

Sample Distribution Shadow Maps

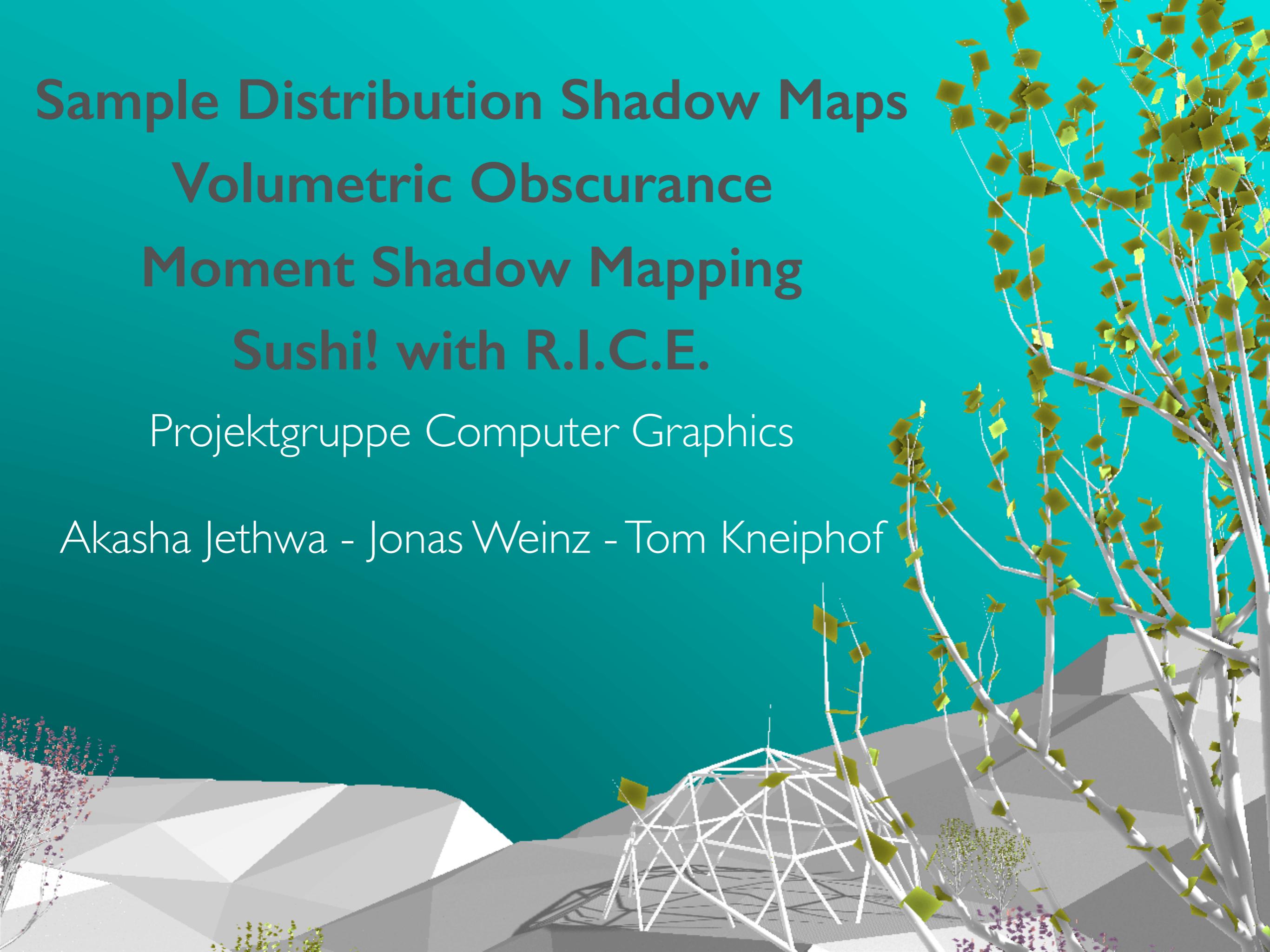
Volumetric Obscurrence

Moment Shadow Mapping

Sushi! with R.I.C.E.

Projektgruppe Computer Graphics

Akasha Jethwa - Jonas Weinz - Tom Kneiphof



INHALT

1. Inhalt

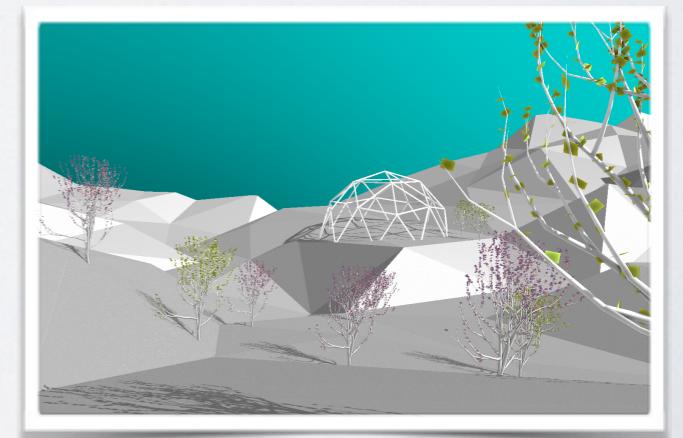
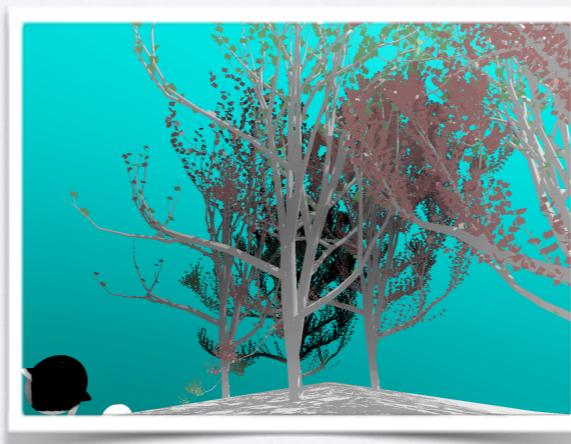
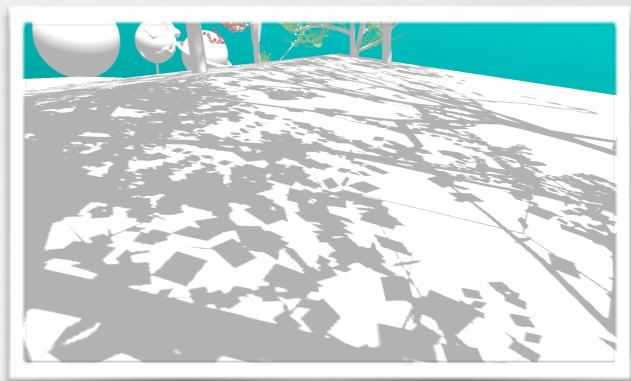
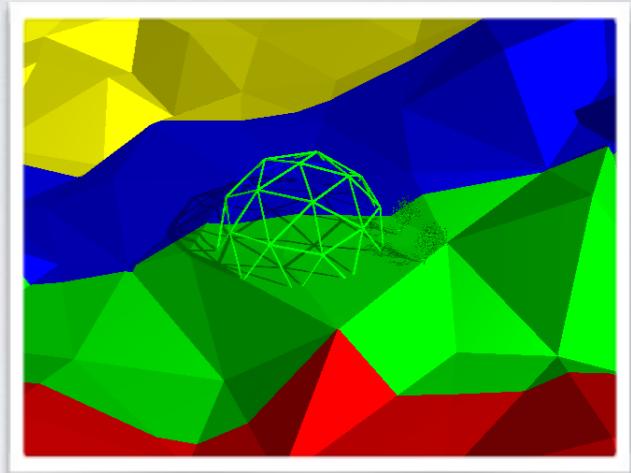
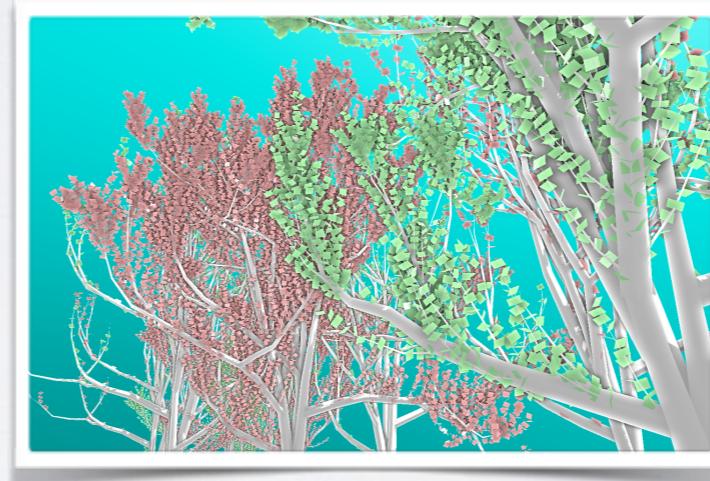
2. Sample Distribution Shadow Maps

3. Volumetric Obscurrence

4. Moment Shadow Mapping

5. Sushi! with R.I.C.E.

6. Fazit und Aussichten



2. SAMPLE DISTRIBUTION SHADOW MAPS

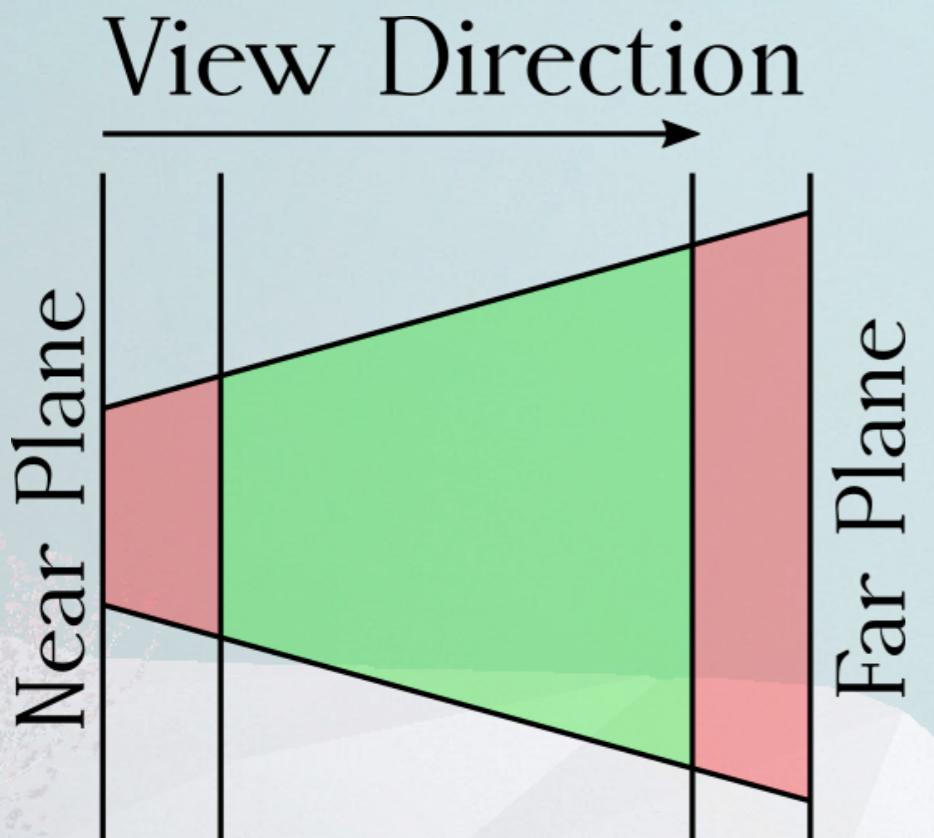
- Betrachtet Screen Samples im Light Space
- Vermeide Oversampling, Undersampling, Geometrisches Aliasing
- Partitionieren Kamera Frustum entlang Z-Achse
- Adaptive Partitionierung
- Shadow Map Frustum eng um Geometrie

2.I SDSM: PARALLEL SPLIT SHADOW MAPS

- Partitionierung zwischen Near- und Farplane der Kameraprojektion
- Konvexkombinierte Kaskadengrenzen uniformer und logarithmischer Partitionierung
- Shadow Map Frusta um Eckpunkte der Kaskaden

2.2 SDSM: NEAR & FAR PLANE

- Verschiebe Near- und Far-Plane dicht an die Geometrie
- Min/Max Reduktion auf Depth-Buffer

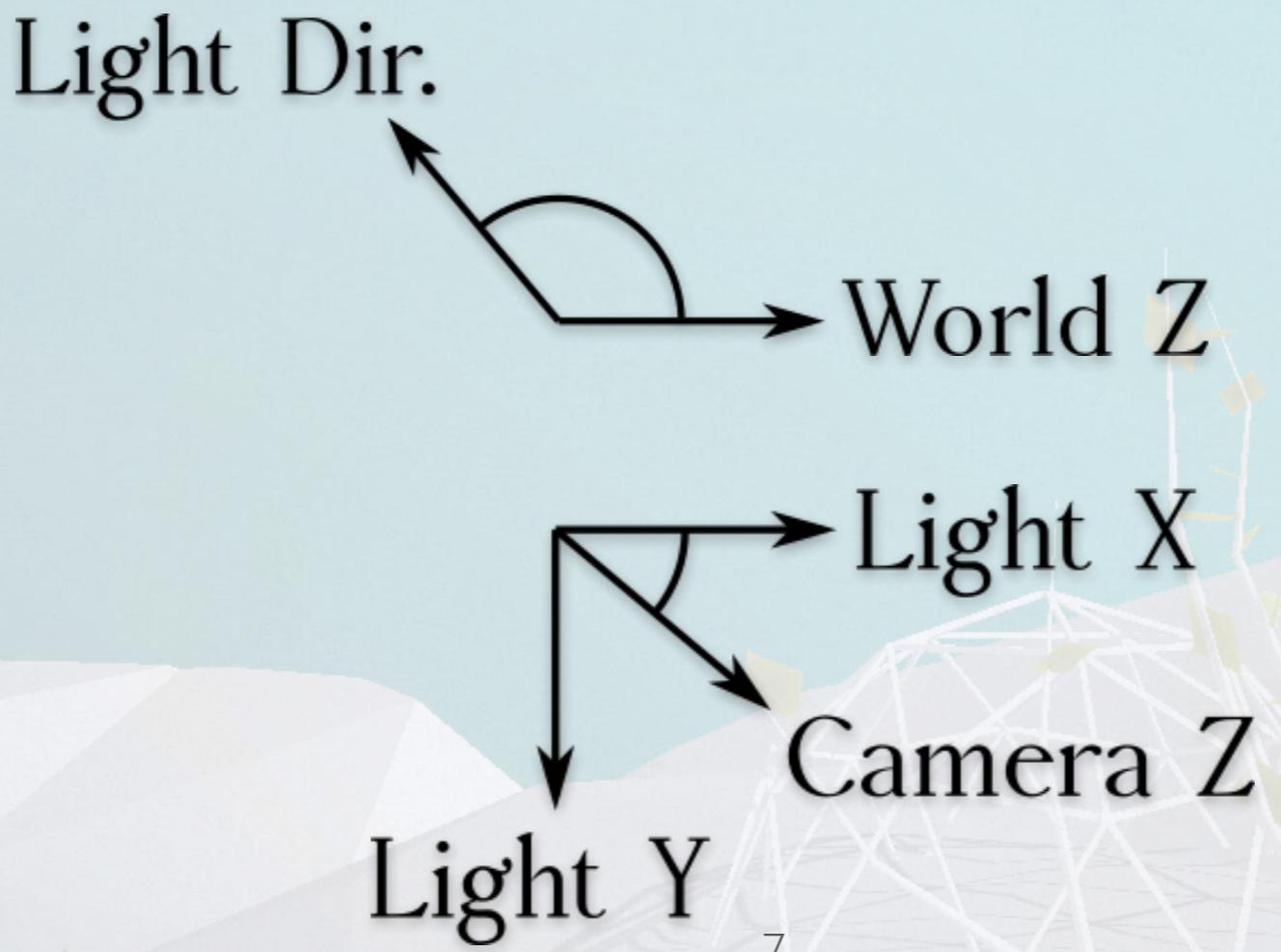


2.3 SDSM: DEPTH PARTITIONING

- Logarithmisch
- Adaptiv logarithmisch
 - Vermeidung von Lücken im Depth-Buffer Histogramm
- K-Means
 - Partitionieren um Maxima des Depth-Buffer Histogramms

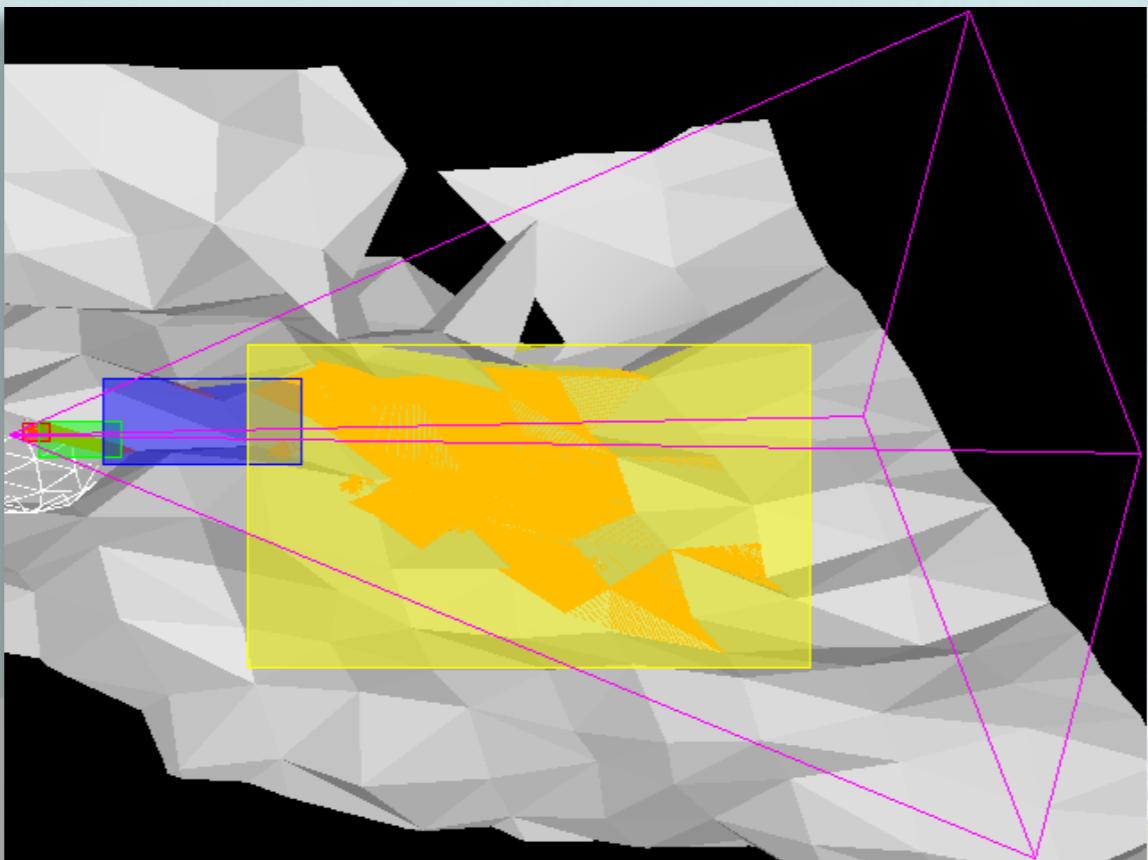
2.4 SDSM: LIGHT SPACE

- Rotiere Light Direction auf Z-Achse
- Rotiere Kamera Z-Achse um die resultierende Z-Achse auf X-Achse



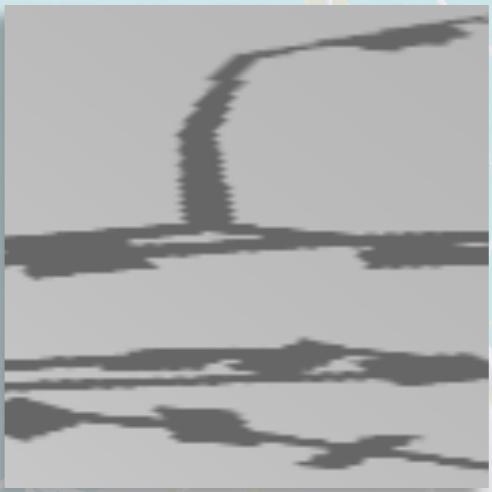
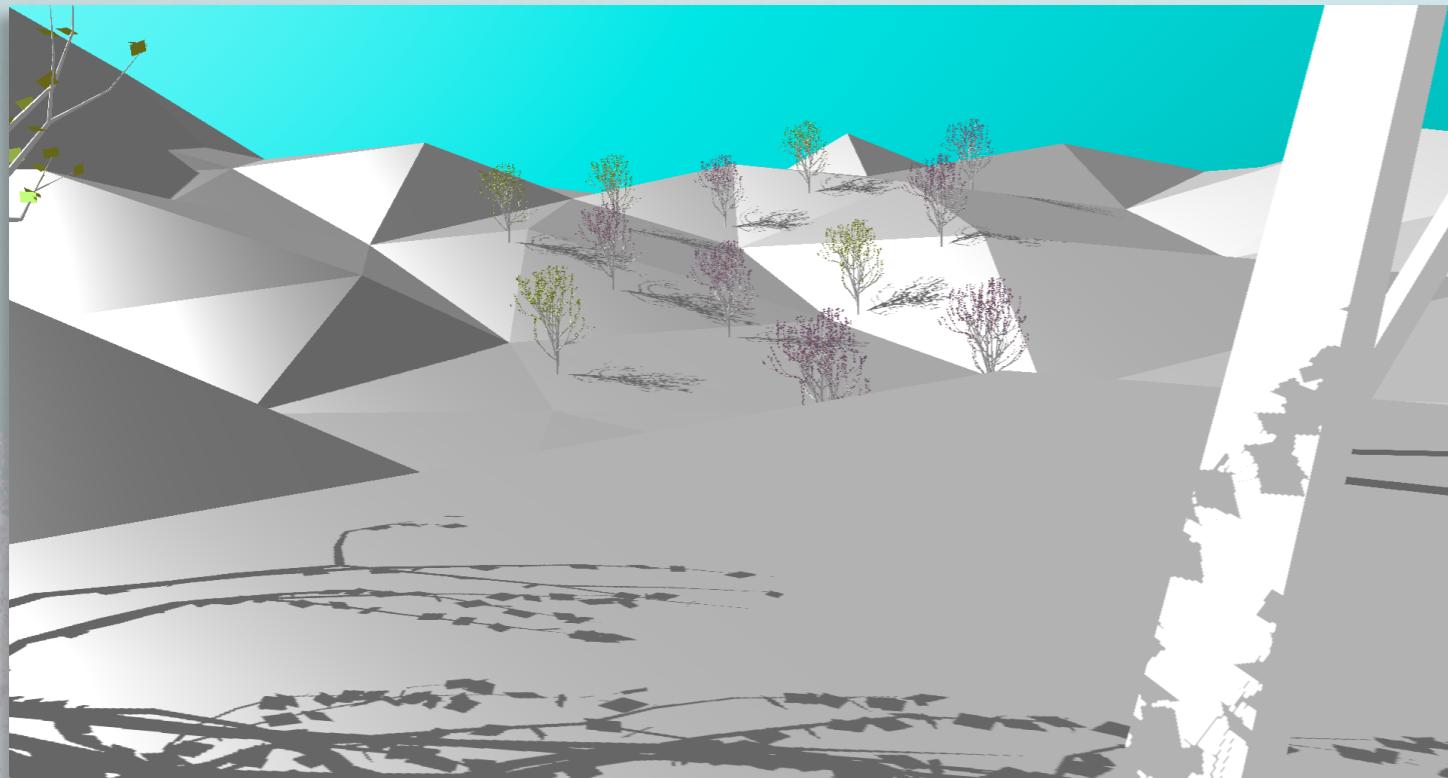
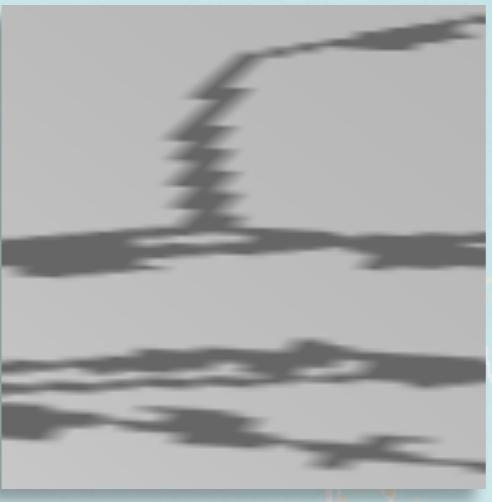
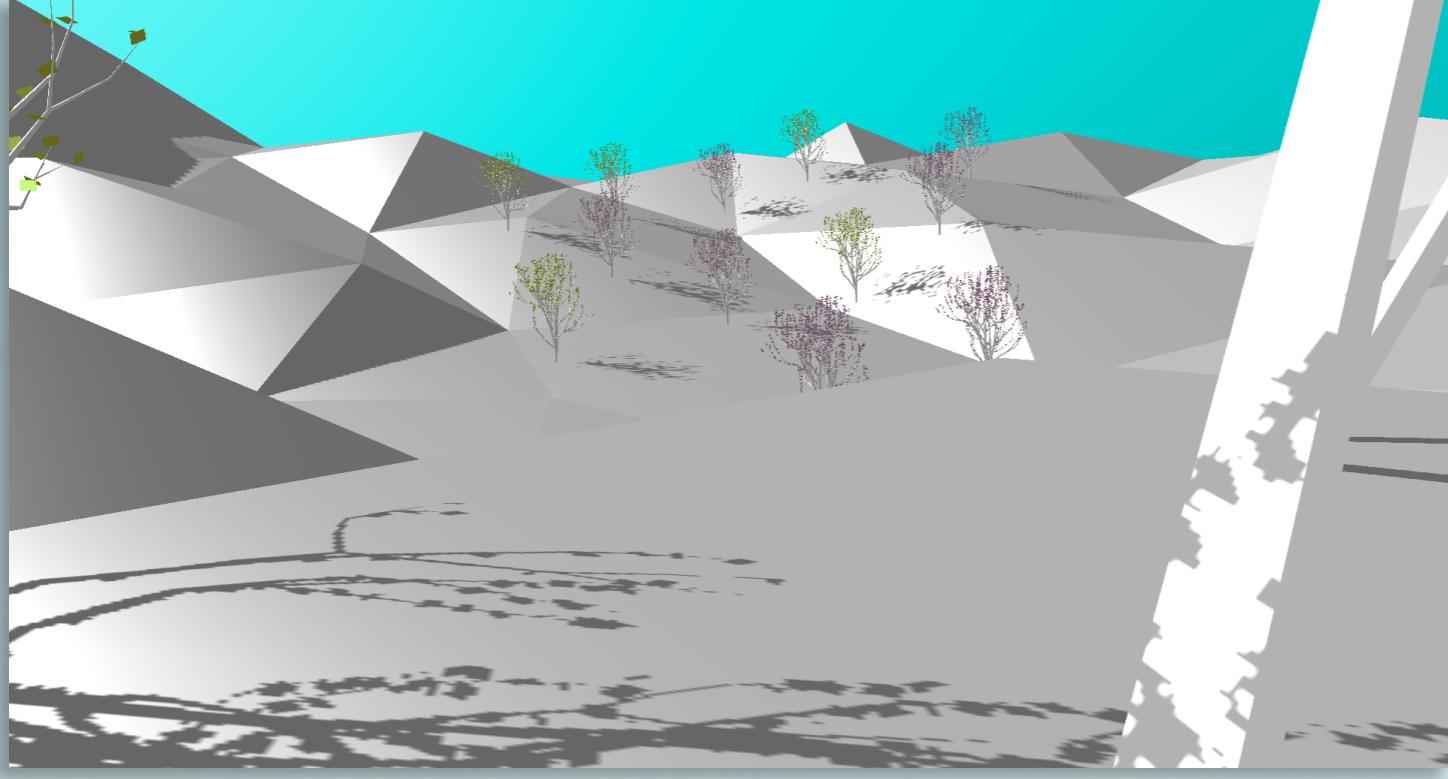
2.5 SDSM:TIGHT PARTITION FRUSTA / LIGHT SPACE

- Shadow Map Frusta eng um sichtbare Geometrie
- Berechne AABB der sichtbaren Geometrie per Reduktion

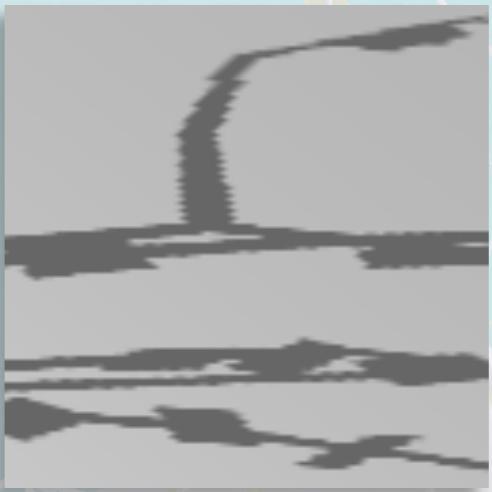
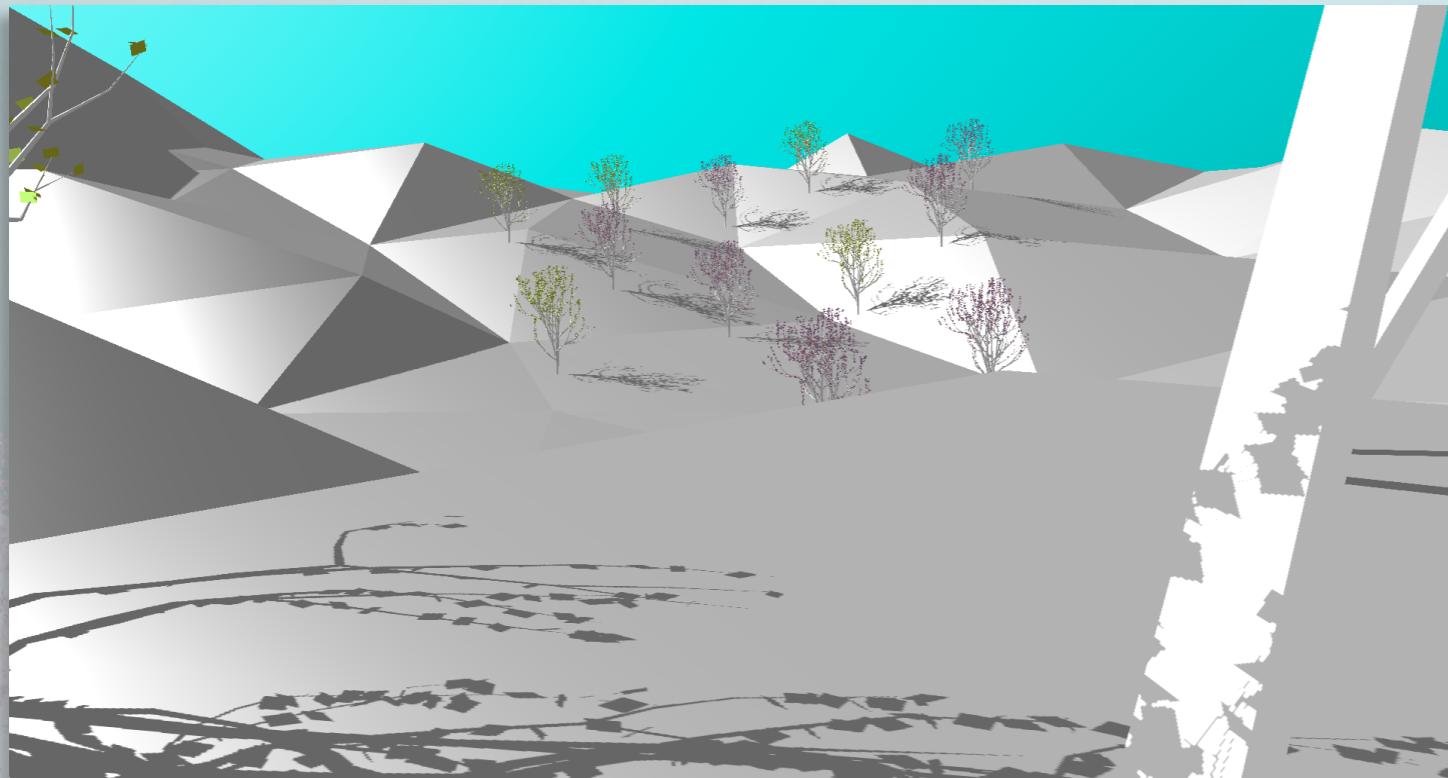
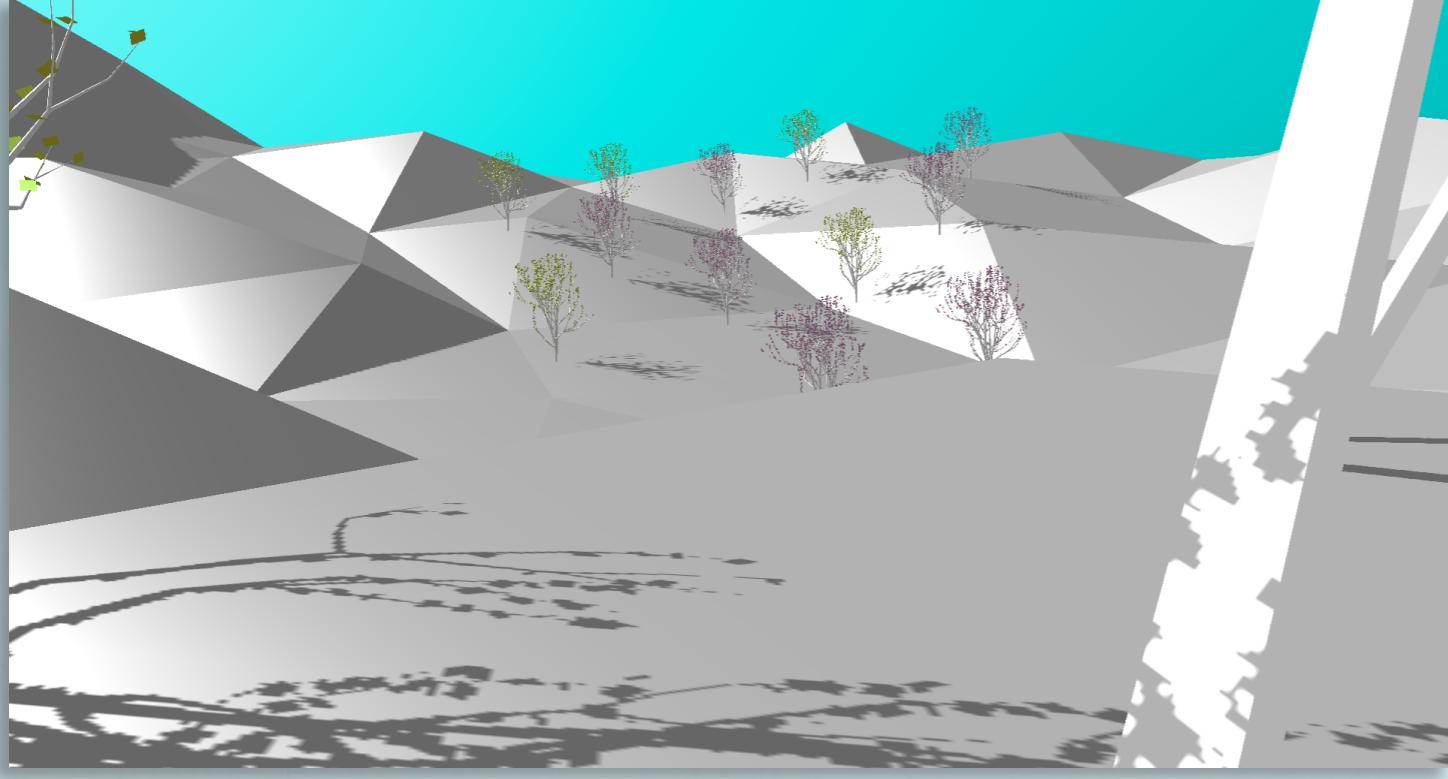


2.6 SDSM: REDUKTION

- Betrachte Minimierungsprobleme
- Nutze 1D Textur mit mehreren Kanälen
- Eine Ausgabe pro Threadgruppe
- 16×16 Threads pro Gruppe

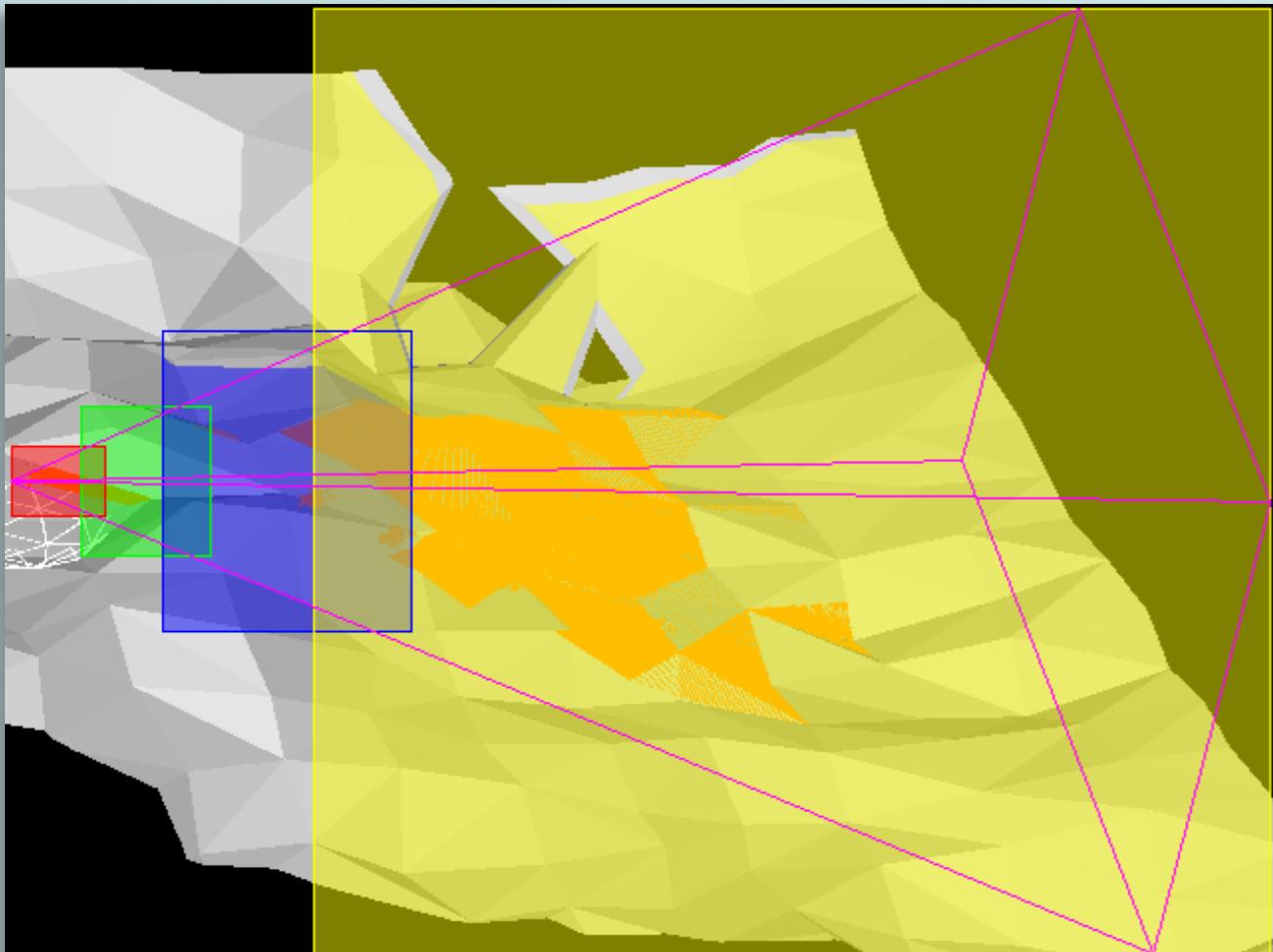


PSSM

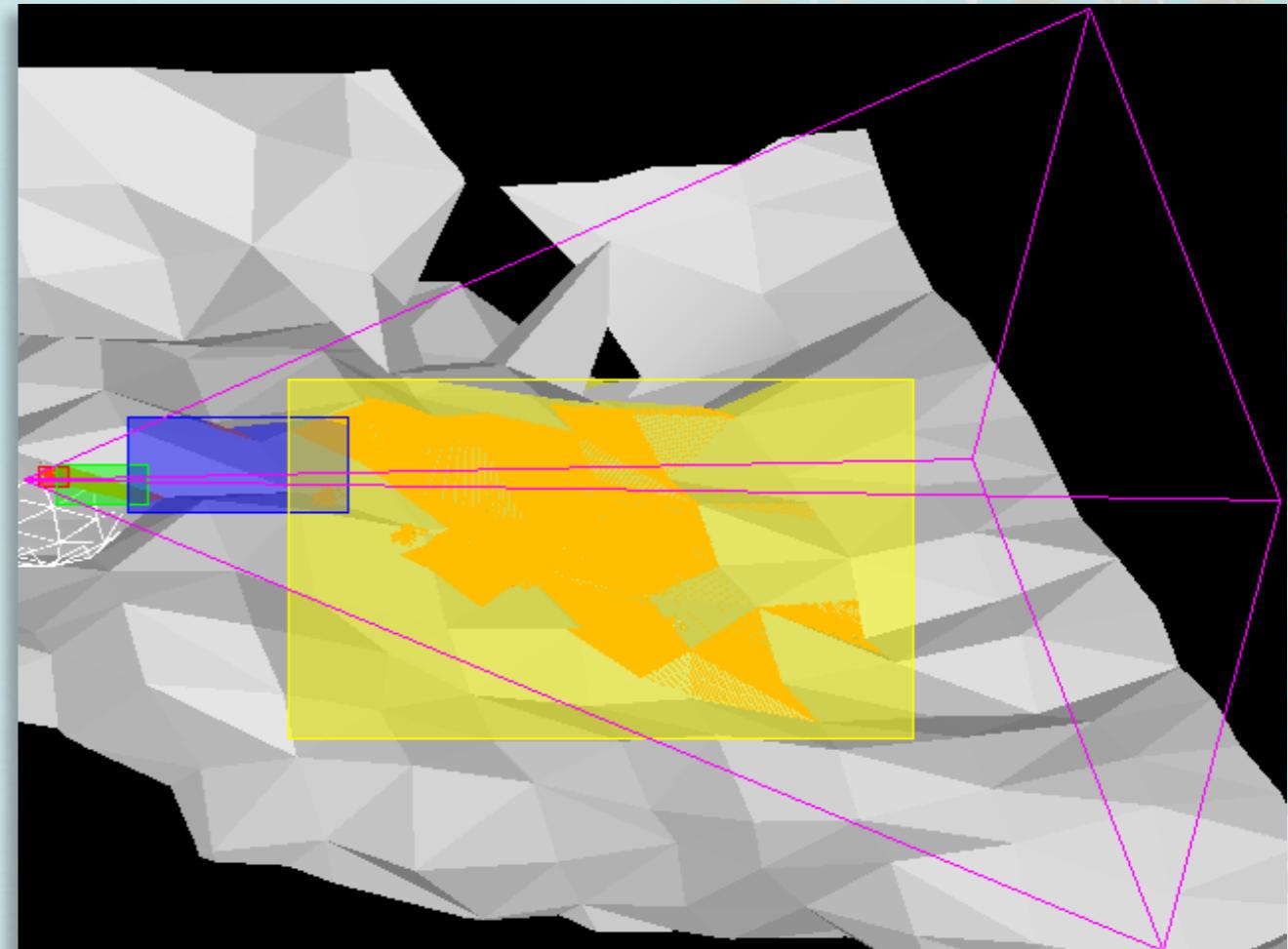


SDSM

2.7 SDSM: LIGHT SPACE FRUSTA

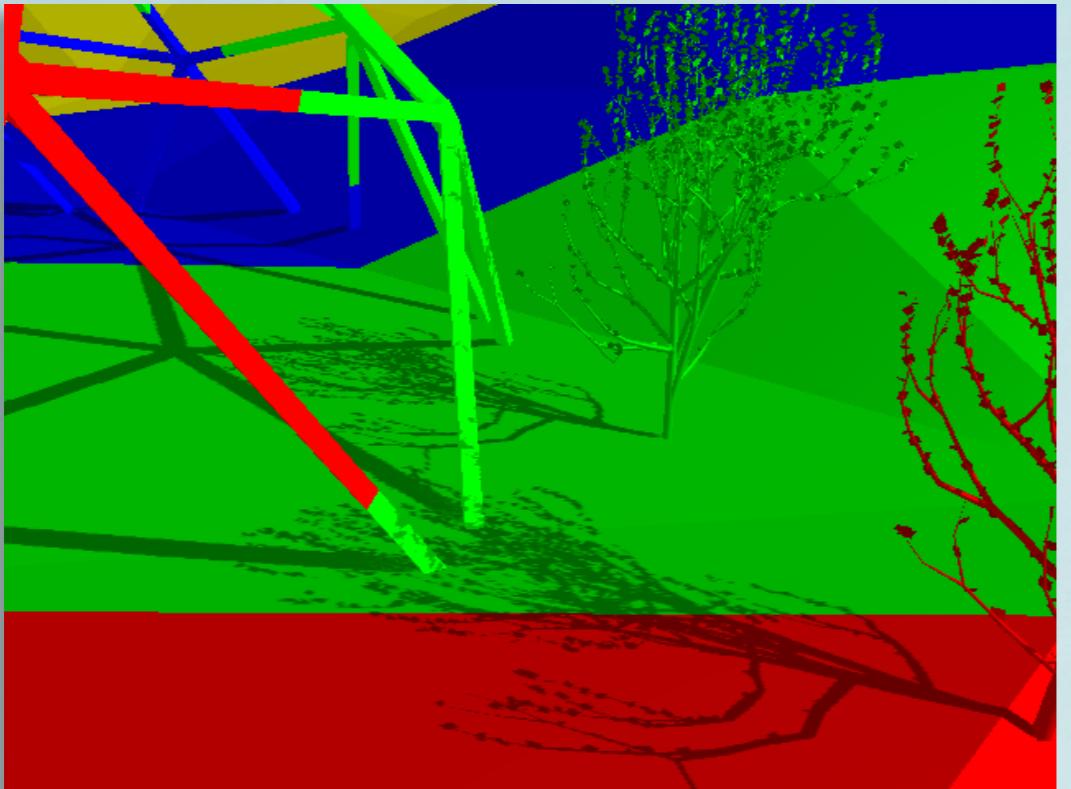


PSSM

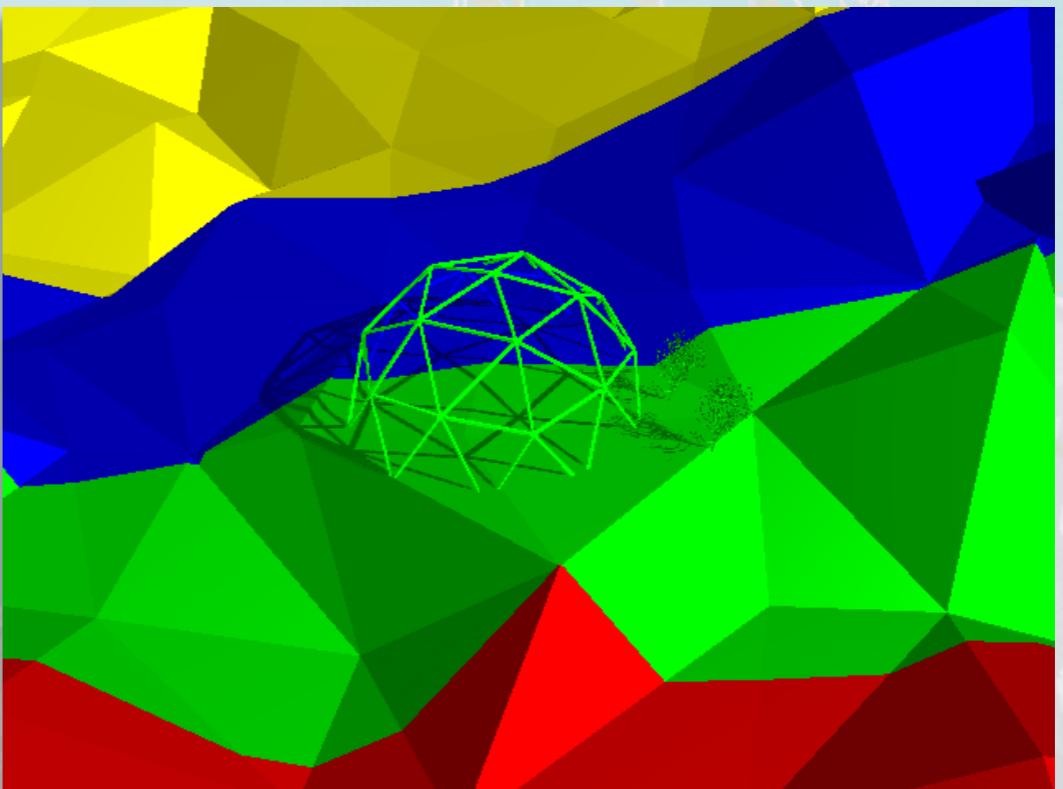
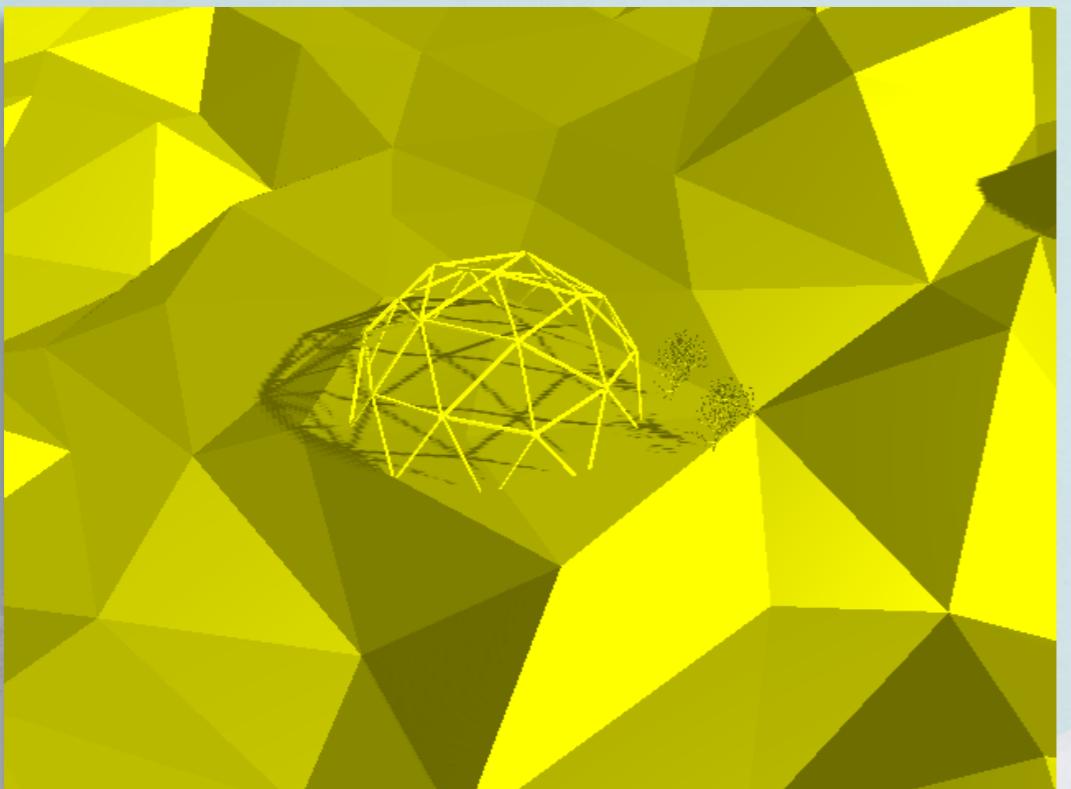
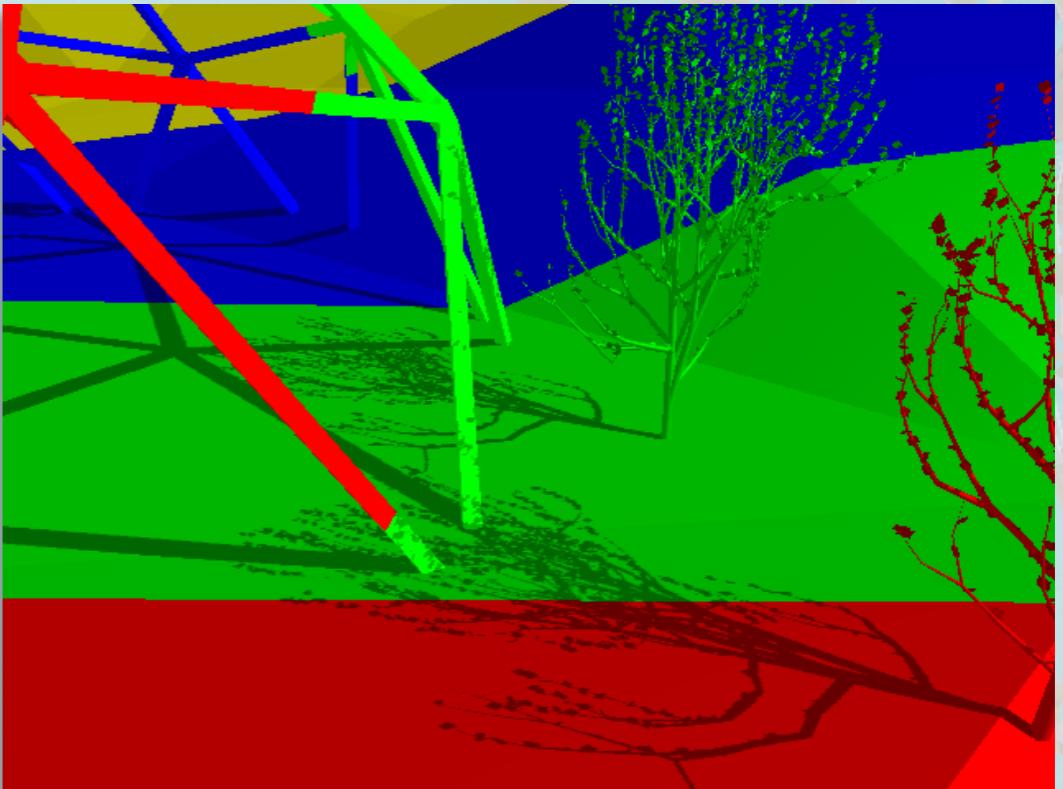


SDSM

PSSM

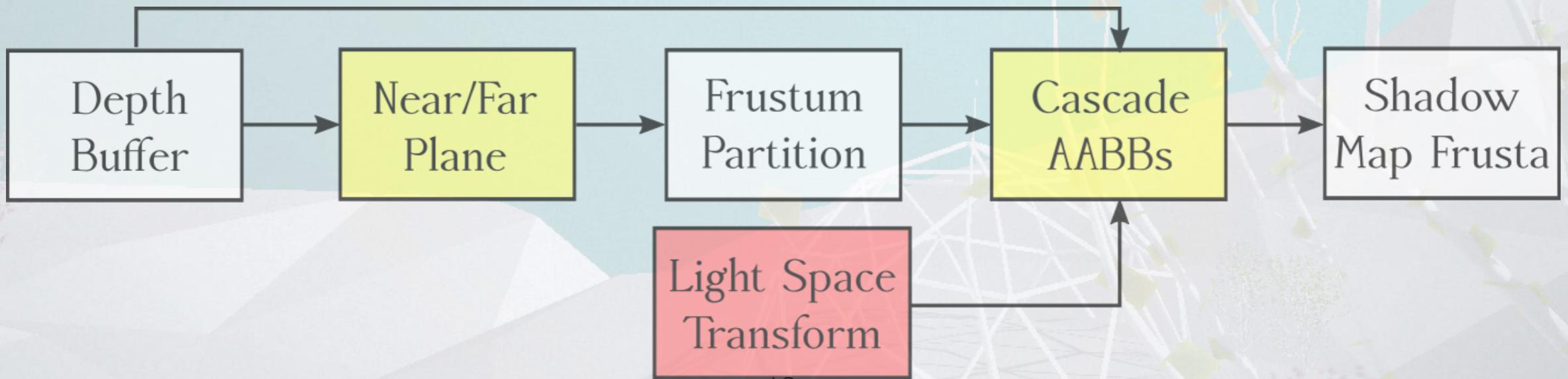


SDSM

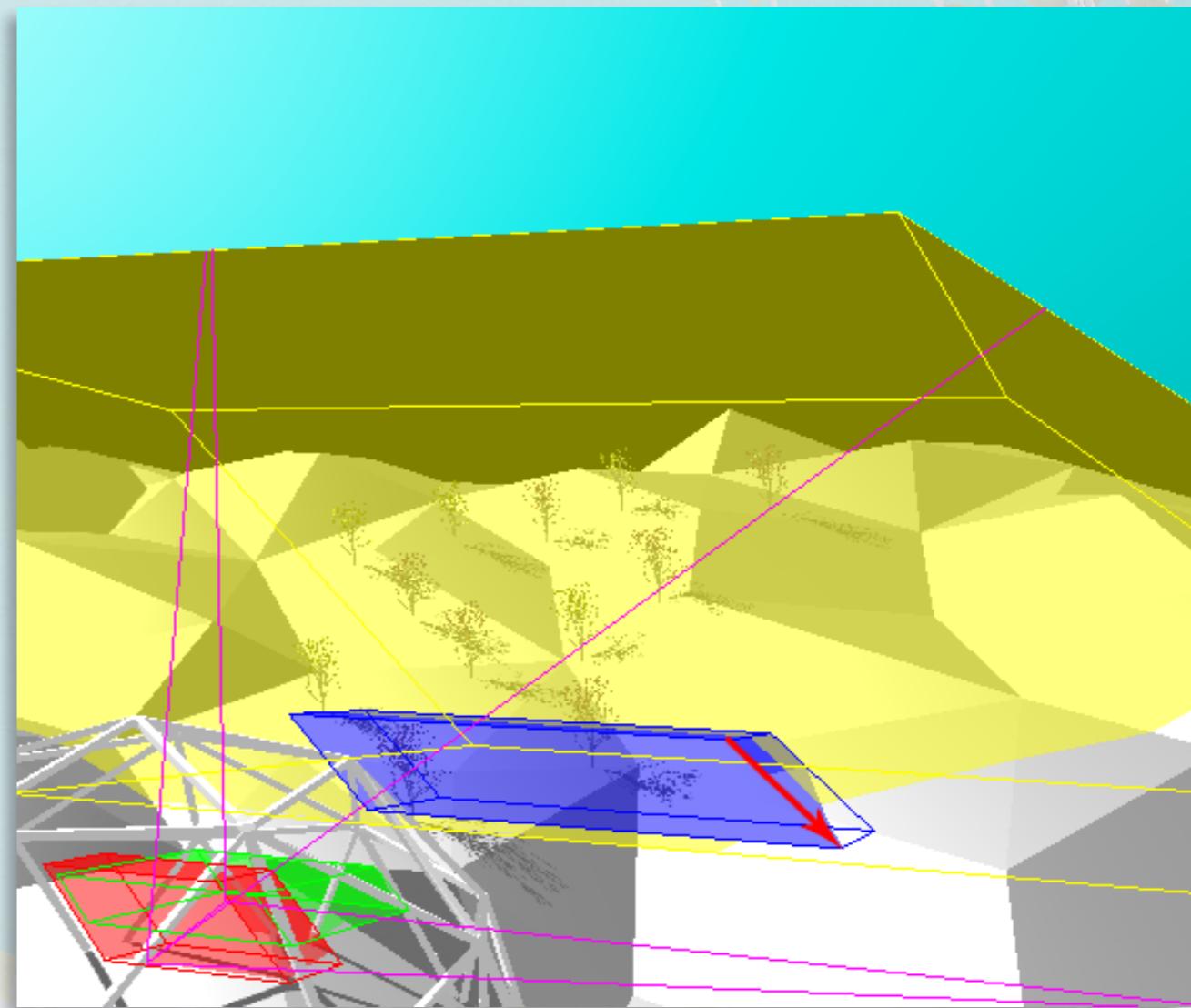
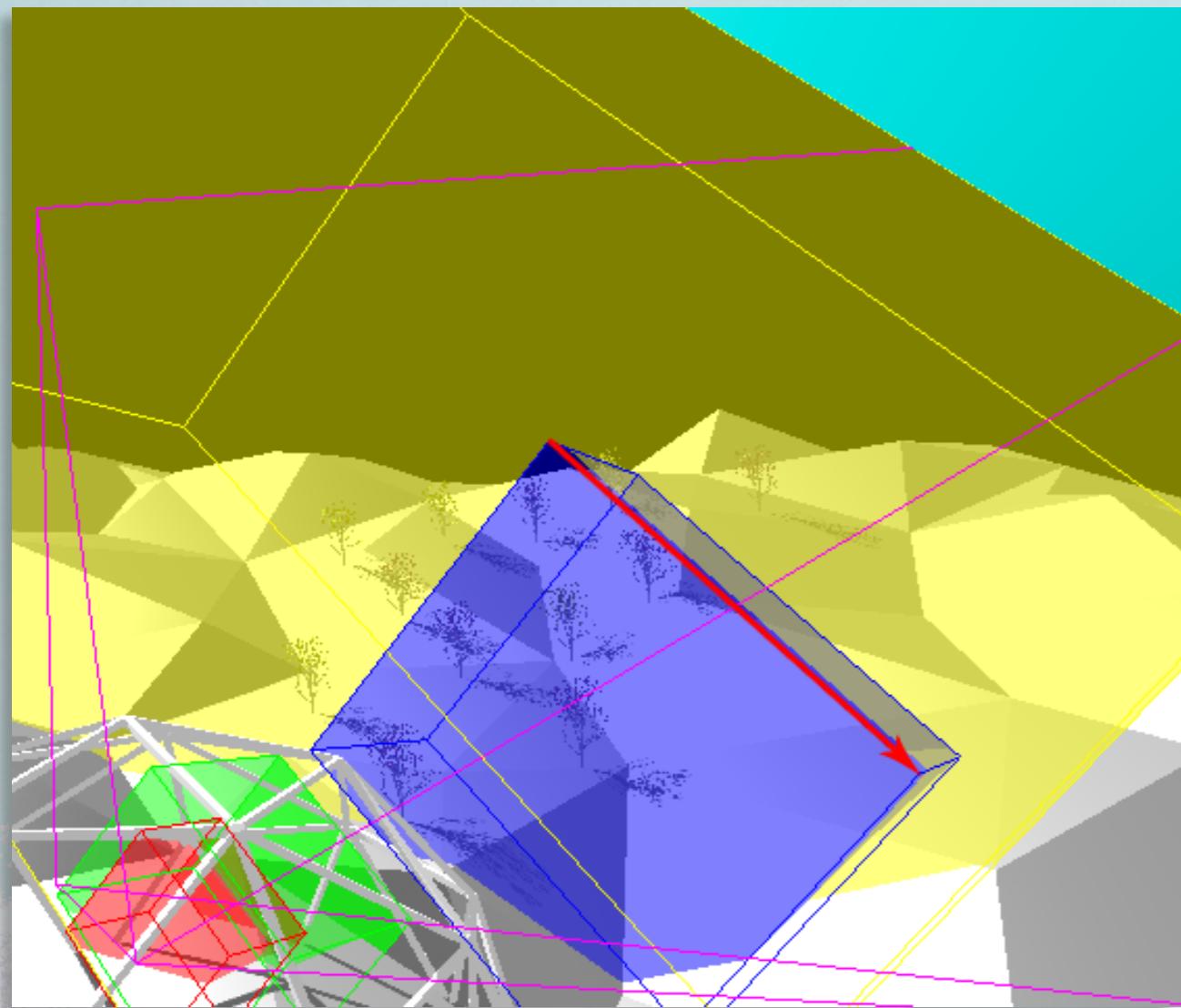


2.8 SDSM: ERGEBNISSE

- SDSM ist sehr robust
- Zuverlässige Ergebnisse
- Screenspace Algorithmus
- Vollständig auf GPU implementierbar

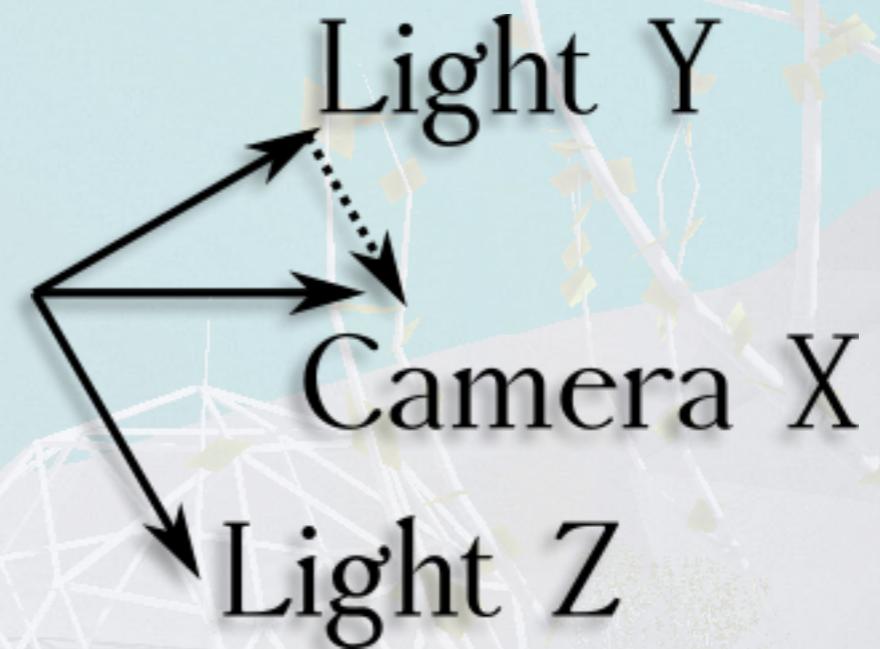
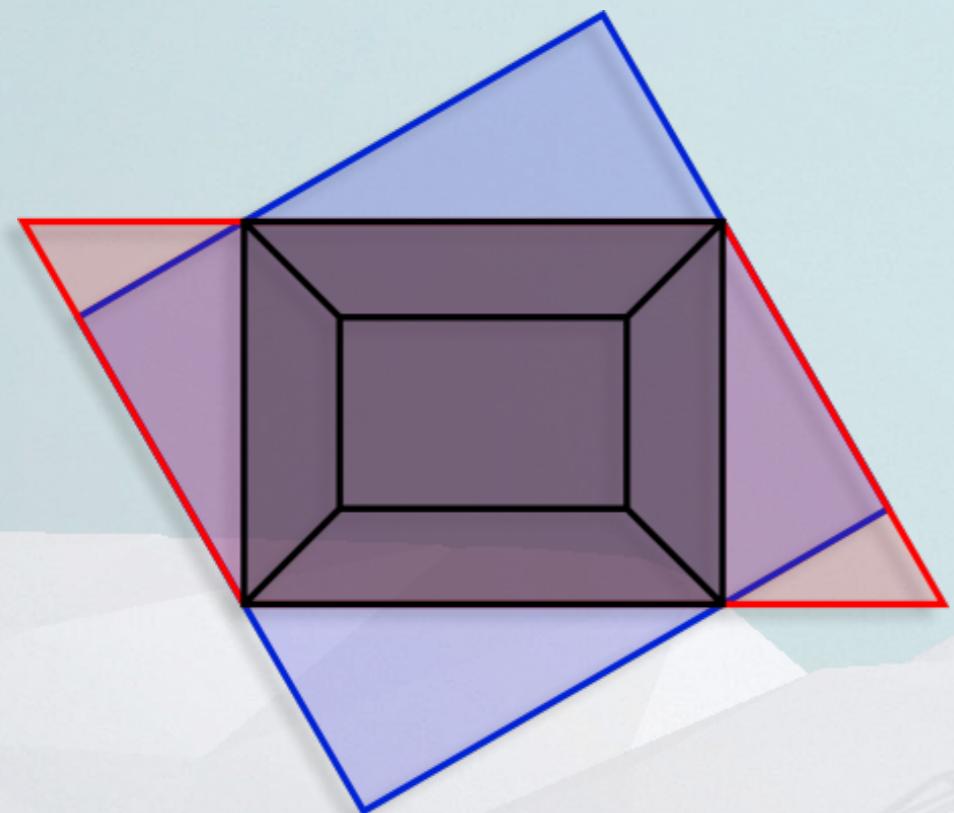


2.9 SDSM: SHEARED SAMPLE DISTRIBUTION SHADOW MAPS



2.9 SDSM: SHEARED SAMPLE DISTRIBUTION SHADOW MAPS (2)

- Szene oft sehr flach, geneigtes Licht
- Horizontale Near- und Farplane der Shadow Map



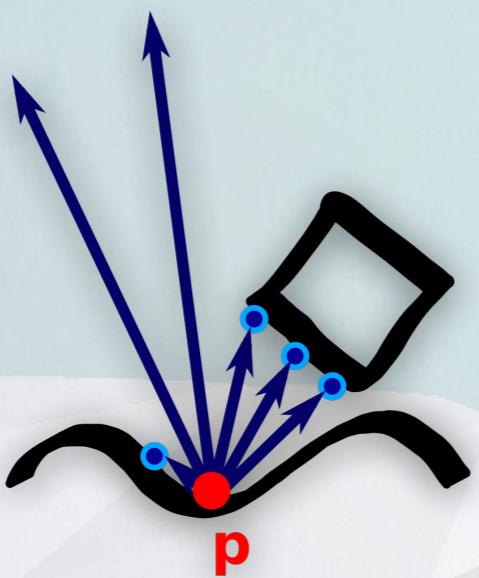
3. VOLUMETRIC OBSCURANCE

- Physikalisch korrekte Schattenberechnung in Echtzeit sehr aufwendig
- „Wo viel Geometrie ist, ist auch viel Schatten“
- Ambient Occlusion basiert auf diesem Prinzip
- Volumetric Obscurrence bietet Alternative zu AO

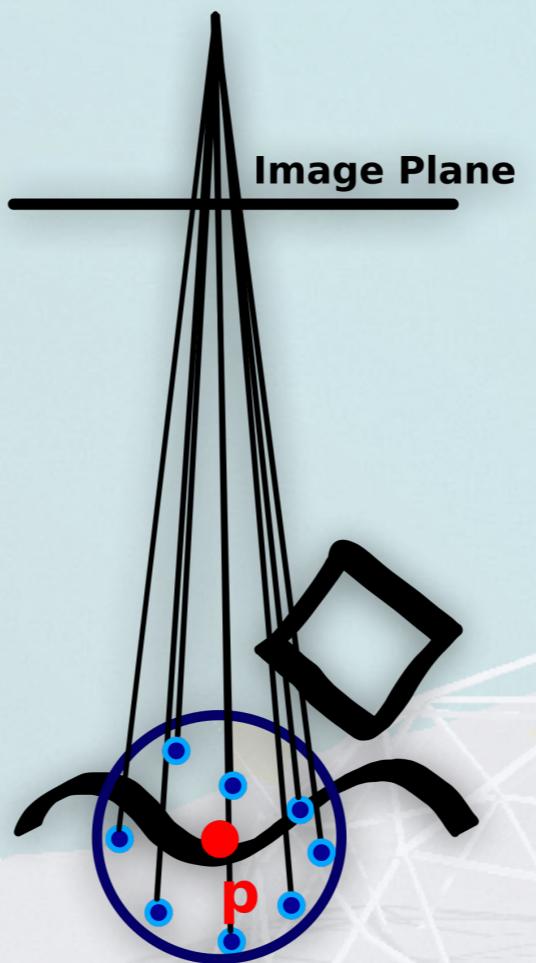
3. | VO: AMBIENT OCCLUSION

- Ambient Occlusion Ansätze:
 - (Global) Ambient Occlusion & Screenspace Ambient Occlusion

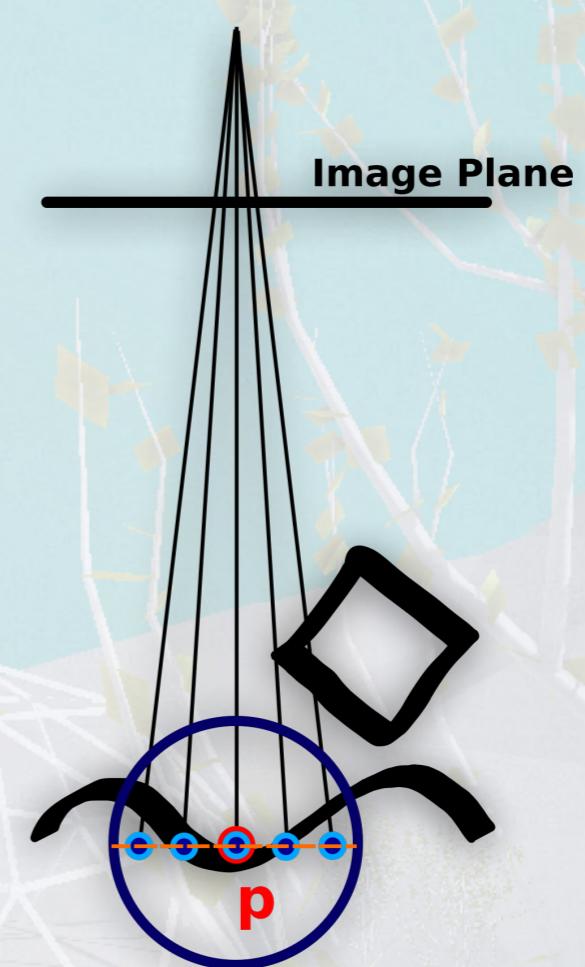
Ambient Occlusion



Screenspace Ambient Occlusion

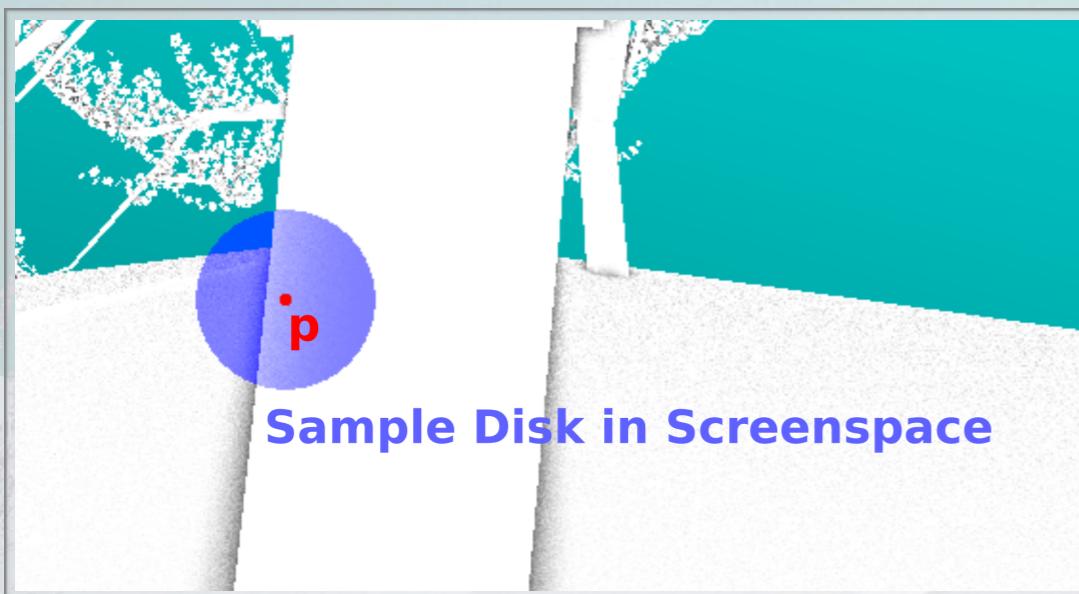


Volumetric Obscurance



3.2 VO: LINE SAMPLING

- Betrachte für jeden Texel den korrespondierenden Punkt
- Sample Tiefenbuffer in Screenspace projizierter Umgebung
- Berechne verdecktes Volumen



3.3 VO: LINE SAMPLING ALGORITHMUS

Line Sampling Pseudo Code

```
Function LineSampling(nSamples, depth)
    sumSamples = 0;
    sumVolume = 0;
    for i = 0 → nSamples - 1 do
        radius = rand();
         $\alpha$  = rand()· $2\pi$ ;
         $v_i$  =  $\sqrt{1 - radius^2}$ ;
        p = ( $\cos(\alpha) \cdot radius, \sin(\alpha) \cdot radius$ )T;
        sumSamples = sumSamples +  $v_i \cdot \text{SamplePoint}(p, depth)$ ;
        sumVolume = sumVolume +  $v_i$ ;
    end
    return  $\frac{\text{sumSamples}}{\text{sumVolume}}$  ;
end

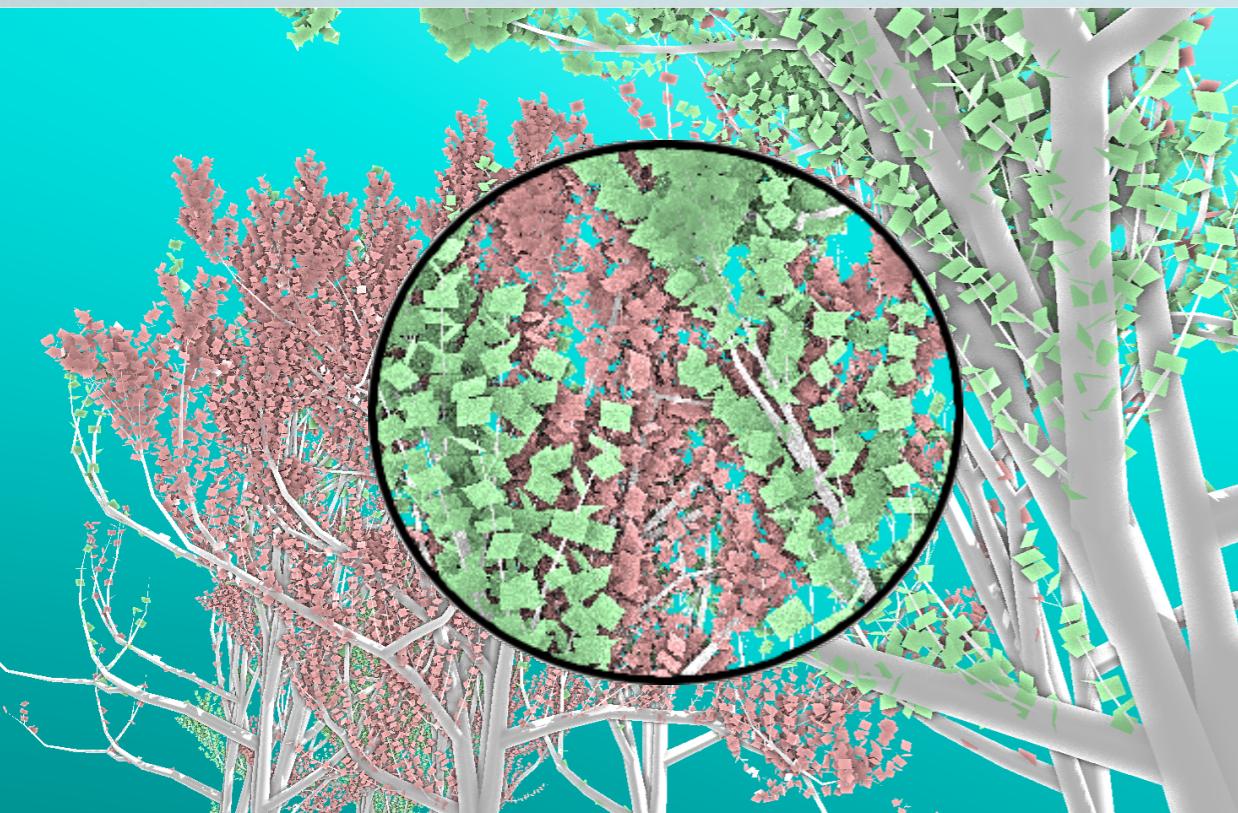
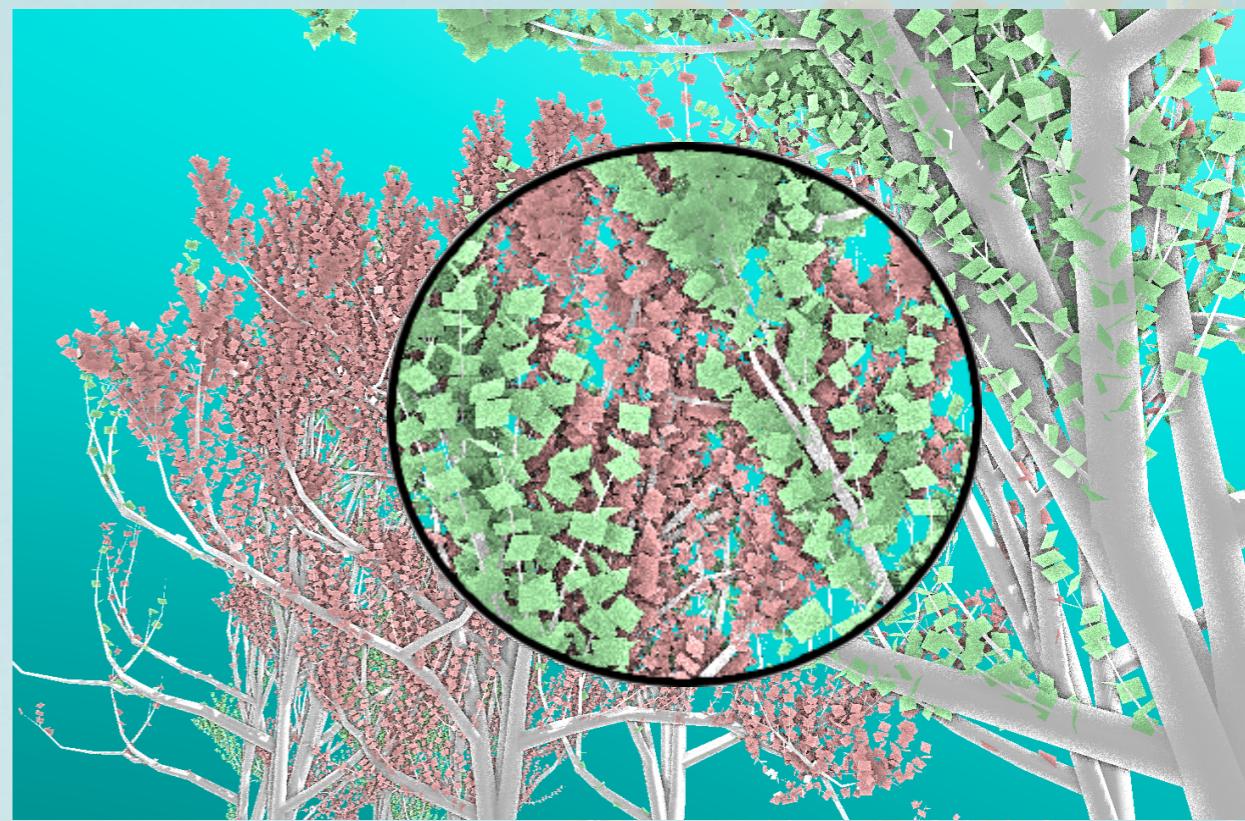
Function SamplePoint(p, depth)
    | hier wird ein einzelner Punkt gesampled, wie oben beschrieben
end

Function rand()
    | return zufälliger Wert im Intervall [0, 1];
end
```

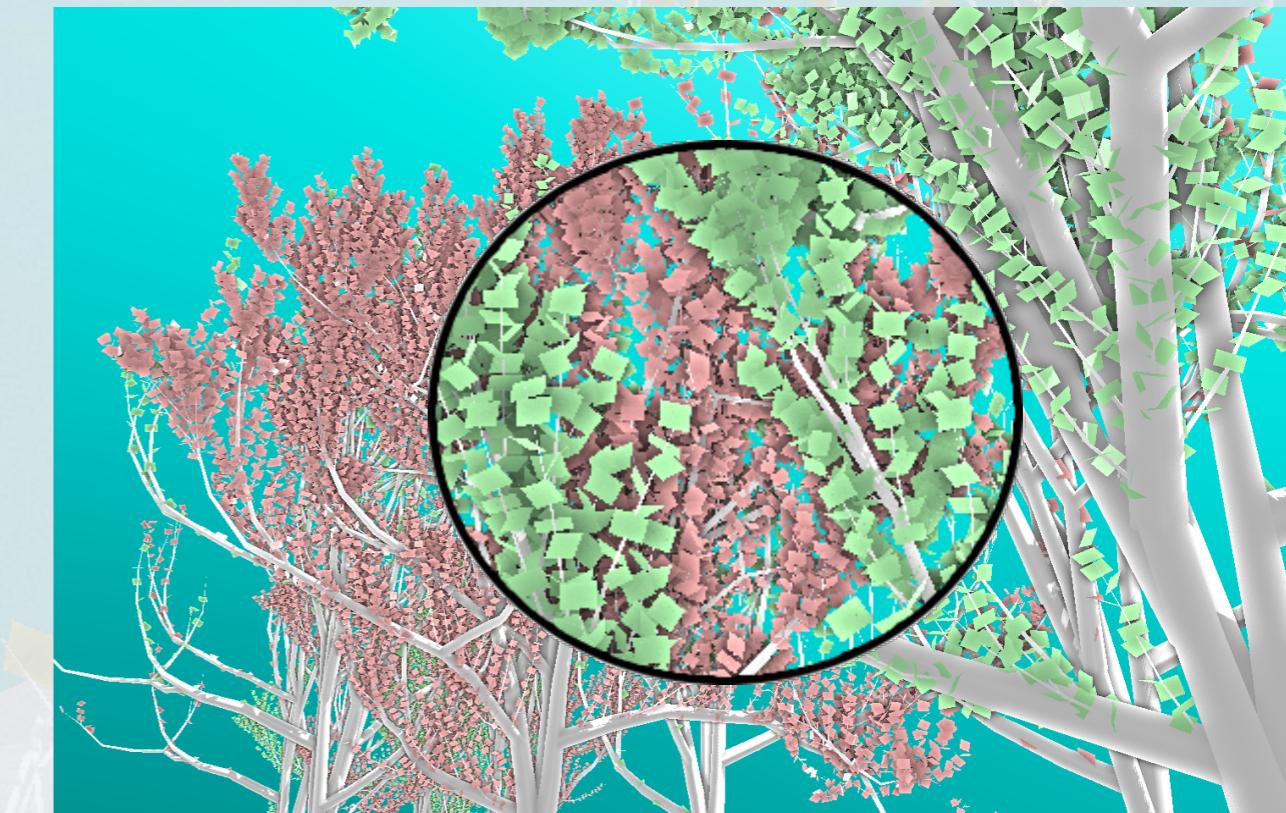
1 Samples



10 Samples



100 Samples



1000 Samples

3.4 VO: AREA SAMPLING

- Line Sampling teuer, da viele Samples pro Pixel
- Statistische Beschreibung der Tiefenverteilung die Zeitgleich genaue Rückrechnung ermöglicht
- VSM bietet gleiches Prinzip und kann auf VO angewendet werden

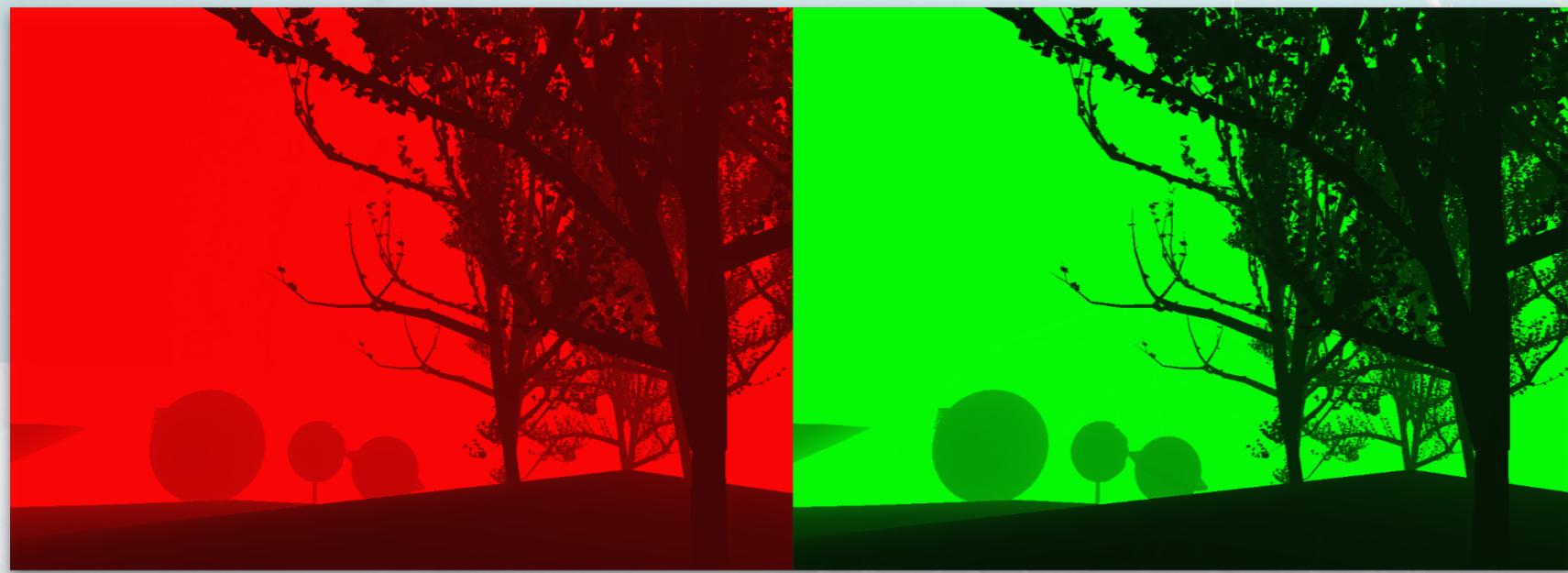
3.4 VO: AREA SAMPLING (2)

- Vorgehensweise: schreibe z, z^2 in 2-Kanal Textur
- Mip-mappe jene Textur
- Bestimme im Compose Schritt zu nutzenden Mip-Map Level abhängig von Tiefe
- Darüber wird Moment gefiltert

3.4 VO: AREA SAMPLING (3)

- Statistik zum Rückrechnen anwenden:
- $\mu = R, \sigma = G - \mu^2$
- $V_c(z_0, z_1, a, b) = a \cdot (z_1^2 - z_0^2)/2 + b \cdot (z_1 - z_0)$
- $a = \frac{-1}{2 \cdot \sigma}; \quad b = -a \cdot (\mu + \sigma)$

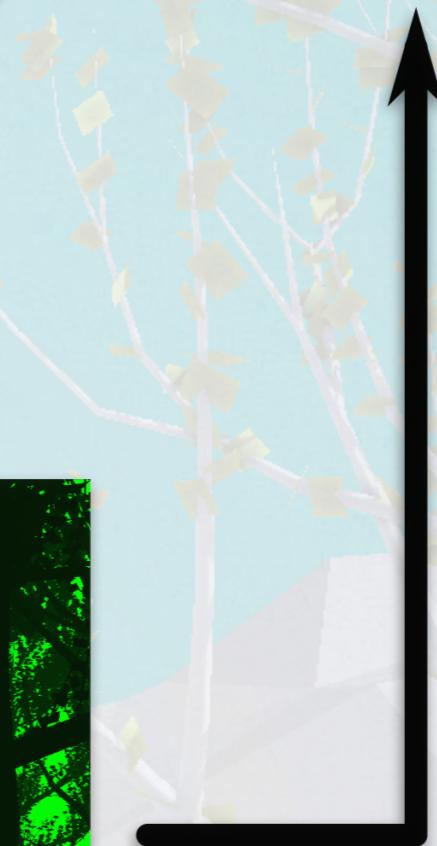
Depth (z)

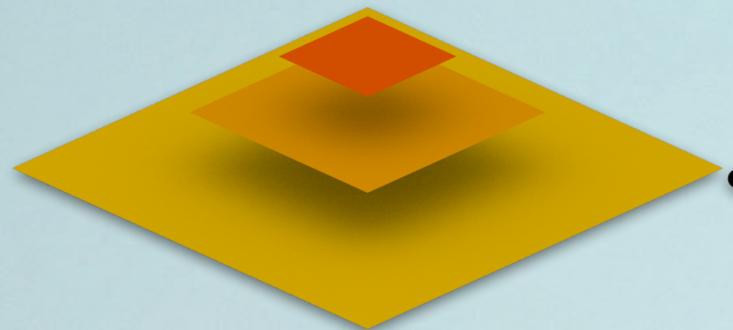


z^1

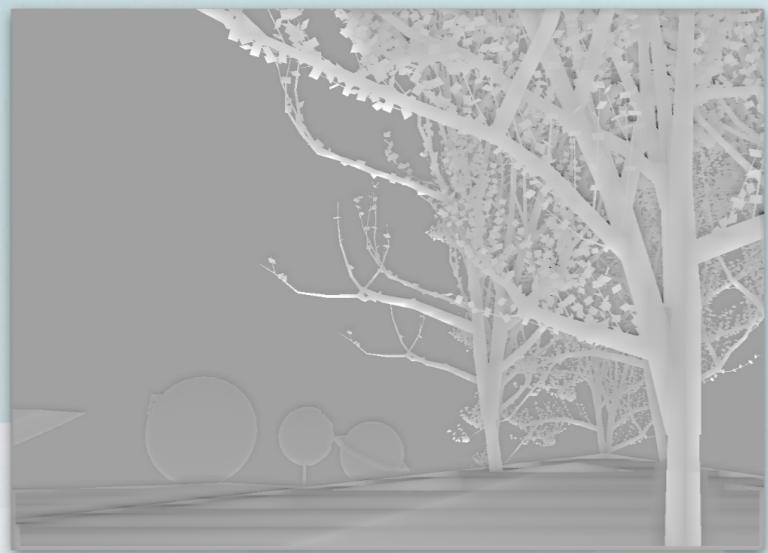
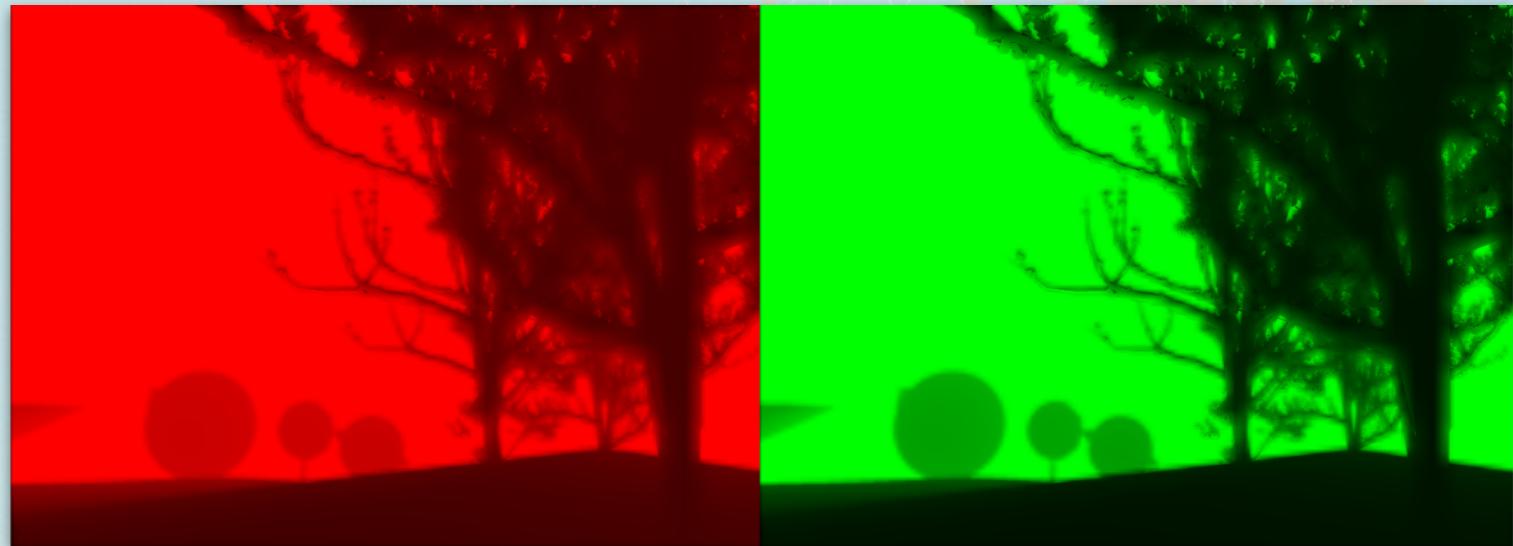
z^2

generate Mipmaps

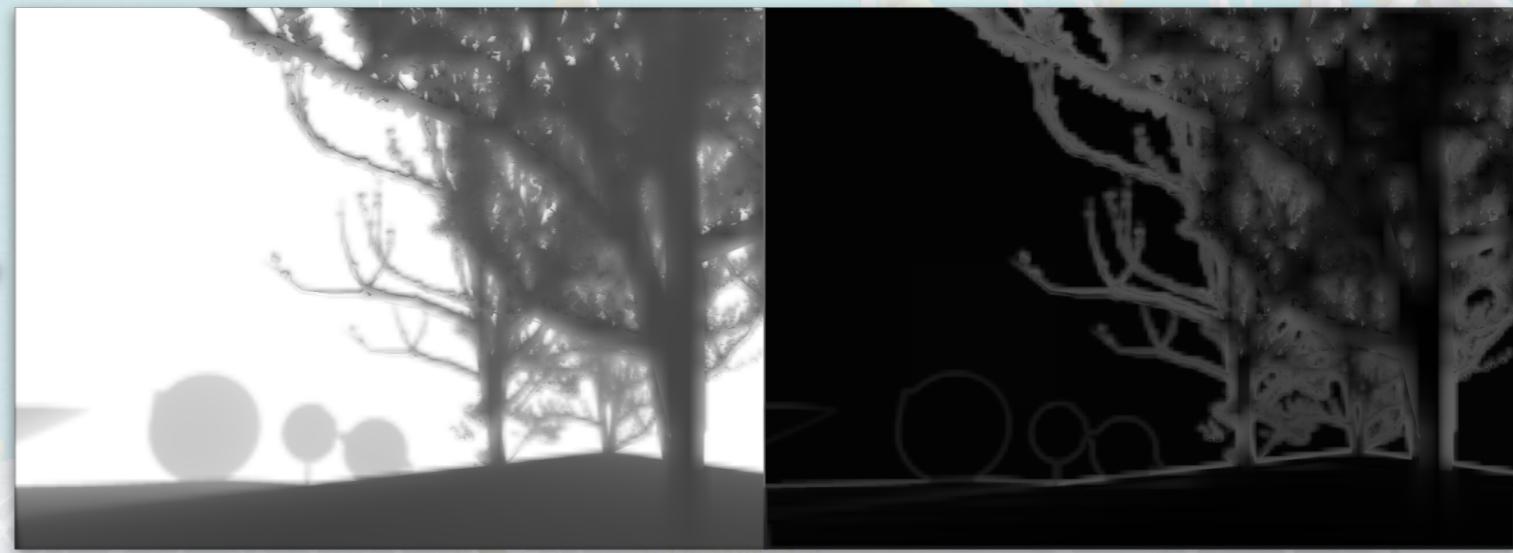




Mipmaps



obscurance Term

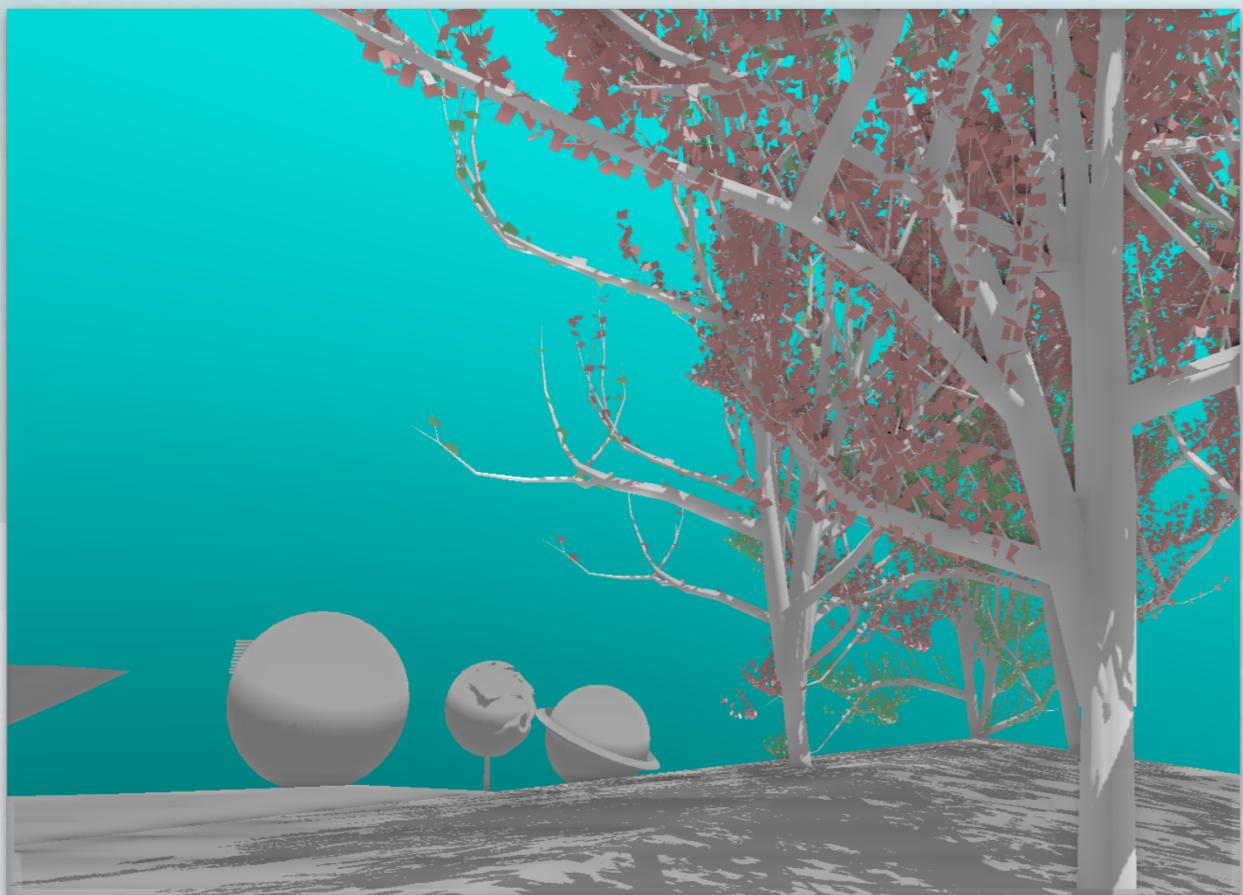


mean

variance

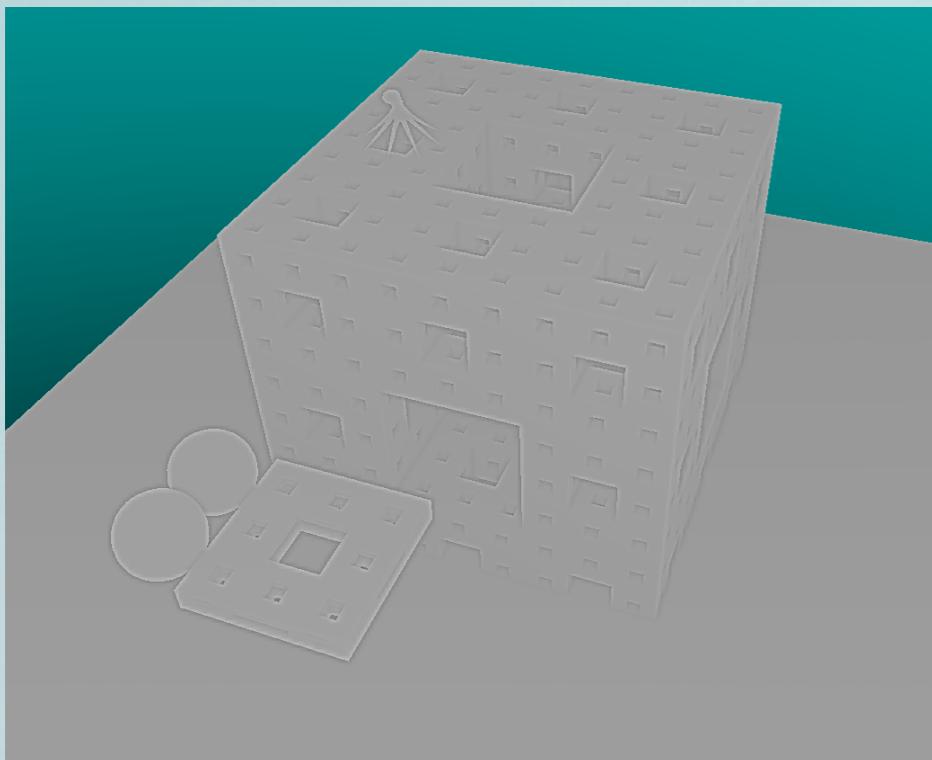


obscurance Term

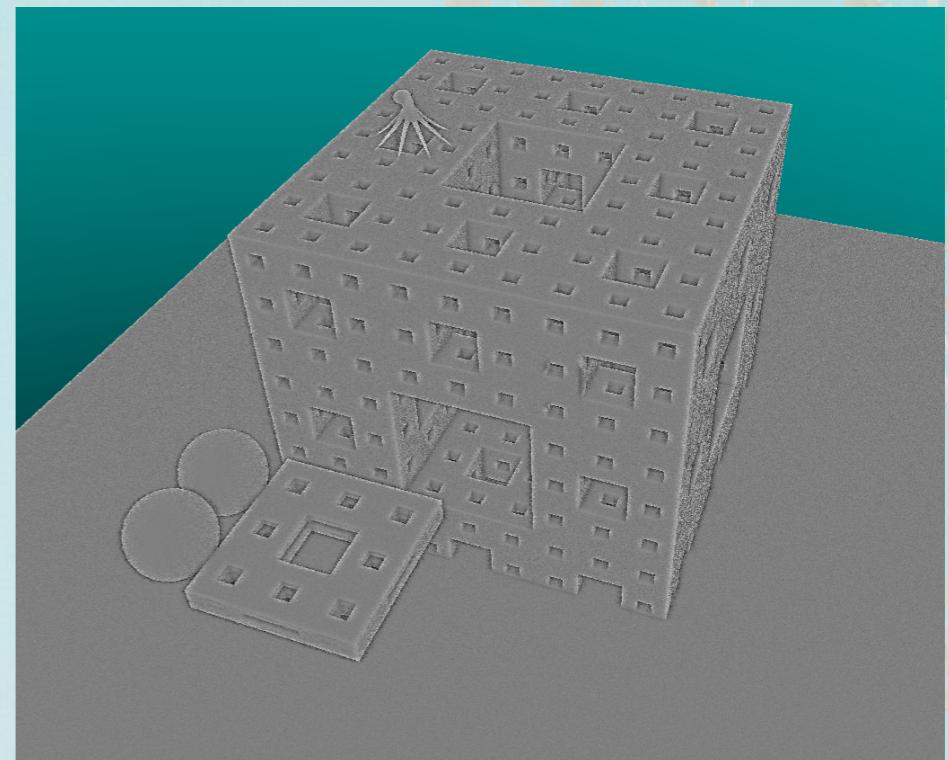


verrechnetes Bild

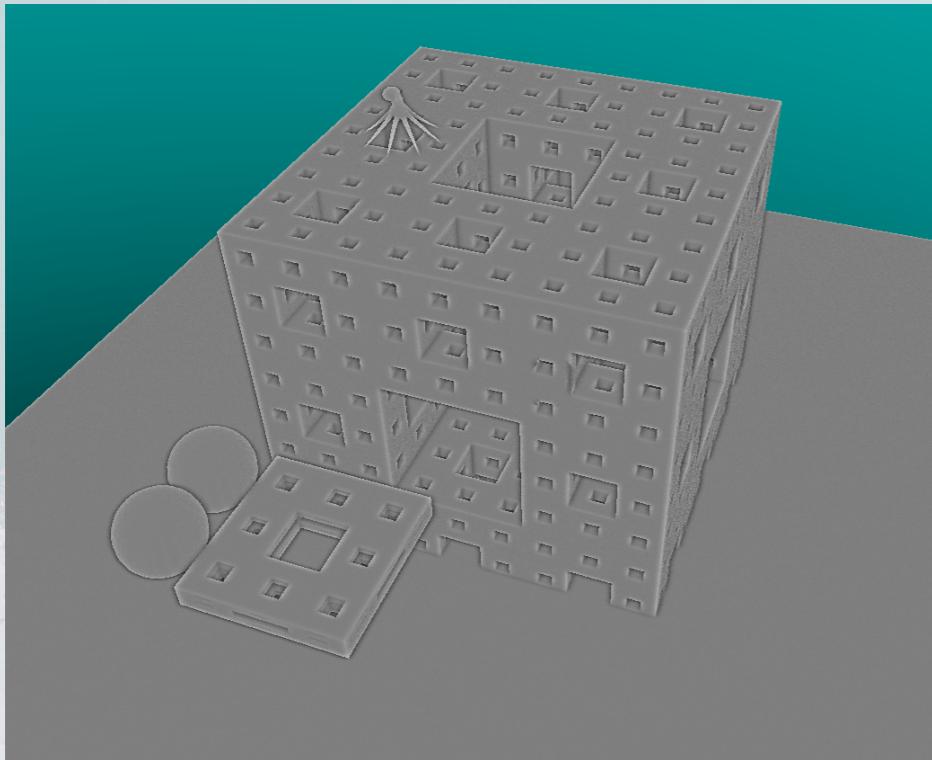
3.5 VO: ERGEBNISSE



ASVO



LSVO 10 Samples



LSVO 100 Samples



4. MOMENT SHADOW MAPPING

- Hohe Schattenqualität
- Filterbare Shadow Map
- Momente aus Tiefe generieren (ähnlich VSM)
- MSAA und Gauss Pre-Processing
- Moment Based Volumetric Obscurrence

4. I MSM: ALGORITHMUS

- Hamburger Moment Shadow Mapping
- Ähnlich VSM
- Untere Schranke , Schattenintensität wird nicht überschätzt
- Hamburger MSM liefert Schattenwert durch 4 Momente

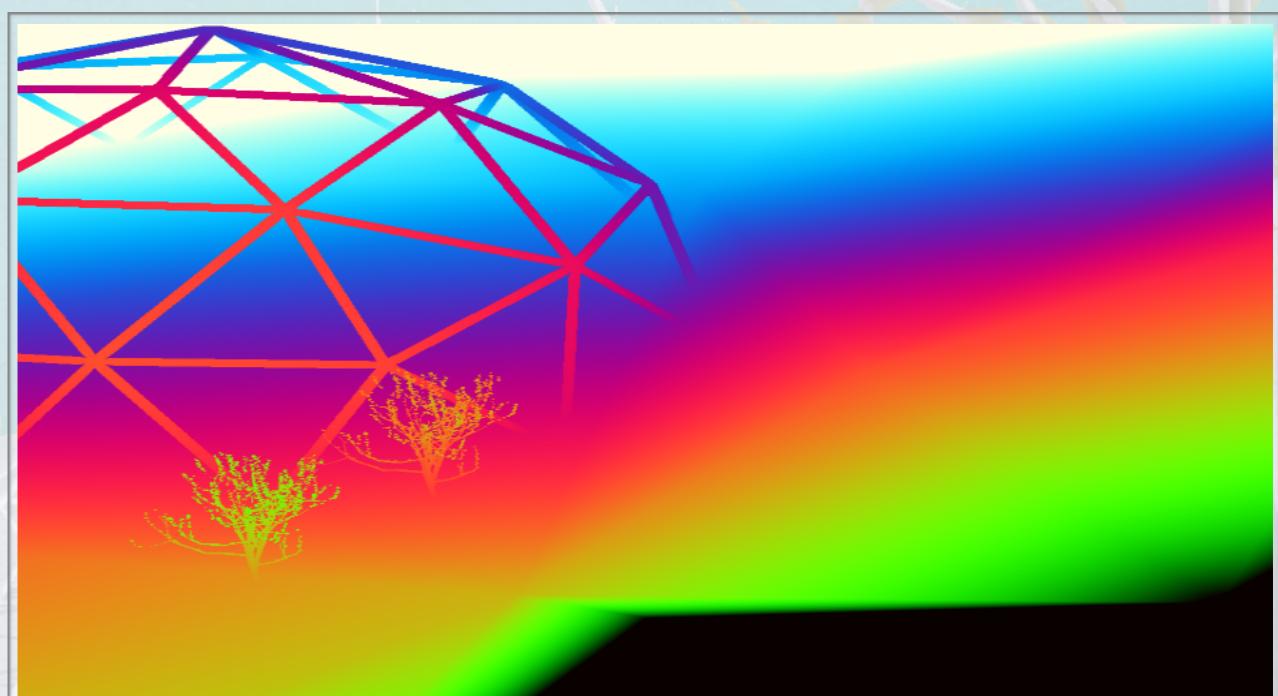
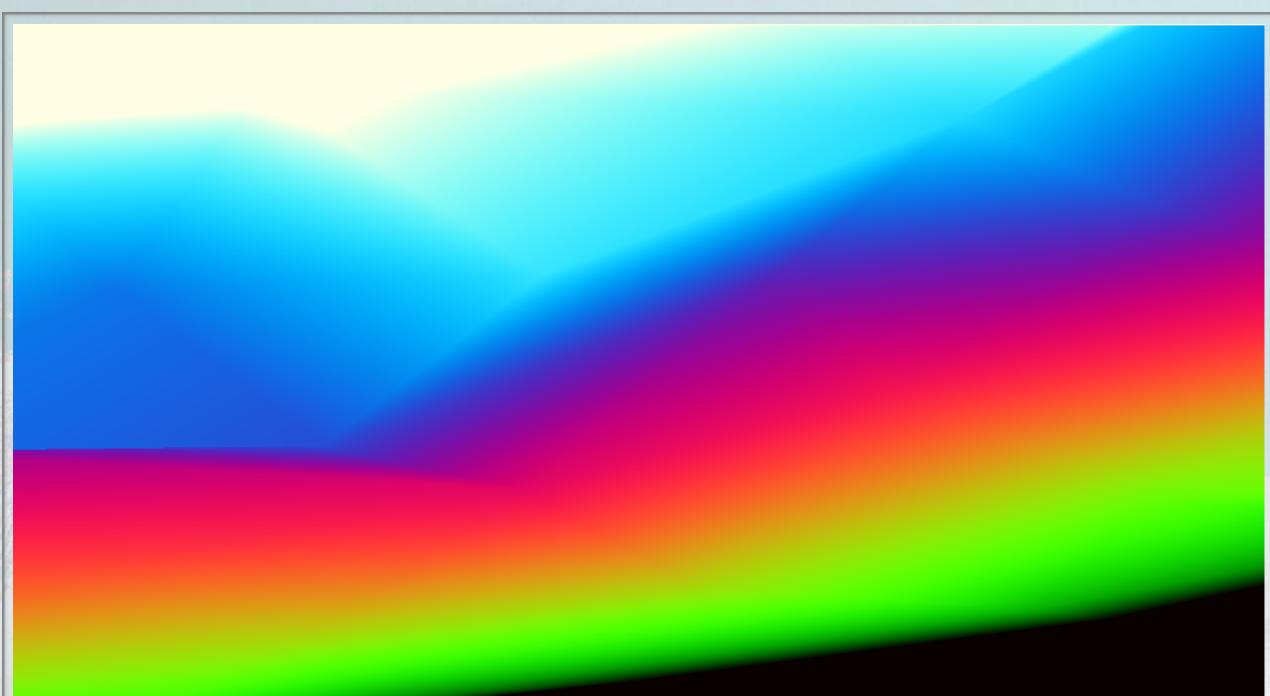
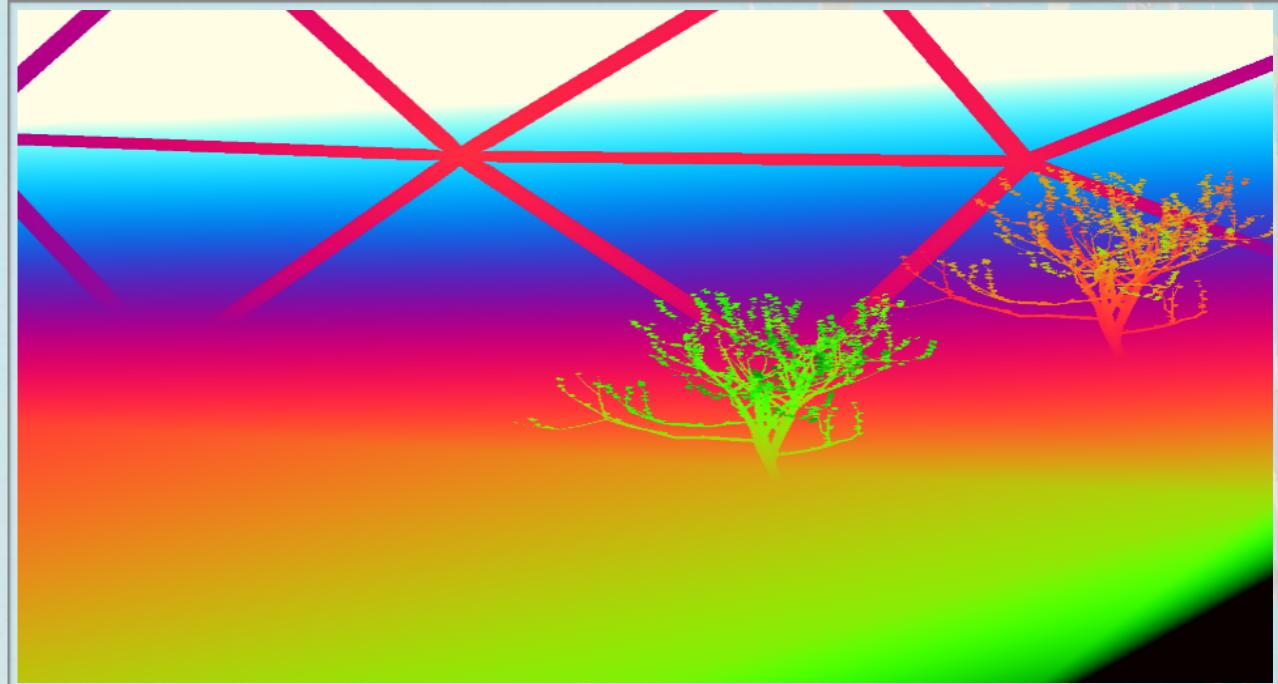
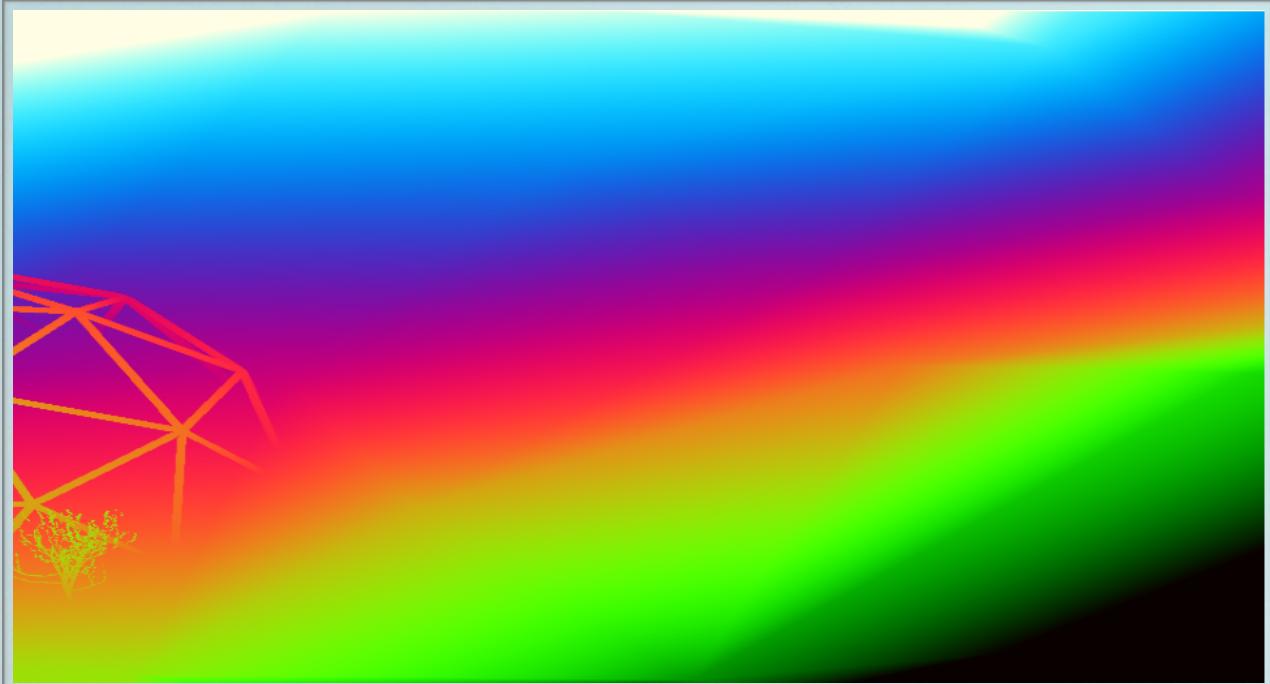
4.2 MSM: IMPLEMENTIERUNG

- Generieren der 4 Momente durch Tiefe z aus Tiefenbuffer
- Speichern in 4-Kanal Textur: z , z^2 , z^3 , z^4
- Matrixtransformation für 16-bit Quantifikation bess. Textur
- Rückrechnen beim lesen
- Berechnen der Schattenintensität



4.3 MSM: MOMENT TEXTUR

- Speichern in Textur



4.4 MSM: ERGEBNISSE



4.4 MSM: ERGEBNISSE (2)

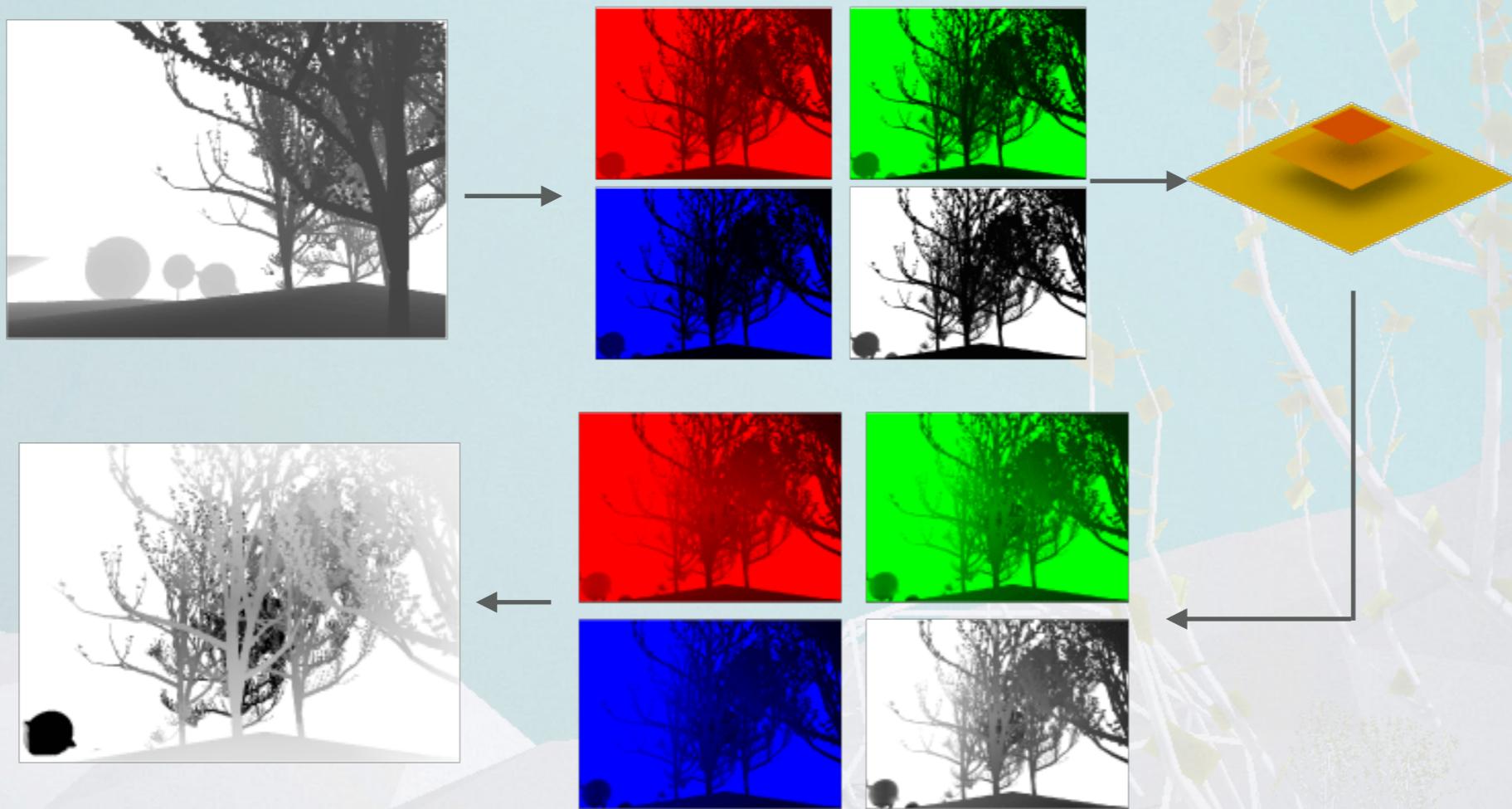
- Anti-Aliasing durch MSAA und Filtern durch Gauss
- Light Bleeding wird besser verringert als bei VSM
- Kaum Auswirkungen auf Engine-Performance, das Filtern allerdings hohe Einbrüche

4.5 MSM: MBVO ALGORITHMUS

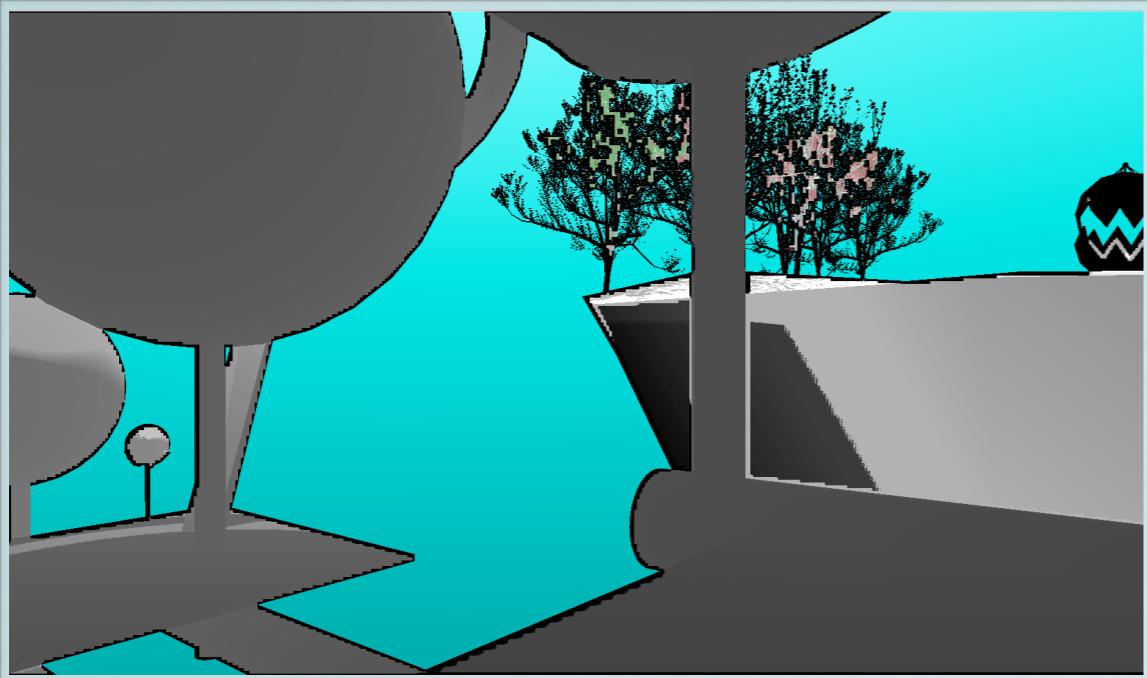
- Moment Based Volumetric Obscurrence
- Nutzbar durch MSM Technik
- Algorithmus liefert Term für VO

4.6 MBVO: IMPLEMENTIERUNG

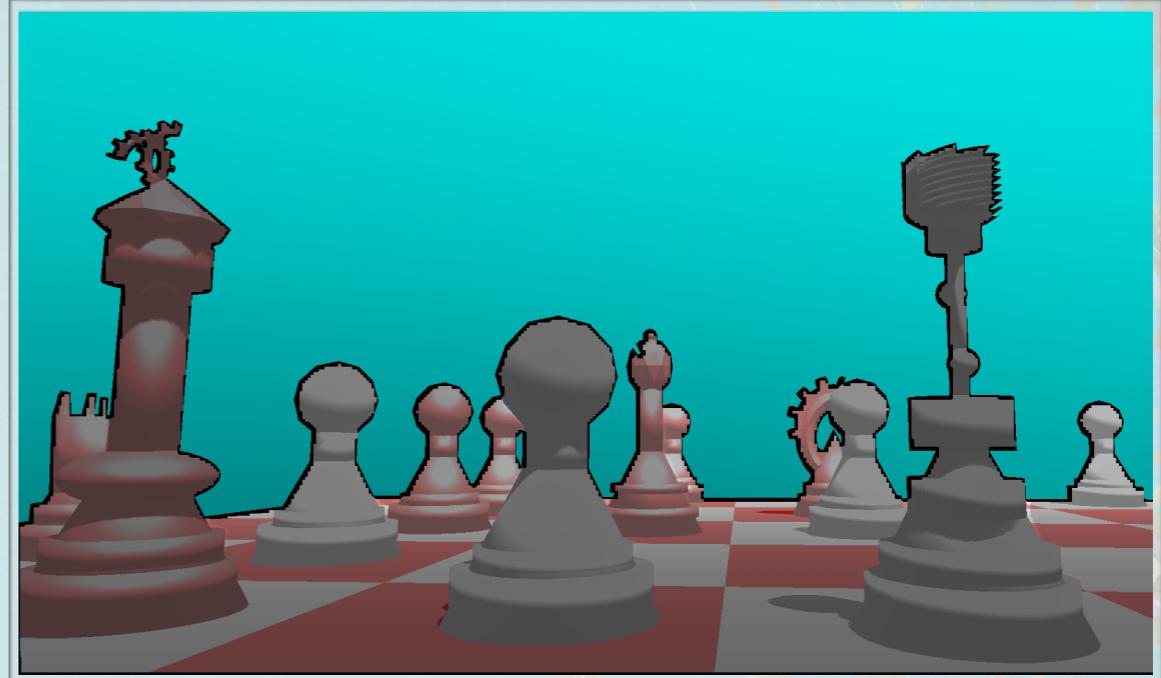
- Moment Based Volumetric Obscurrence Term berechnen
- Ähnlicher Aufbau wie für MSM
- Funktion der Intensität wird erweitert um Gewichte



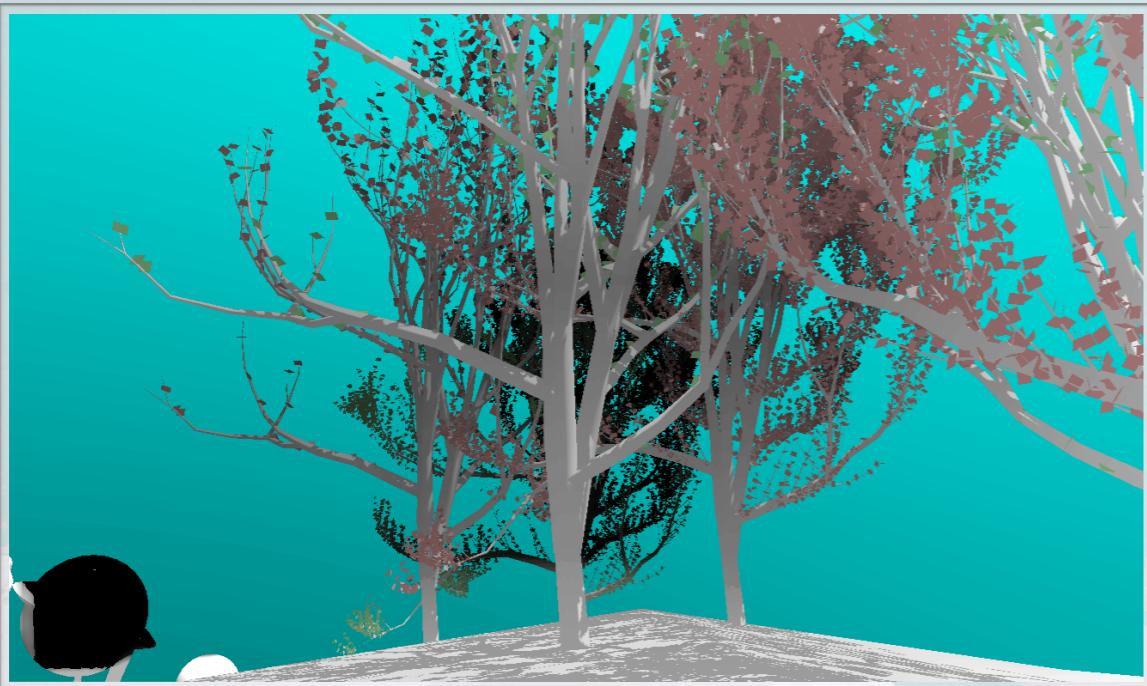
4.7 MBVO: ERGEBNISSE



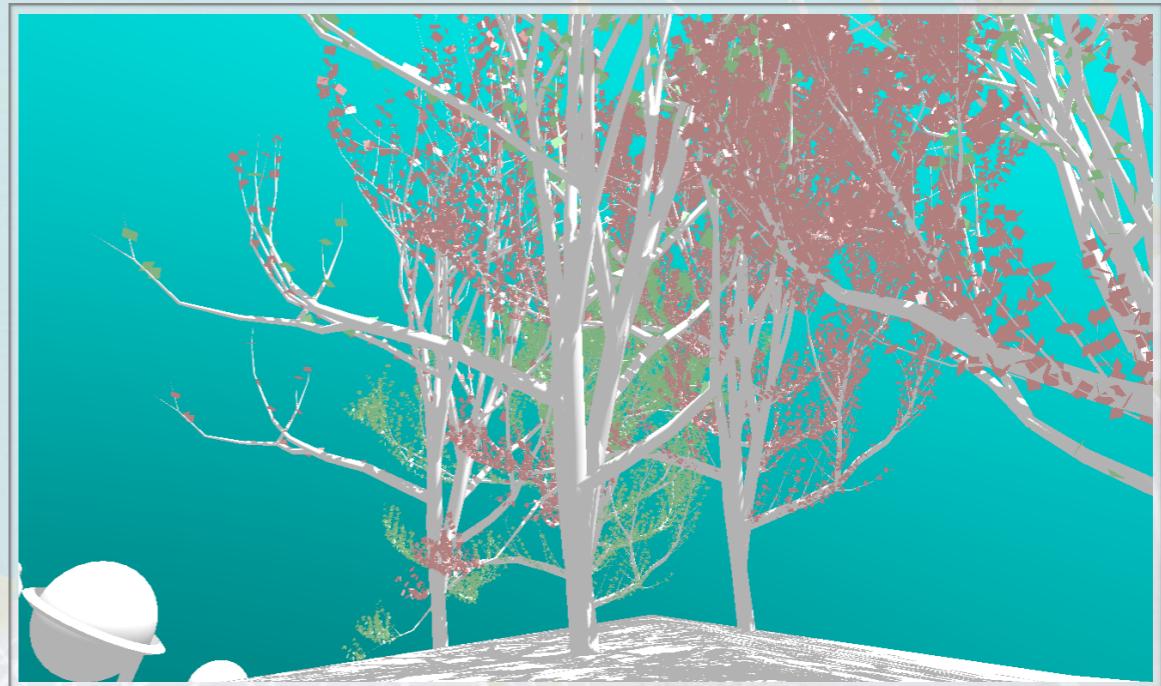
MBVO I



MBVO 2



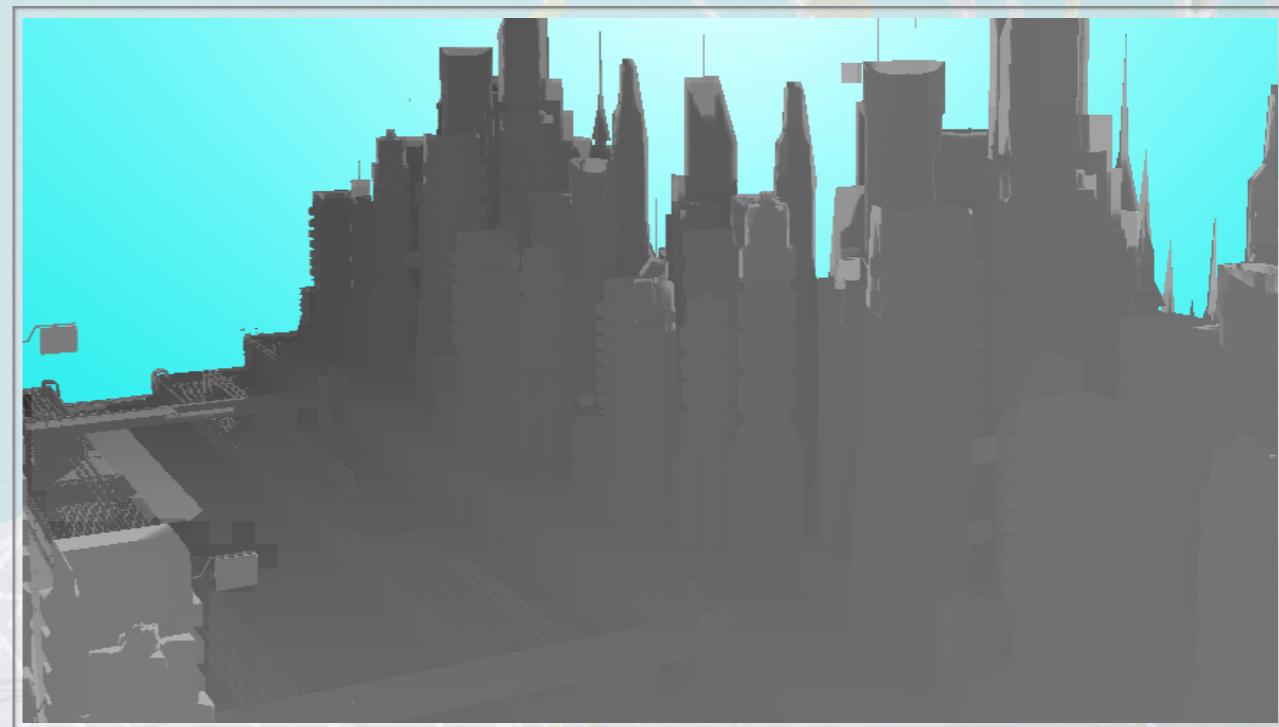
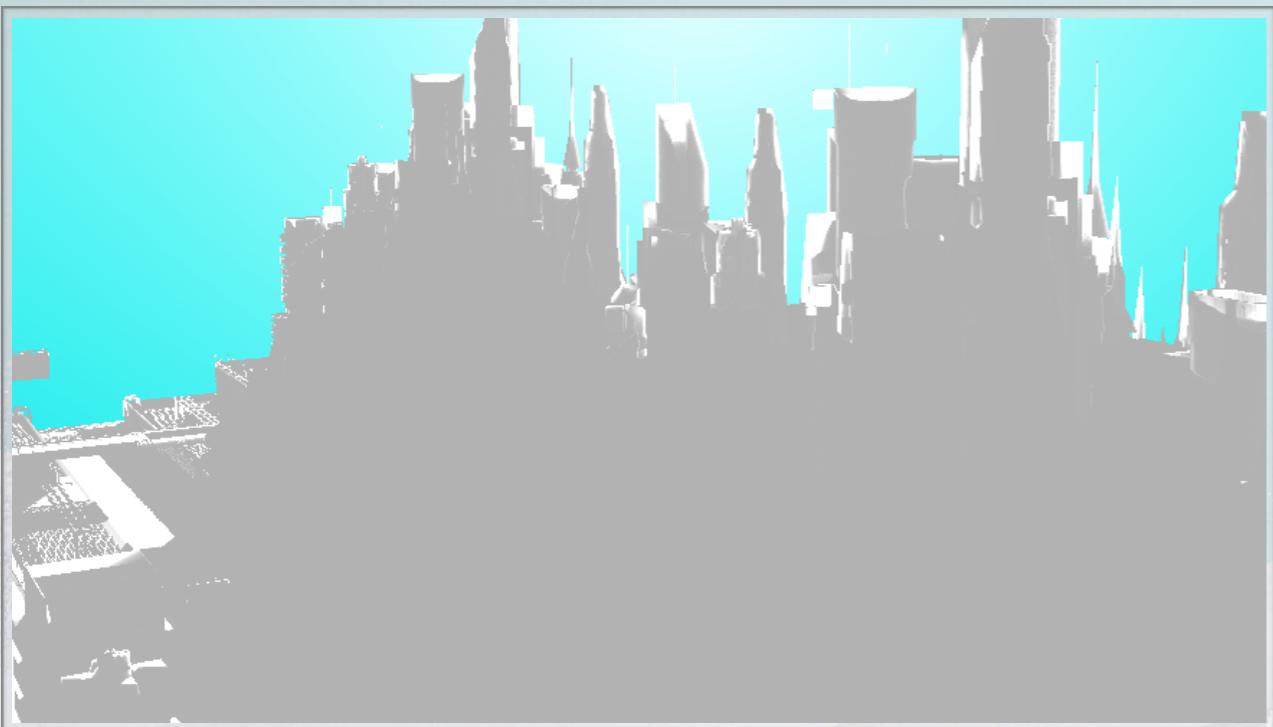
MBVO ON



MBVO OFF

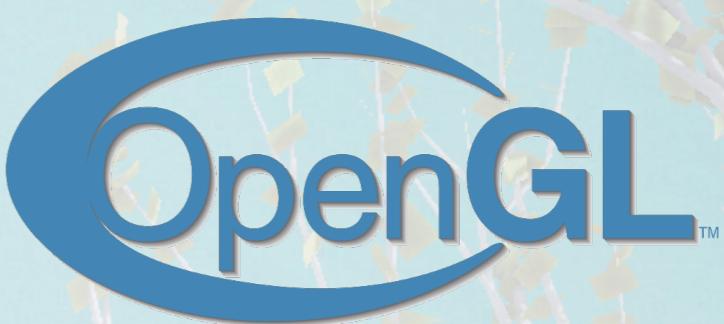
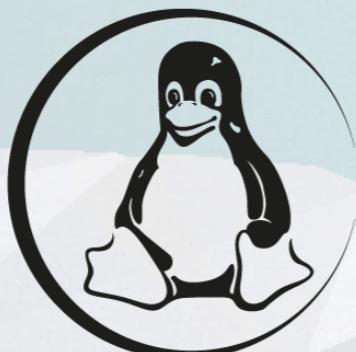
4.7 MBVO: ERGEBNISSE (2)

- Leicht umsetzbar durch Moment Technik
- Cell Shading ähnlicher Effekt mit falschen Werten
- Weiteres experimentieren nötig um ästhetisches Bild zu erhalten



5. SUSHI! WITH R.I.C.E.

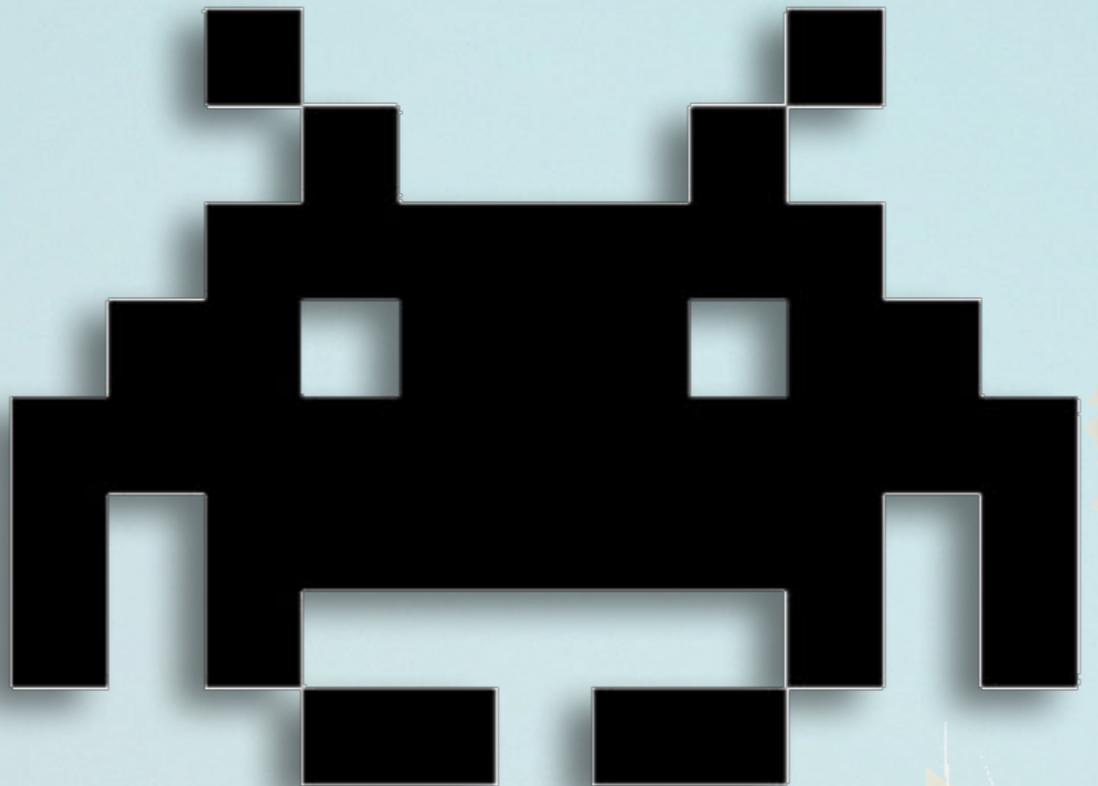
- Realistic Insane Computer-Graphic Environment
- C++, Qt, OpenGL, Bullet Physics
- Linux und Windows Support
- Sushi! Game Demo



5. ENTWICKLERTOOLS

- EDITOR PICS

5. I SUSHI!



PRESS START



6. FAZIT UND AUSSICHTEN

- Spiel soll fertig entwickelt werden
 - weitere Techniken und Verbesserungen
 - weitere Levels, Rätsel, Menüs und Optionen
- Planung vs. Chaos
- Wir haben viel gelernt
- Wir hatten jede Menge Spaß



VIELEN DANK
FÜR DIE
AUFMERKSAMKEIT