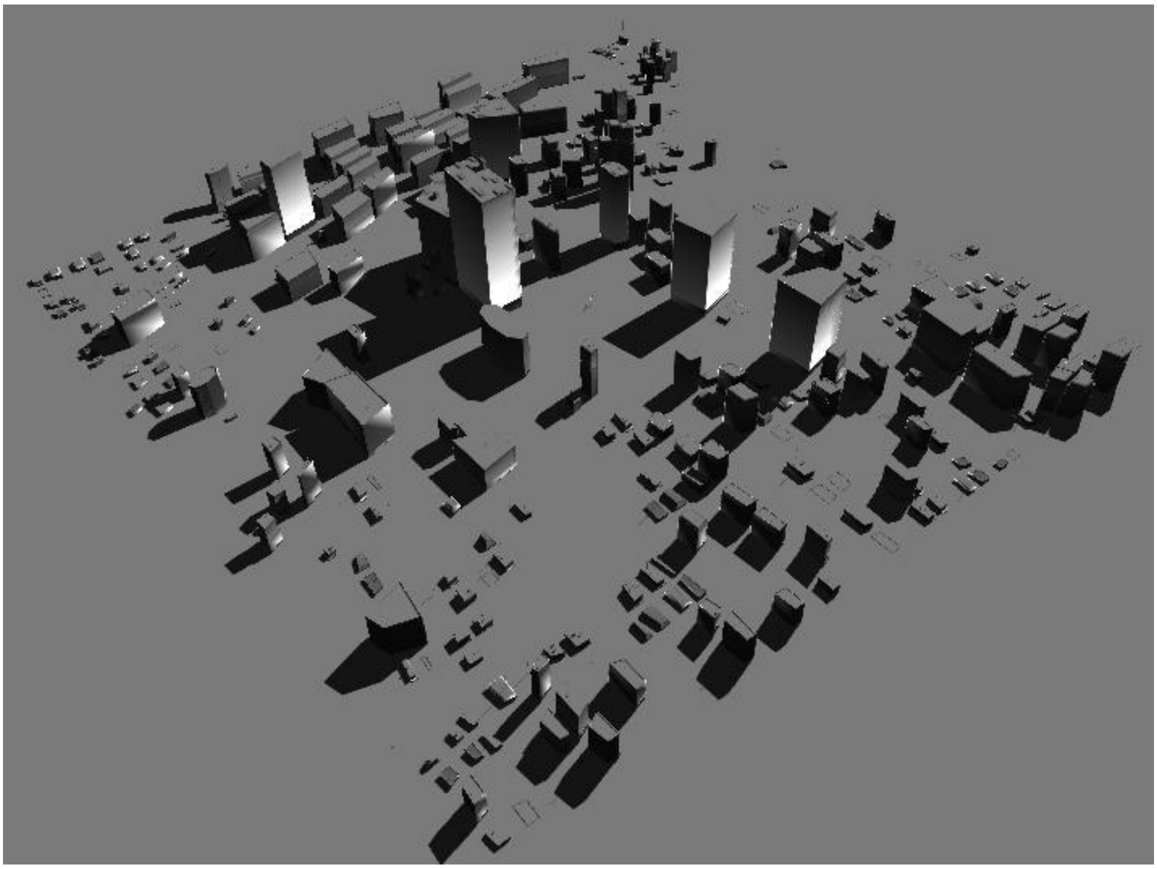

Figure 63: SparseCity rendered with SMRA

Standard Shadow Maps

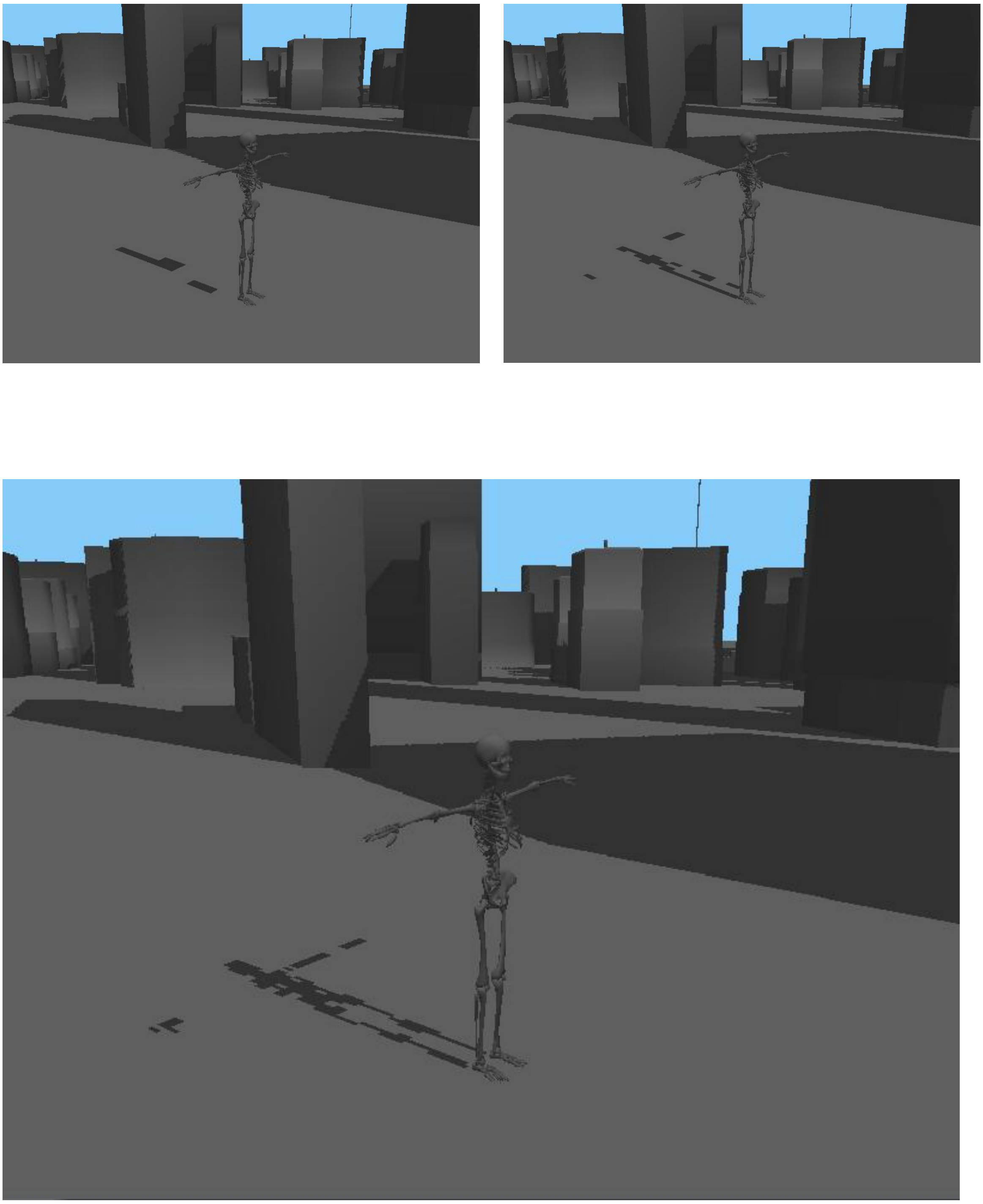

Figure 64: SparseCity scene being rendered with SSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom)

Figure 64 shows the large scene being rendered with SSM. None of the available resolutions is enough to produce acceptable shadows for the entity in the image. This is mainly due to resolution constraints that are not addressed by this algorithm.

Figure 65 shows the same large scene rendered with TSM. The trapezoidal non-uniform distribution of depth shows far better results near the camera, however in the buildings the shadows are worse than with SSM. All resolution shadow maps introduce some pixelation to the scene shadows, mainly in the buildings.

PSSM is rendered in Figure 65. This algorithm was specially proposed for this type of scenes, which leads to satisfactory results with high resolution shadow maps. However, it is still difficult to avoid aliasing since PSSM does not address any special parameterization for that purpose. The errors in the shadows are more distributed than with TSM as expected.

VSM, again produces smooth shadows, but based on SSM. Since SSM failed in producing acceptable shadows for such a large surface, VSM will also fail, alleviating the visual effect a bit due to the shadow map filtering.

Our approach again succeeds at providing better results. The resolution is now better distributed, and the VSM provides smooth shadows in the skeleton. There is however an artifact that is introduced by the usage of VSM, light leakage.

Trapezoidal Shadow Maps

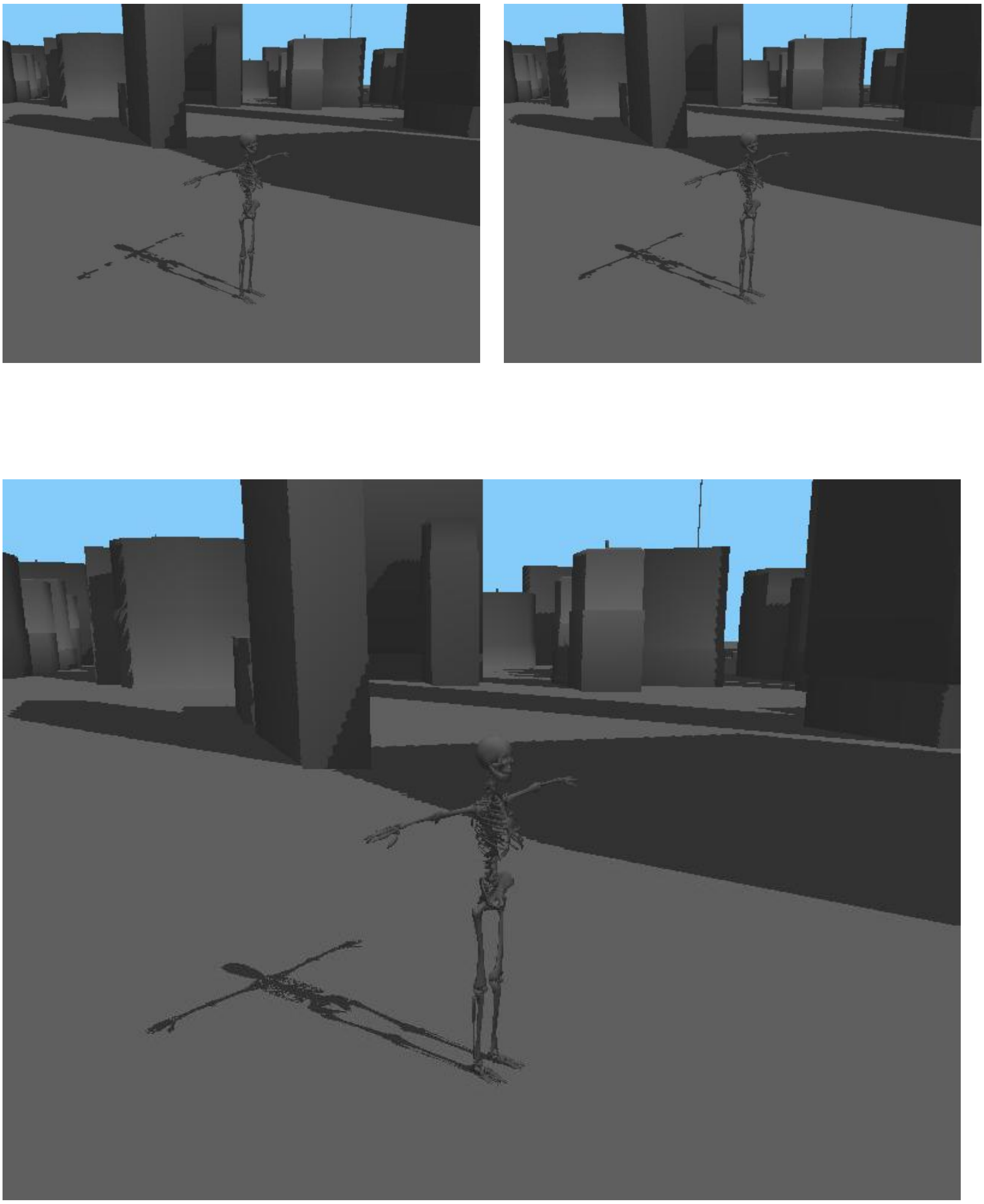

Figure 65: SparseCity scene being rendered with TSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom)

Parallel-Split Shadow Maps

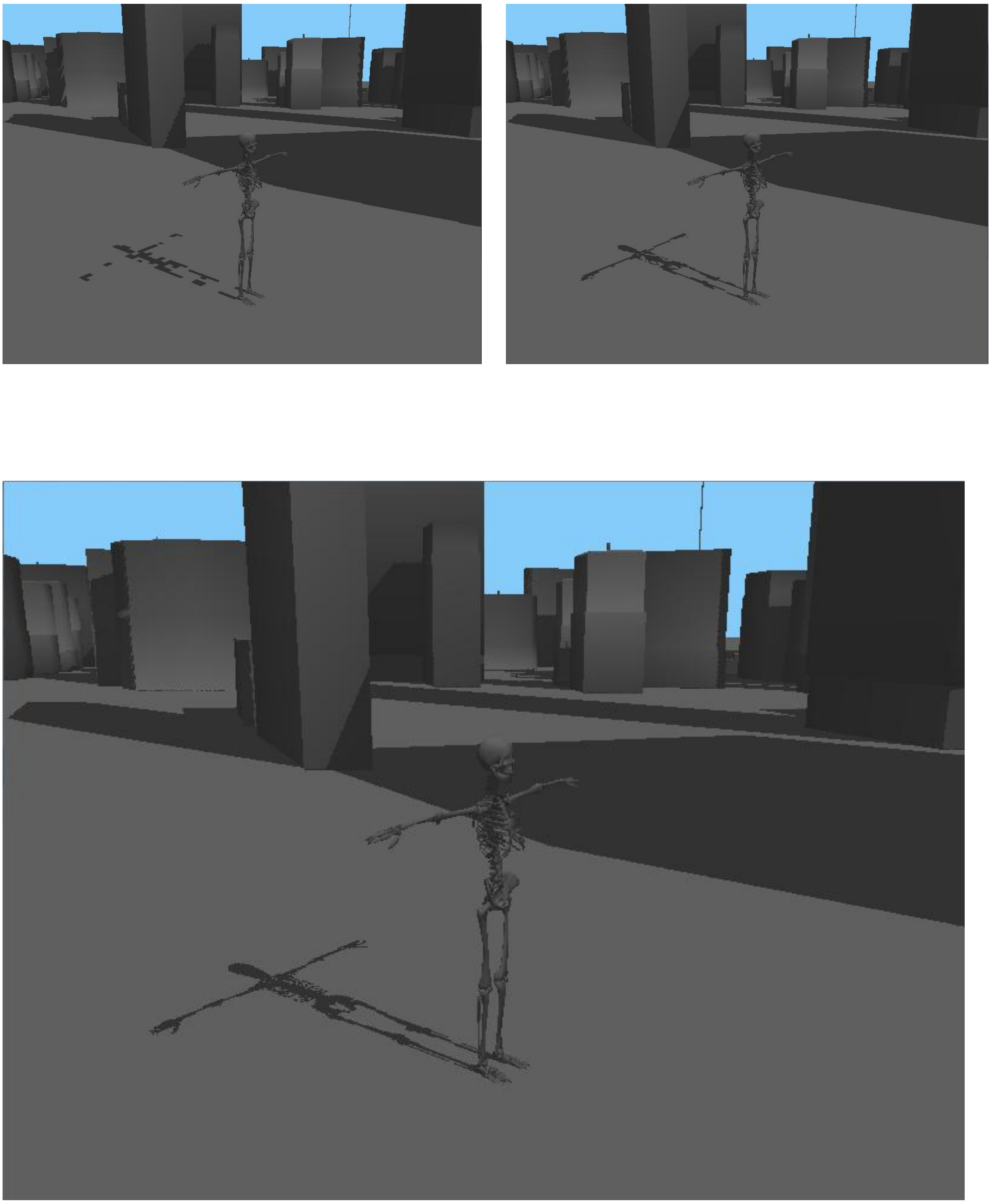

Figure 66: SparseCity scene being rendered with PSSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom)

Variance Shadow Maps

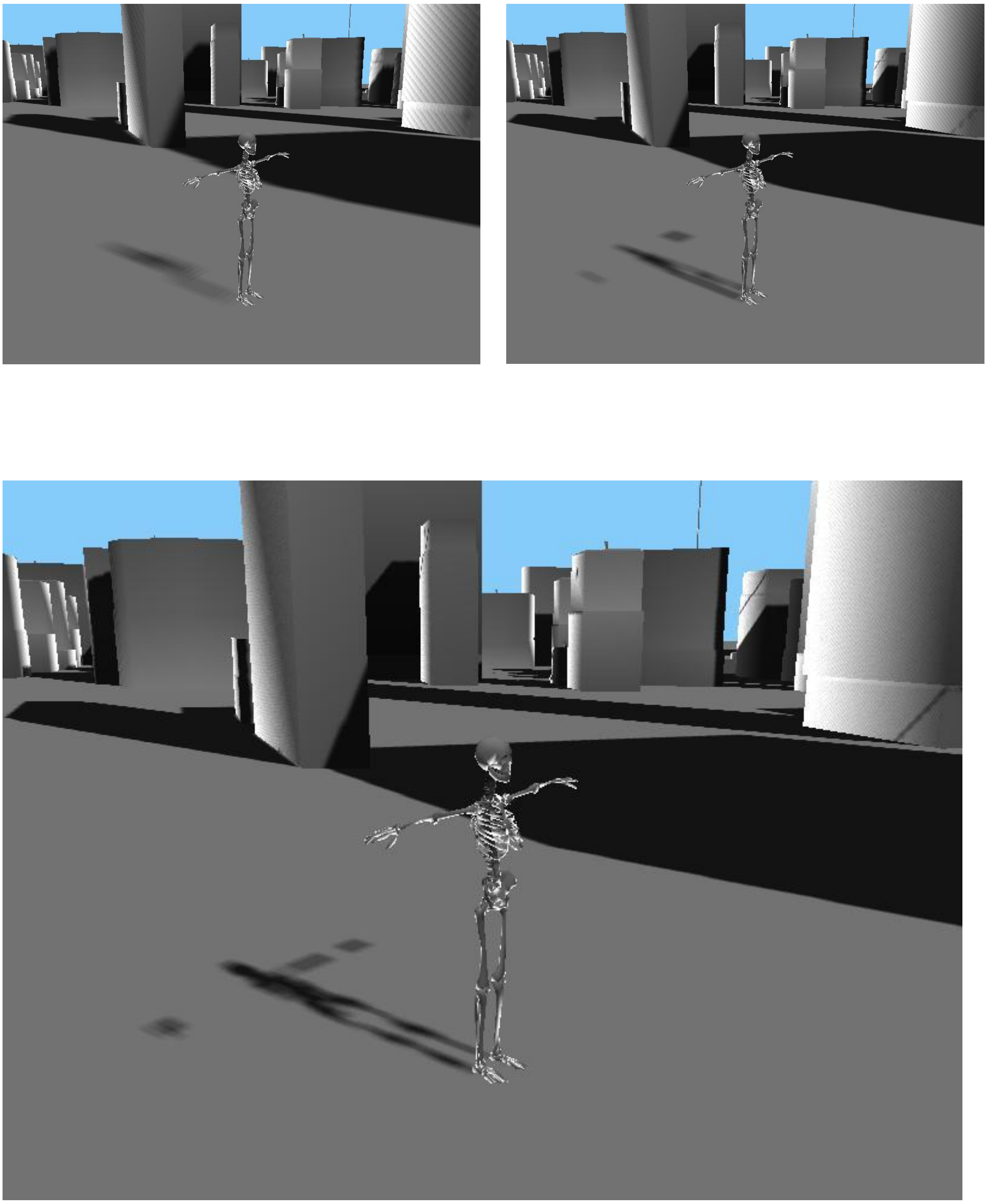

Figure 67: SparseCity scene being rendered with VSSM using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom)

Shadow Mapping Remix Approach

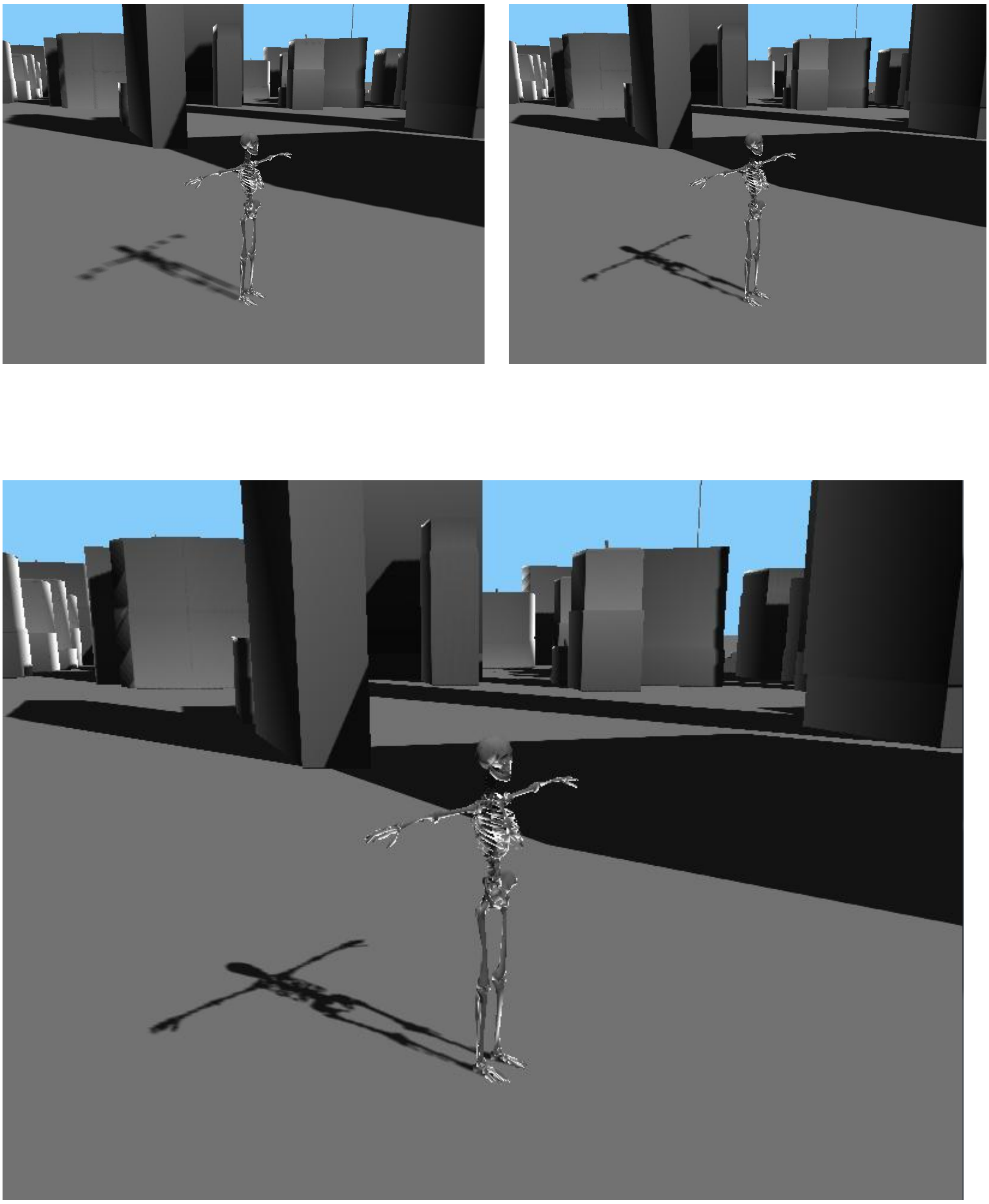

Figure 68: SparseCity scene being rendered with SMRA using shadow mapping resolutions 512x512 (Top Left), 1024x1024 (Top Right) and 2048x2048 (Bottom)

5.3. Performance Results

Application Speed

fps		SSM	TSM	PSSM	VSM	SMRA
SparseCity	512x512	102	88	85	98	79
	1024x1024	98	84	78	96	64
	2048x2048	97	70	63	88	49
TreeLine	512x512	78	56	27	77	40
	1024x1024	74	54	26	74	38
	2048x2048	72	52	24	68	36

Table 2: Rendering speed (in frames per second) for the most relevant algorithms.

As we can see, TSM and VSM are very lightweight. The slowest algorithm, besides SMRA, is PSSM, due to its multiple steps. Since SMRA is based on these three contributor’s main strengths, with the algorithm complexity (created by mixing many complementary techniques, each one of the proposing additional computation to the standard algorithm), it turns also slower than the others, but still acceptable for real-time rendering with a noticeable visual improvement.

5.4. Conclusion

From the results presented above, we can conclude that no shadow mapping algorithm flawlessly solves all the standard shadow mapping artifacts, since it always depend on the scene complexity (size, camera relative to the light) and also on hardware capability (texture sizes, filtering techniques). Despite of that, our algorithm Shadow Mapping Remix Approach, which is formed from the partitioning of TSM, PSSM and VSM behaves well, far better than any of the contributors, although introducing some light leakage from VSM. The expected performance penalty is considered acceptable.

6. Conclusions

Our investigation focused in the analysis of the most relevant contributions for the technique Shadow Mapping. Shadow Mapping is one of the most popular techniques for real-time shadows generation, since its basic algorithm is relatively simple to implement, producing high quality shadows. Although, as an algorithm that is partially based on image-space operations, it is prone to aliasing problems. These problems can be introduced in many ways, such as relationship between the eye and the light, texture size, angle of the light relative to the surfaces being lit, precision errors, among others. Many algorithms have proposed different approaches to solving each of the shadow mapping issues. Some of them have themselves introduced other errors while alleviating others, but at the same time, served as a basis for other more recent algorithms that tackle the shadow mapping problems more efficiently.

We have presented Shadow Mapping Remix Approach, an algorithm based on different, complementary approaches. From our investigation, we conclude that shadow mapping approaches can be split into three main categories: scene partitioning, perspective parameterizations and texture filtering.

Parallel split shadow maps are efficient for large scale scenes, namely outdoor scenes providing an even distribution of the shadow map along a large range of depths. However, each split is performed with standard shadow mapping, hence it has the same issues, although to a lesser extent due to the partitioning of the space. PSSM also introduced a greedy adaptive frustum approach which was not used in our approach since it introduces the continuity problem.

Trapezoidal Shadow Mapping provides a perspective parameterization that redistributes the resolution focusing on the areas near the camera. However this produces worse shadows far away from the camera than SSM, although these are not as relevant.

Variance Shadow Mapping provides a mechanism to filter the shadows efficiently, taking advantage of the hardware enabled filtering operations the shadows. It adds some softness to the shadows without incurring into a significant performance penalty. The main issue with VSM is the introduced light leakage.

Based on these three complementary categories, we chose the most relevant contributions and extracted from them the most effective techniques for addressing the shadow mapping issues, merging all the extracted techniques into a more robust algorithm that is able to address most of the categories of problems with which shadow mapping is associated. There is a natural performance penalty, but it remains highly acceptable. The visual results show a more effective distribution of the resolution of the shadow maps, with a nice soft effect, although it introduces some light leakage associated with VSM.

6.1. Future Work

The redistribution of the resolution of the depth maps provides better results but still not optimal. The resolution of each split does not necessarily have to be the same. Exploring strategies to better distribute the resolution in this axis is still unchartered territory that may prove fruitful.

Also the light leakage from VSM is too noticeable in some situations. Other filtering strategies should be explored to overcome this problem.

Appendix A

Trapezoid side lines computation

Assuming the first *d* distance from the near plane as the focus region, the camera frustum is then truncated at that distance. Let P*L* be the point *d* distant from the near plane, lying on *l* in post-perspective space of the light L (Figure 69). Let also *d'* be the distance of P*L* from the top line of the trapezoid. The trapezoid is then constructed in order to contain E, so that the transformation matrix maps P*L* to a point on the line *l* of 80% of the shadow map. This approach is termed by TSM as the 80% rule.

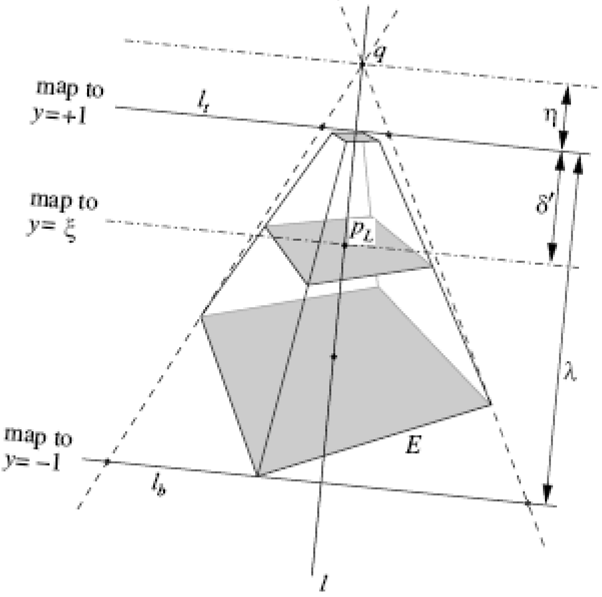

Figure 69: A 1D homogenous perspective projection problem to compute q

For doing this, it is necessary to formulate a perspective projection problem in order to compute the position *q* on *l*, being *q* the center of projection on *l*, mapping P*L* to the so called 80% line ξ = y = 1,6 where the base line is y=-1 and the top line is y=1. Let also $\overline{nf}$ [λ] be the distance between base and top lines of the trapezoid.

The distance $\overline{qn}$ [η] from *q* to the top line will be computed through a homogenous perspective projection*:

$$\overline{qn} = \frac{\overline{nf}\,d' + \overline{nf}\,d'\xi}{\overline{nf} - 2d' - \overline{nf}\,\xi}$$

* For more details on this homogenous perspective projection, please refer to [TSM04].

From here, *q* is calculated and the two side lines of the trapezoid are extracted. These lines pass through *q* and touch the convex hull of E.

References

[ASM01] Fernando R., Fernandez S., Bala K, Greenberg D.P.; Adaptive Shadow Maps, 2001

[ASSMT08] Bavoil L.; Advanced Soft Shadow Mapping Techniques, 2008

[CasSM07] Dimitrov R.; Cascaded Shadow Maps, 2007

[CCSCS78] Williams L.; Casting curved shadows on curved surfaces, 1978

[CSM07] Annen T., Mertens T., Bekaert P., Seidel H.P., Kautz J.; Convolution Shadow Maps, 2007

[EHSRA04] Chan E., Durand F.; An Efficient Hybrid Shadow Rendering Algorithm, 2004

[ESM08] Annen T., Mertens T., Seidel H.P., Flerackers E. Kautz J.; Exponential Shadow Maps, 2008

[FastSTM92] Segal M.,Korobkin C.,Widenfelt R.v.,Foran J.,Haeberli P.; Fast Shadows and Lighting Effects Using Texture Mapping, 1992

[Heckbert86] Heckbert P.; Survey of Texture Mapping,1986

[MathGMCG] Lengyel E., Mathematics for 3D Game Programming and Computer Graphics

[LiSPSM04] Wimmer M., Scherzer D., Purgathofer W.; Light Space Perspective Shadow Maps, 2004

[OPG6ed] Shreiner D., Woo M., Neider J., Davis T.; OpenGL Programming Guide, 6th Edition

[PCF87] Reeves W., Salesin D., Cook R.; Rendering anti-aliased shadows with depth maps, 1987

[PluSM01] Tadamura K., Qin X., Jiao G., Nakamae E.; Rendering optimal solar shadows with plural sunlight depth buffers, 2001

[PraSM02] Brabec S., Annen T., Seidel H.P.; Practical Shadow Mapping, 2002

[PSM02] Stamminger M., Drettakis G.; Perspective Shadow Maps, 2002

[PSSM06] Zhang F., Sun H., Xu L., Lun L.K.; Parallel-Split Shadow Maps for Large-scale Virtual Environments, 2006

[SACG77] Crow F.; Shadows Algorithms for Computers Graphics, 1977

[SalviESM08] Salvi M.; Rendering filtered shadows with exponential shadow maps, 2008

[SoftSM05] Valient M., Bujnak T.; GPU friendly, anti-aliased, soft shadow mapping, 2005

[SRTSSA03] Hasenfratz, J.M., Lapierre M., Holzschuch N., Sillion F.; A survey of Real-time Soft Shadows algorithms, 2003

[SSA90] Woo A., Poulin P., Fournier A.; A Survey of Shadow Algorithms, 1990

[STIRTA04] Brabec S., Shadow Techniques for Interactive and Real-Time Applications, 2004

[TSM04] Martin T., Tan T.S.; Anti-aliasing and Continuity with Trapezoidal Shadow Maps, 2004

[VSM06] Donnelly W., Lauritzen A.; Variance Shadow Maps, 2006

Web References

[WwwARF] Fernandes A.R.; OpenGL @ Lighthouse 3D,

http://www.lighthouse3d.com

[WwwTSM] Martin T.; Trapezoidal Shadow Maps (TSM) – Recipe,

http://www.comp.nus.edu.sg/~tants/tsm/TSM_recipe.html

Models and Utilities

City model generated using Citygen

http://citygen.net/

Trees and other models from 3dVia

http://www.3dvia.com