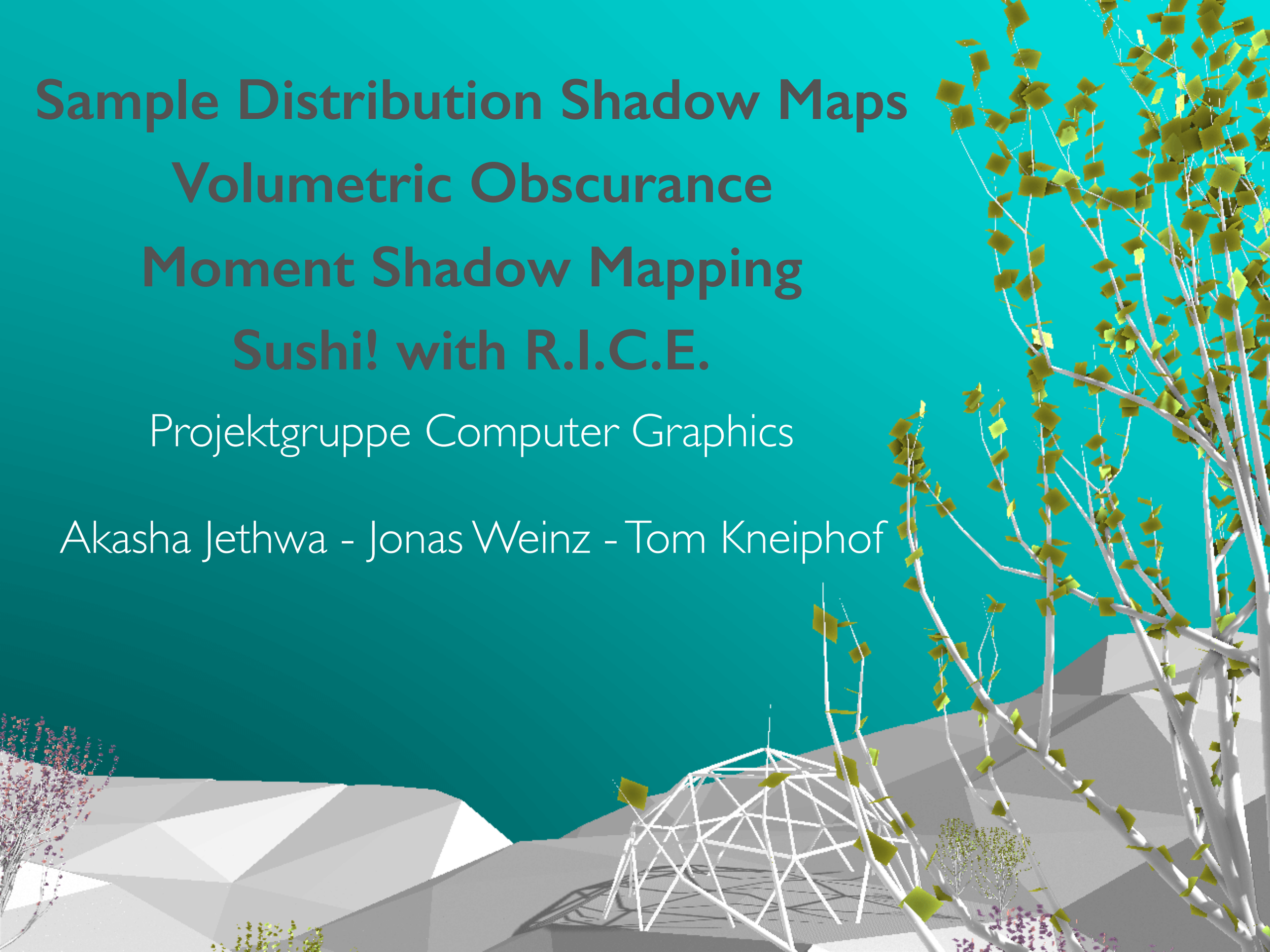
Sample Distribution Shadow Maps
Volumetric Obscurance
Moment Shadow Mapping
Sushi! with R.I.C.E.

Projektgruppe Computer Graphics

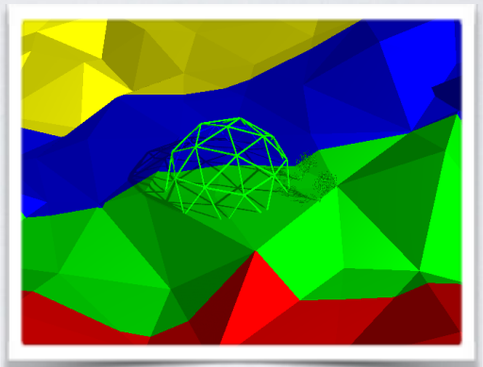
Akasha Jethwa - Jonas Weinz - Tom Kneiphof

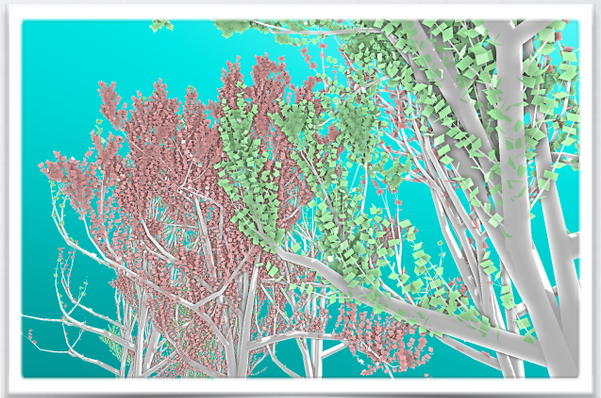


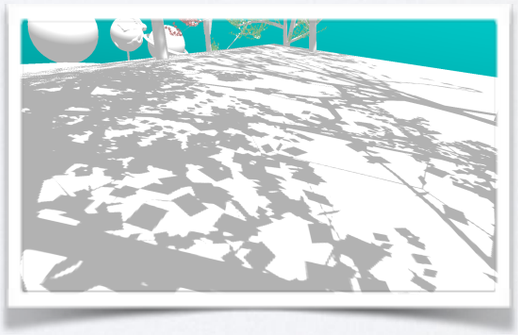
INHALT

TECHNIKEN

DEMO







2. SAMPLE DISTRIBUTION SHADOW MAPS

- Vermeide Oversampling, Undersampling, Geometrisches Aliasing

- Basierend auf Cascaded Shadow Maps

- Partitionieren Kamera Frustum entlang Z-Achse

- Vollautomatische Partitionierung

2.1 SDSM: NEAR & FAR PLANE

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

2.2 SDSM: DEPTH PARTITIONING

- Logarithmisch

- Adaptiv logarithmisch
  -Vermeidung von Lücken im Depth-Buffer Histogramm

- K-Means
  -Partitionieren um Maxima des Depth-Buffer Histogramms

2.3 SDSM: TIGHT PARTITION FRUSTA / LIGHT SPACE

- Shadow Map Frusta eng um sichtbare Geometrie
- Rotiere Light Direction auf Z-Achse
- Rotiere Kamera Z-Achse um die resultierende Z-Achse auf X-Achse
- Berechne AABB der sichtbaren Geometrie per Reduktion

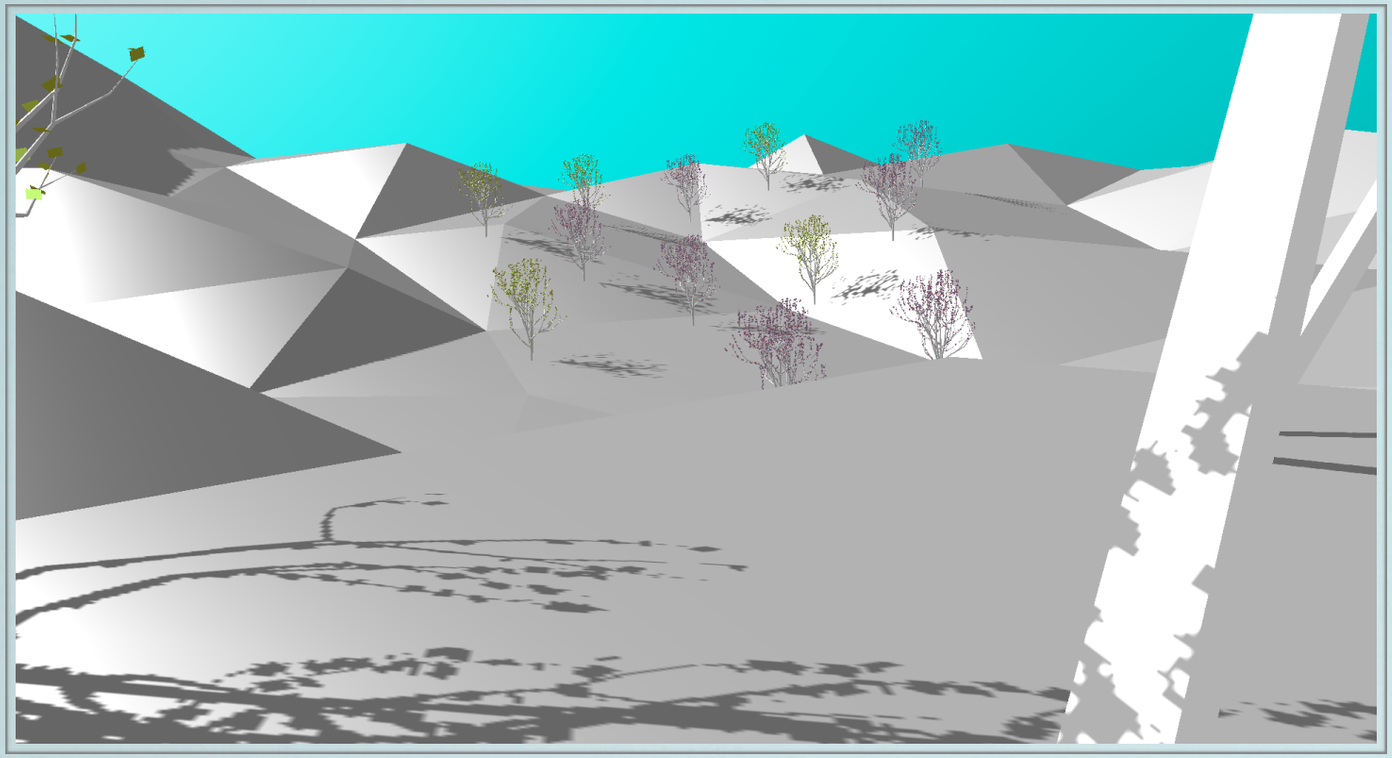
2.4 SDSM: REDUKTION

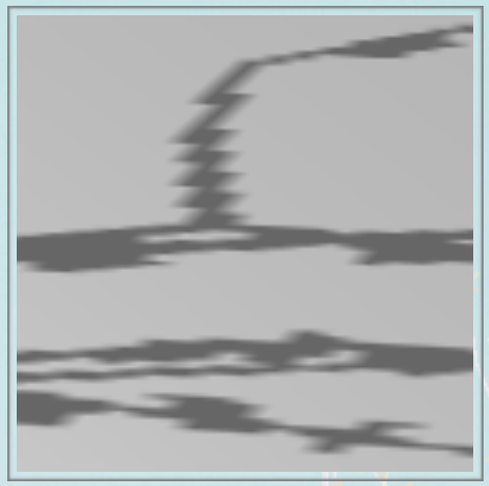
- Betrachte Minimierungsprobleme
- Reduziere mehrere Werte simultan
- Nutze 1D Textur mit mehreren Kanälen
- Eine Ausgabe pro Threadgruppe

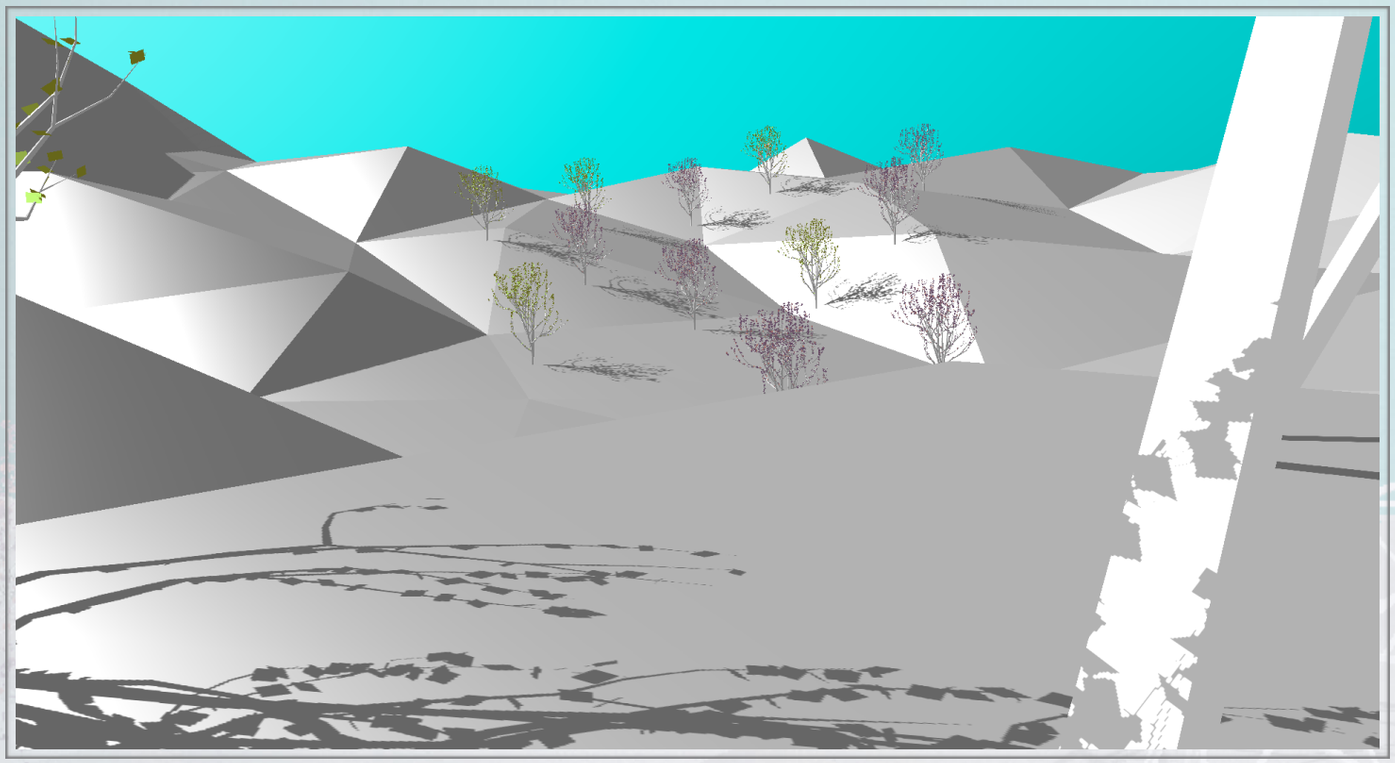
2.5 SDSM: PARALLEL SPLIT SHADOW MAPS

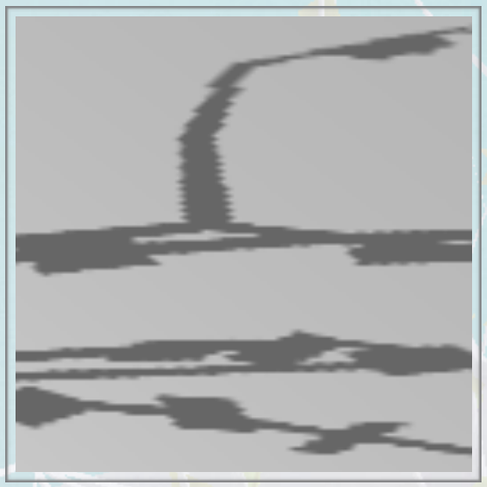
- Near- und Far-Plane unbekannt
- Verschiebe Grenzen von der Kamera weg
- Konvexkombinierte Kaskadengrenzen linearer und logarithmischer Partitionierung

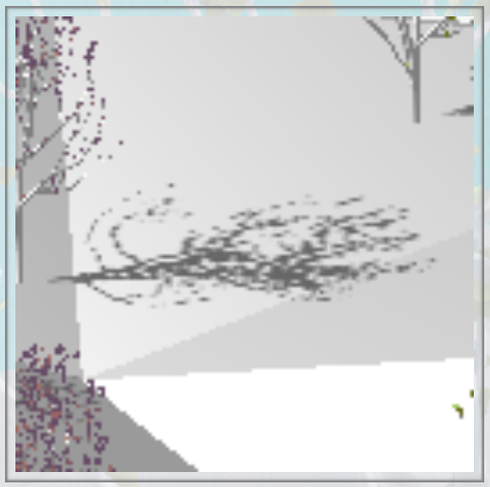
PSSM





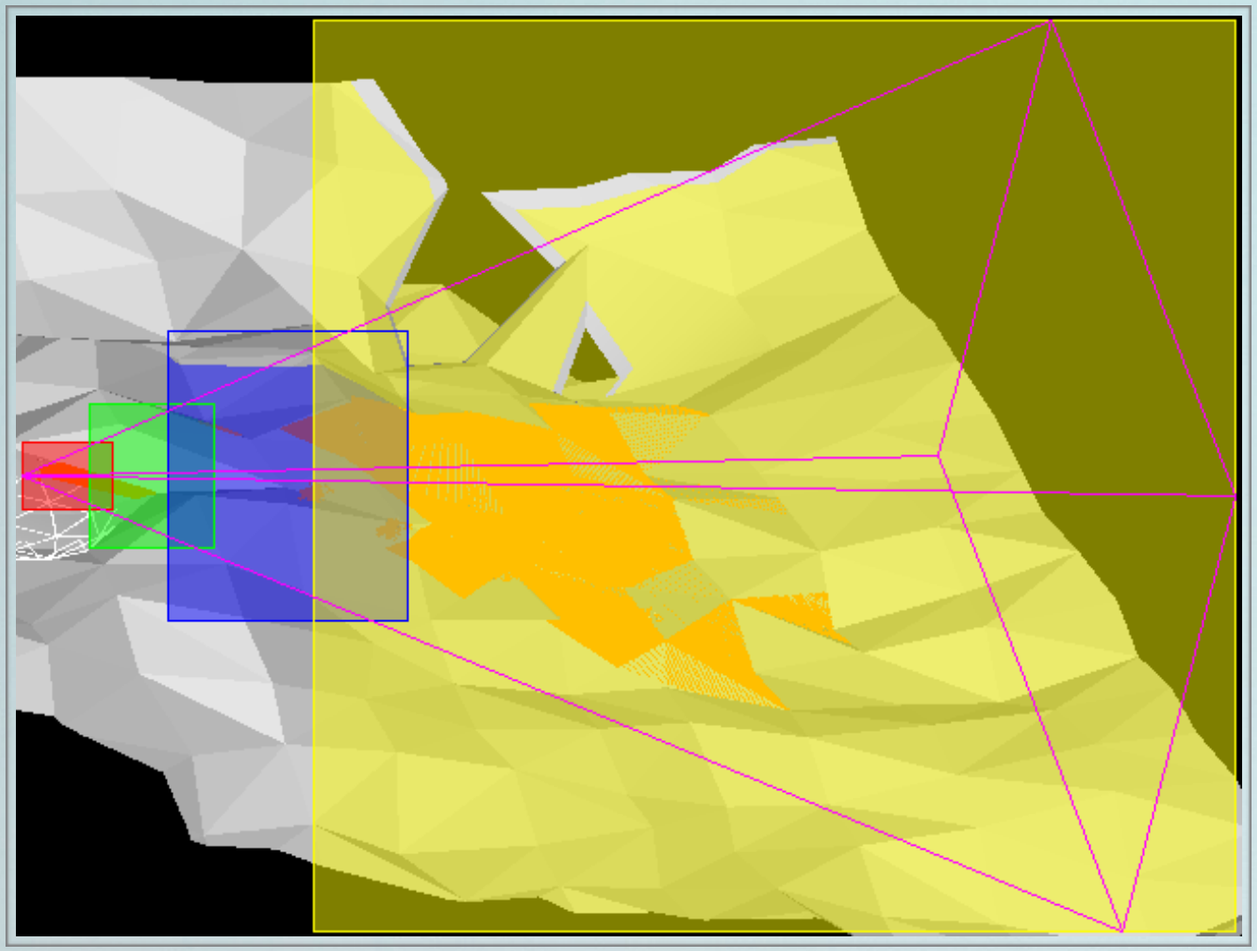




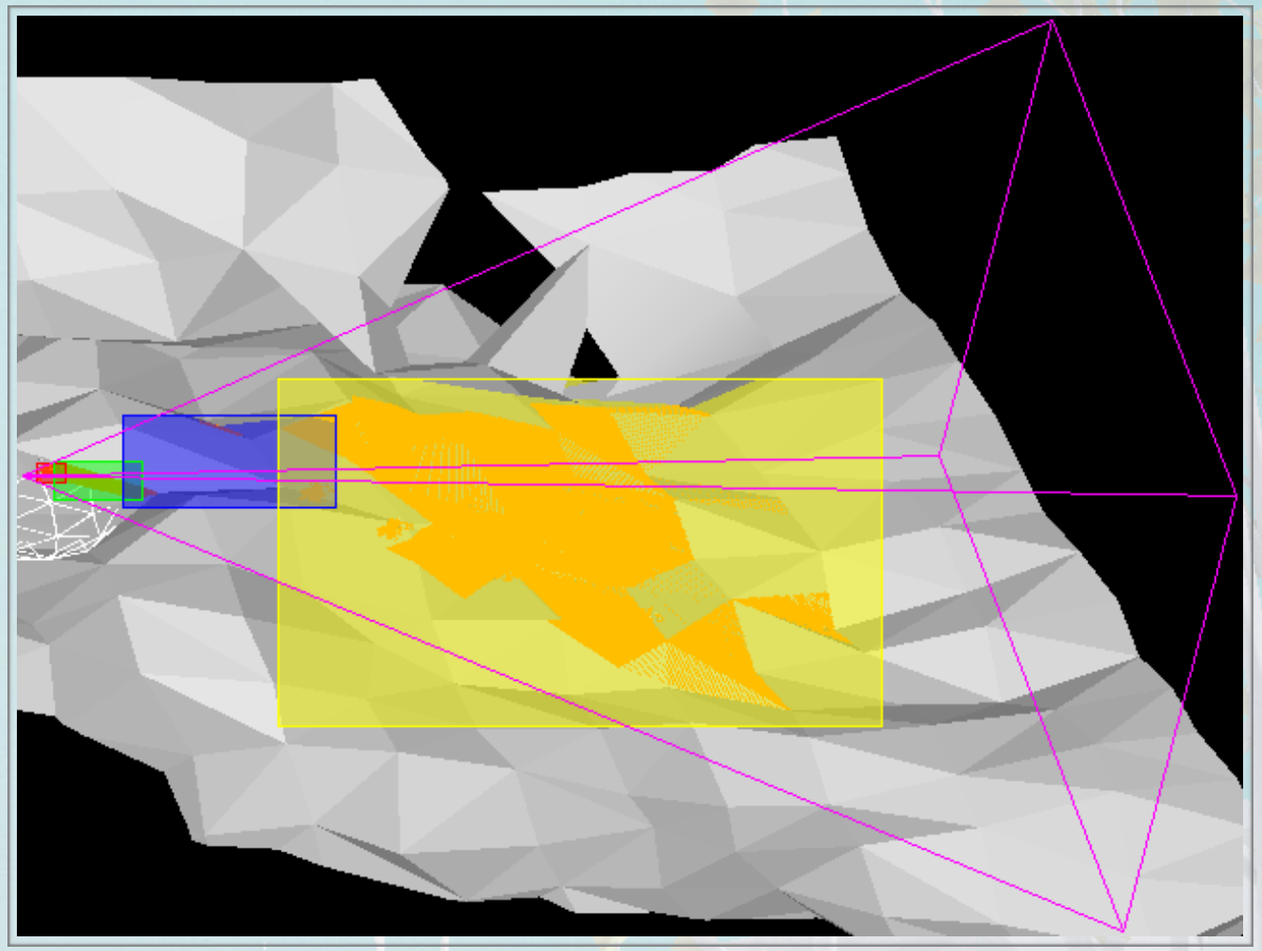


SDSM

2.6 SDSM: LIGHT SPACE FRUSTA

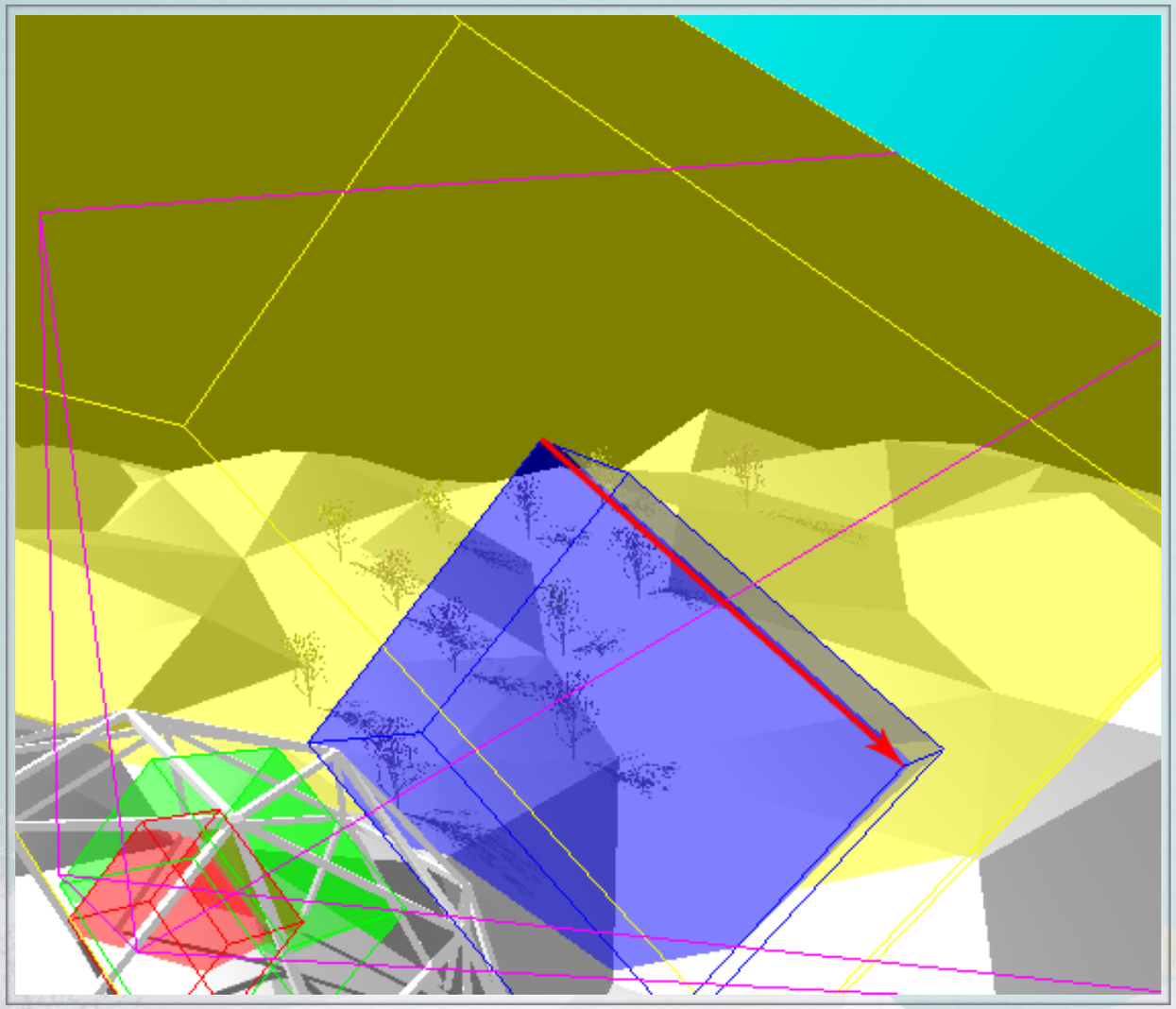


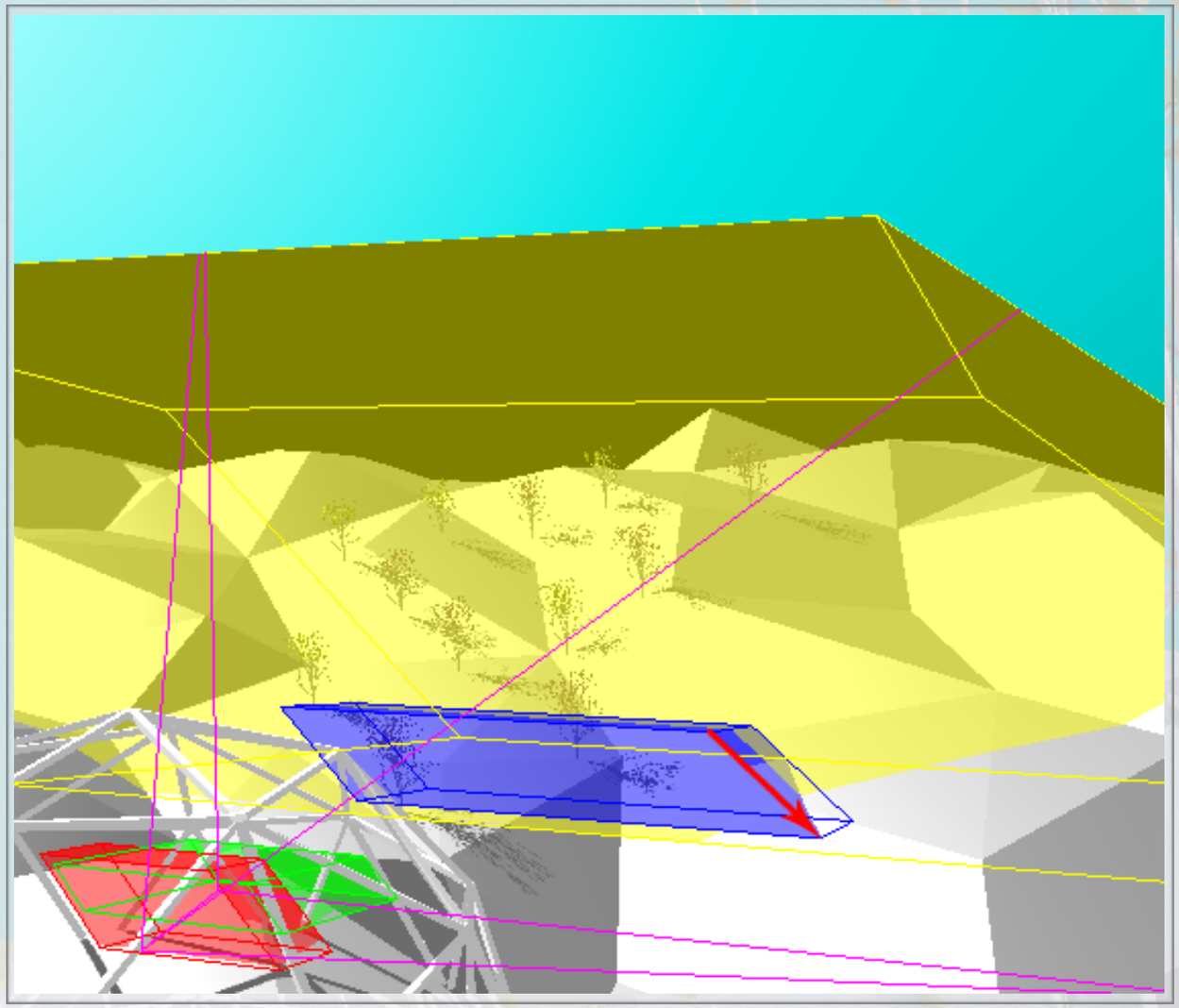
PSSM



SDSM

2.7 SDSM: SHEARED SAMPLE DISTRIBUTION SHADOW MAPS





3. VOLUMETRIC OBSCURANCE

- physikalisch korrekte Schattenberechnung in Echtzeit sehr aufwendig

- „Wo viel Geometrie ist, ist auch viel Schatten“

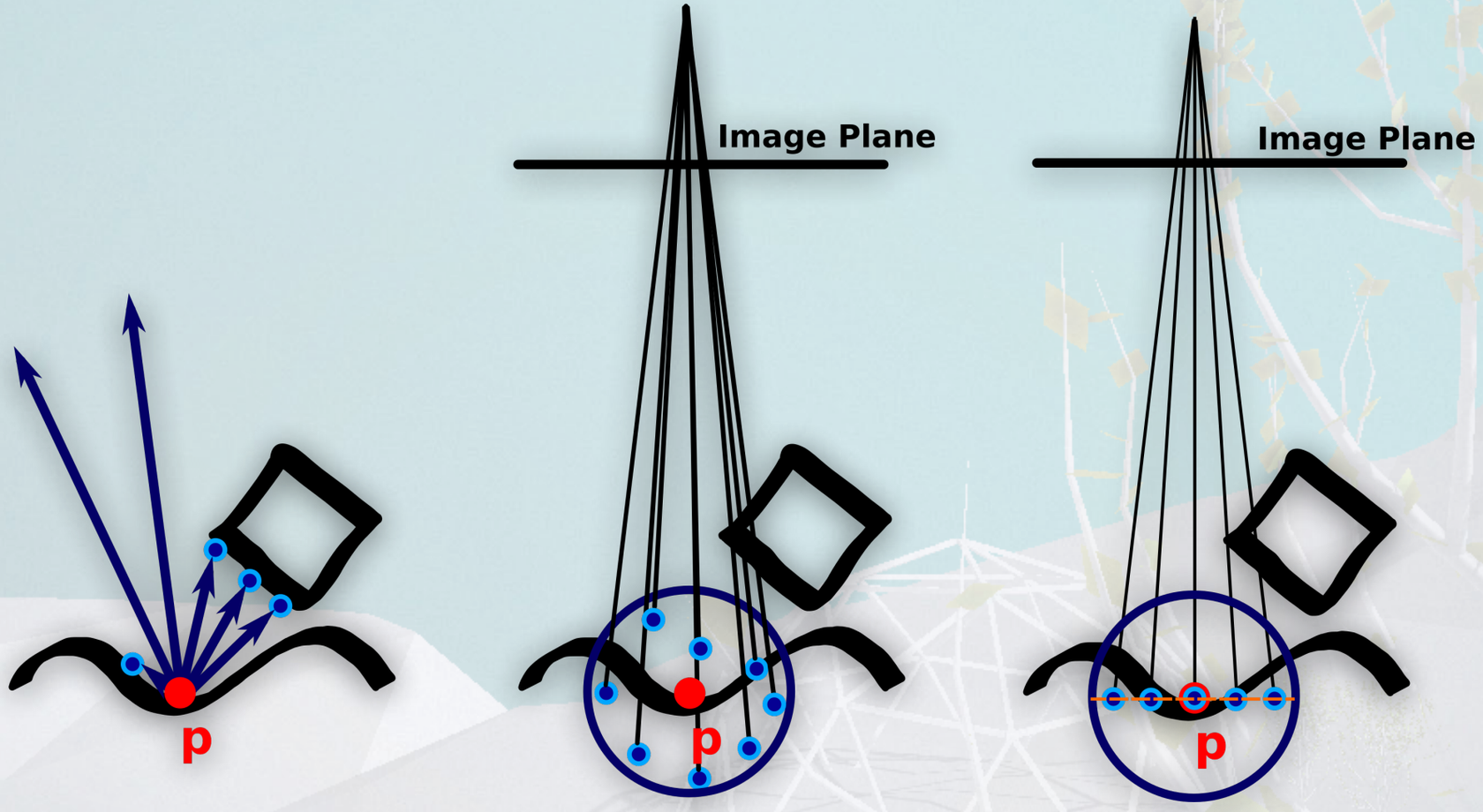
- Ambient Occlusion basiert auf diesem Prinzip

- Volumetric Obscurance bietet Alternative zu AO

3. I VO: AMBIENT OCCLUSION

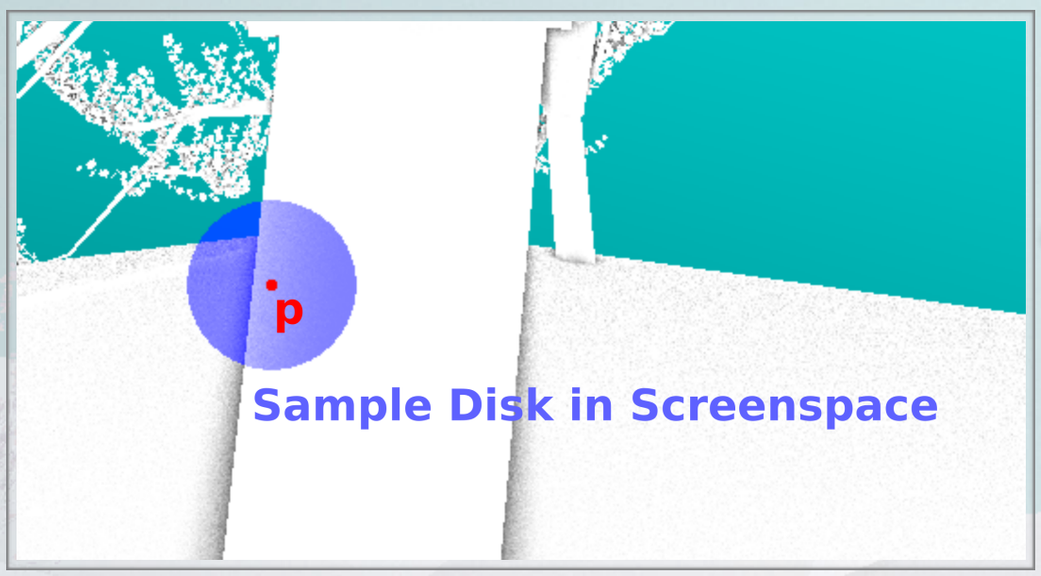
- Ambient Occlusion Ansätze:
  -Global Ambient Occlusion & Screenspace Ambient Occlusion

Ambient Occlusion | **Screenspace Ambient Occlusion** | **Volumetric Obscurance**



3. 2 VO: LINE SAMPLING

- betrachte für jeden vorgeredeten Texel den korr. 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** $i = 0 \rightarrow nSamples - 1$ **do**
  - radius = rand();
  - α = rand()·2π;
  - $v_i = \sqrt{1 - radius^2}$;
  - p = $(\cos(\alpha) \cdot radius, \sin(\alpha) \cdot radius)^T$;
  - sumSamples = sumSamples + v_i · SamplePoint(p, depth);
  - sumVolume = sumVolume + v_i;
- **end**
- return $\frac{sumSamples}{sumVolume}$;

end

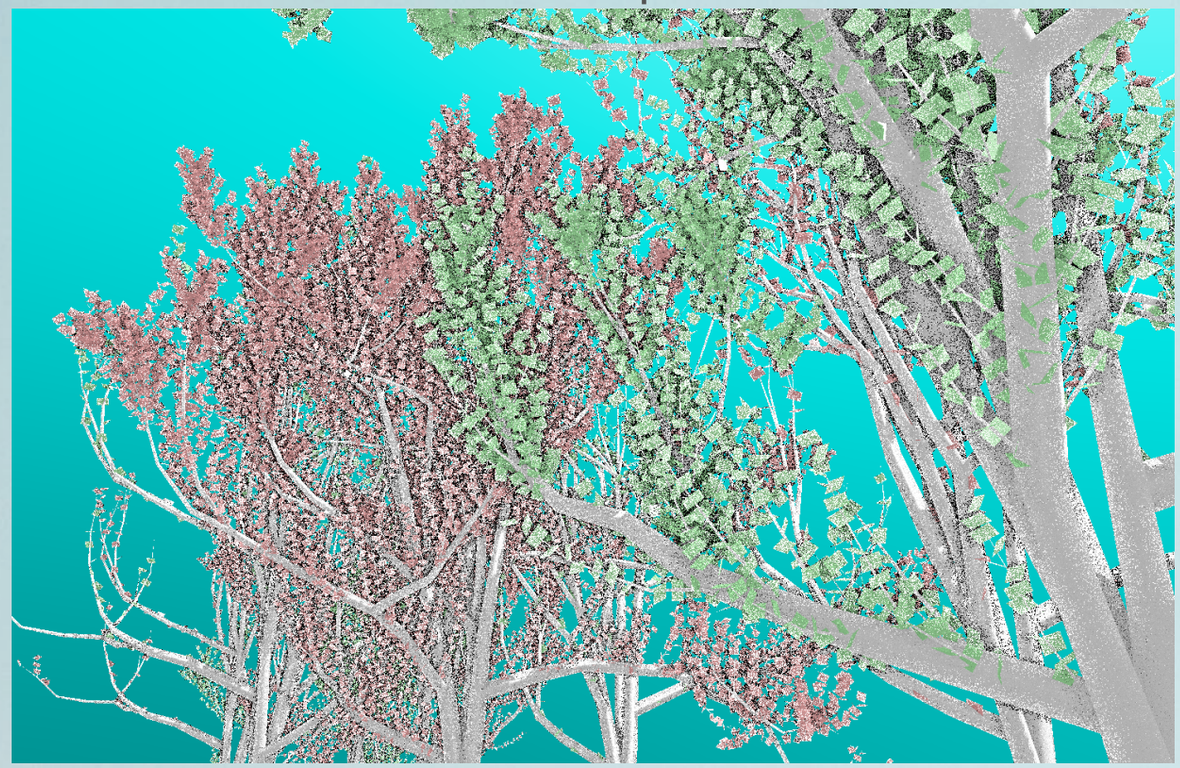
Function *SamplePoint(p, depth)*
- *hier wird ein einzelner Punkt gesampled, wie oben beschrieben*

end

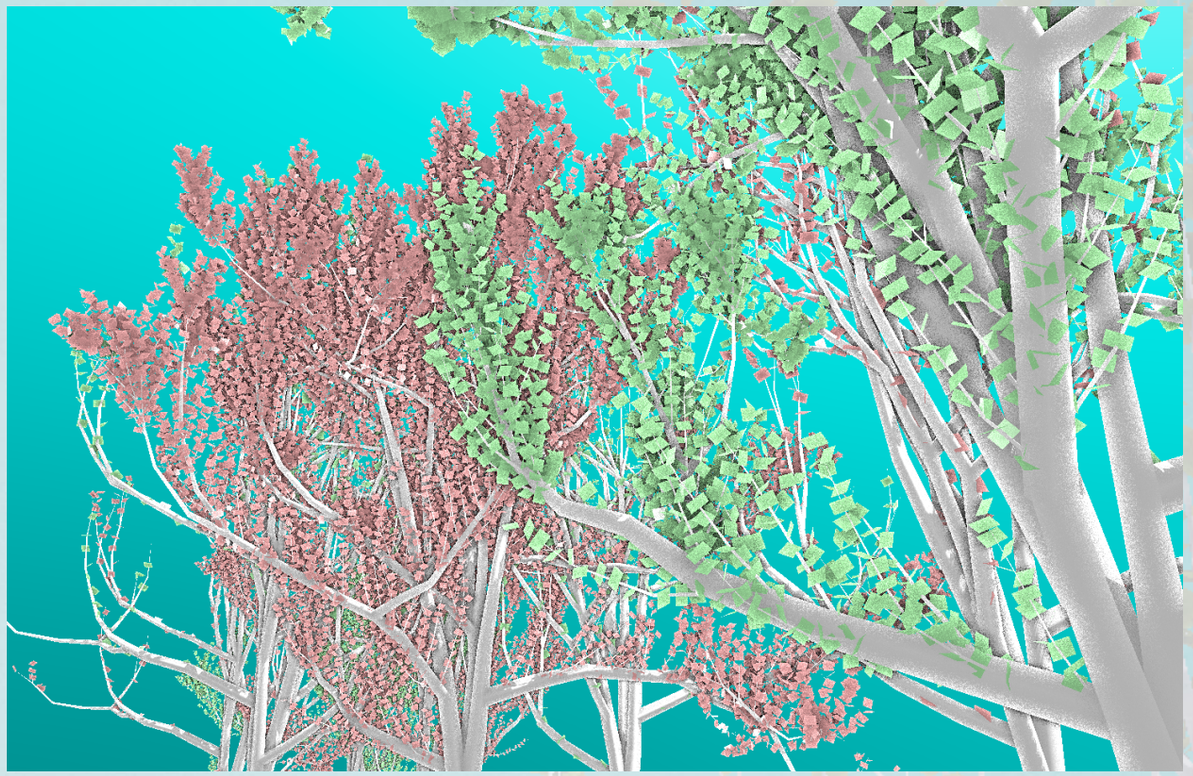
Function *rand()*
- return *zufälliger Wert im Intervall [0,1]*;

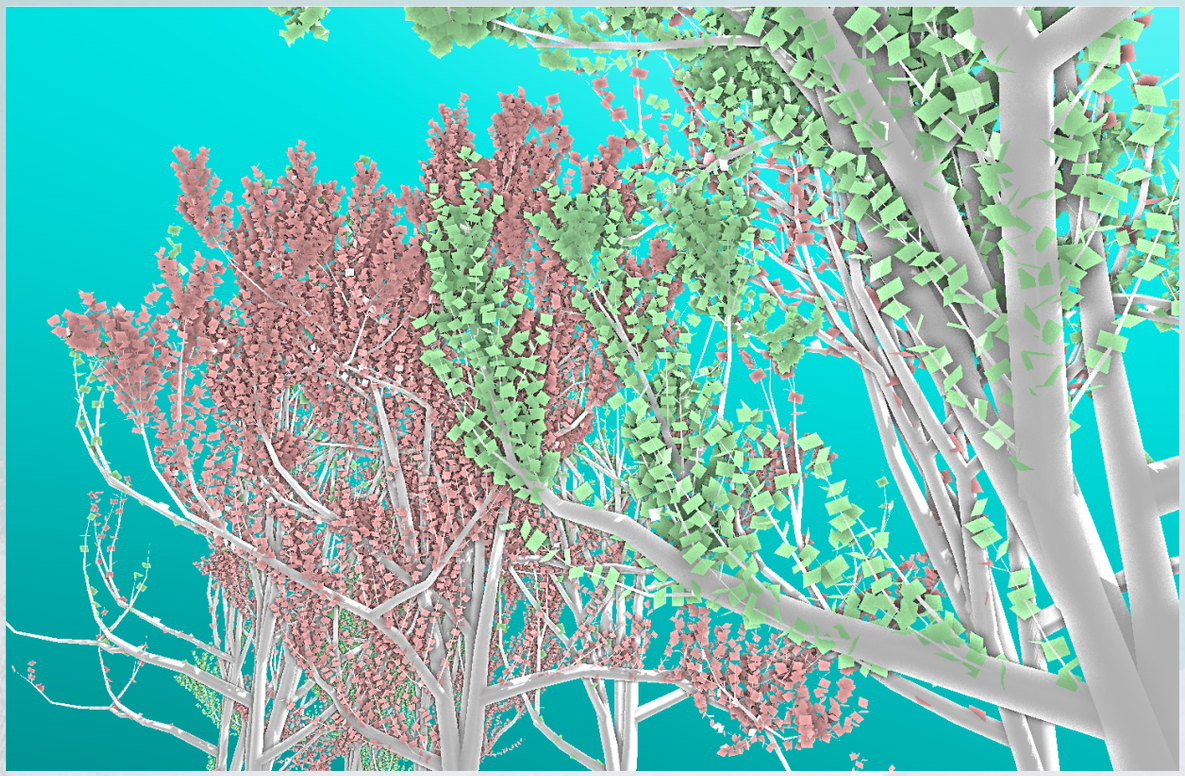
end

1 Samples

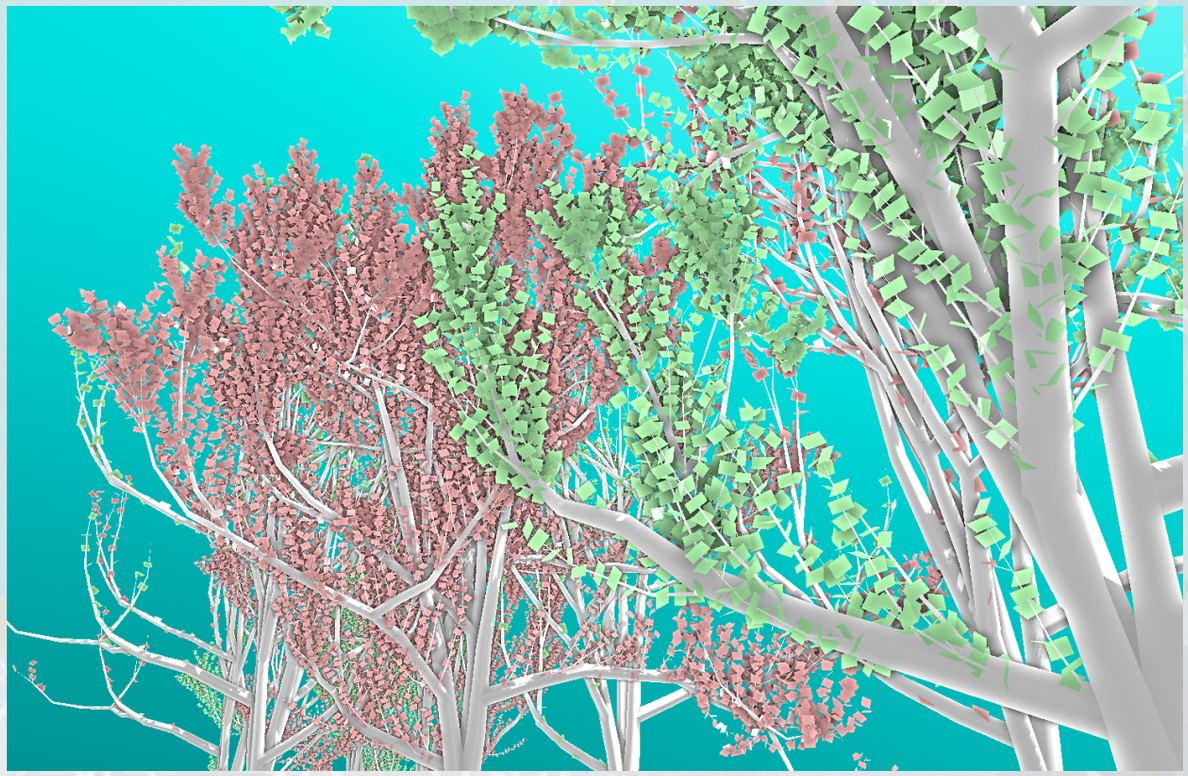


10 Samples





100 Samples



1000 Samples

3.4 VO: AREA SAMPLING

- Line Samping teuer, da viele Samples pro Pixel
- wünschenswert: stat. Beschreibung der Tiefenverteilung die Zeitgleich genaue Rückrechnung ermöglicht
- VSM bietet gleiches Prinzip und kann auf 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)

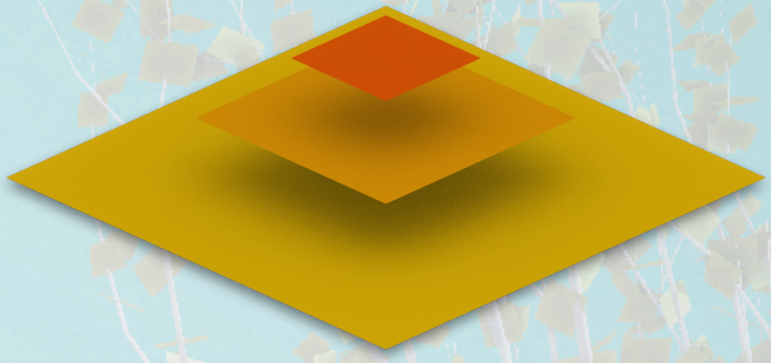
- Statistik zum Rückrechnen anwenden:

- $\mu = R$, $\sigma = G - \mu^2$

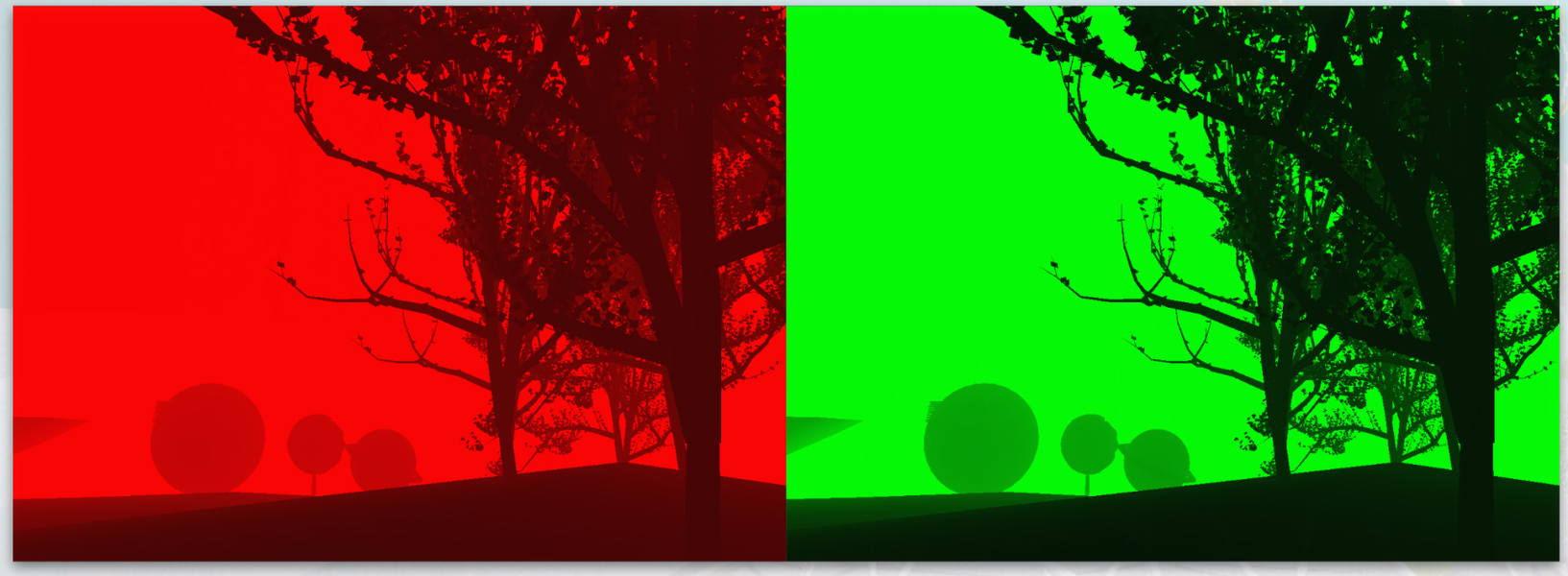
- $V_c(z_0, z_1, a, b) = a \cdot (z_1^2 - z_0^2)/2 + b \cdot (z_1 - z_0)$

- $a = \dfrac{-1}{2 \cdot \sigma}; \quad b = -a \cdot (\mu + \sigma)$

Depth (z)

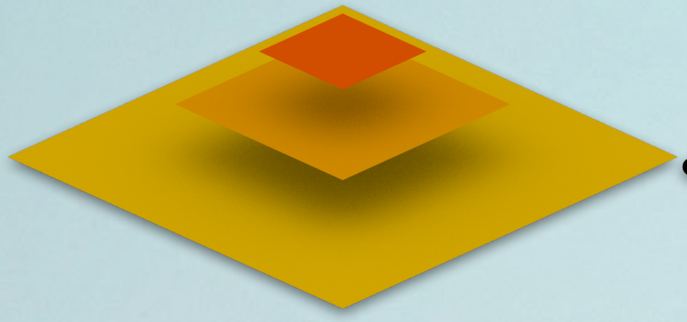


generate Mipmaps

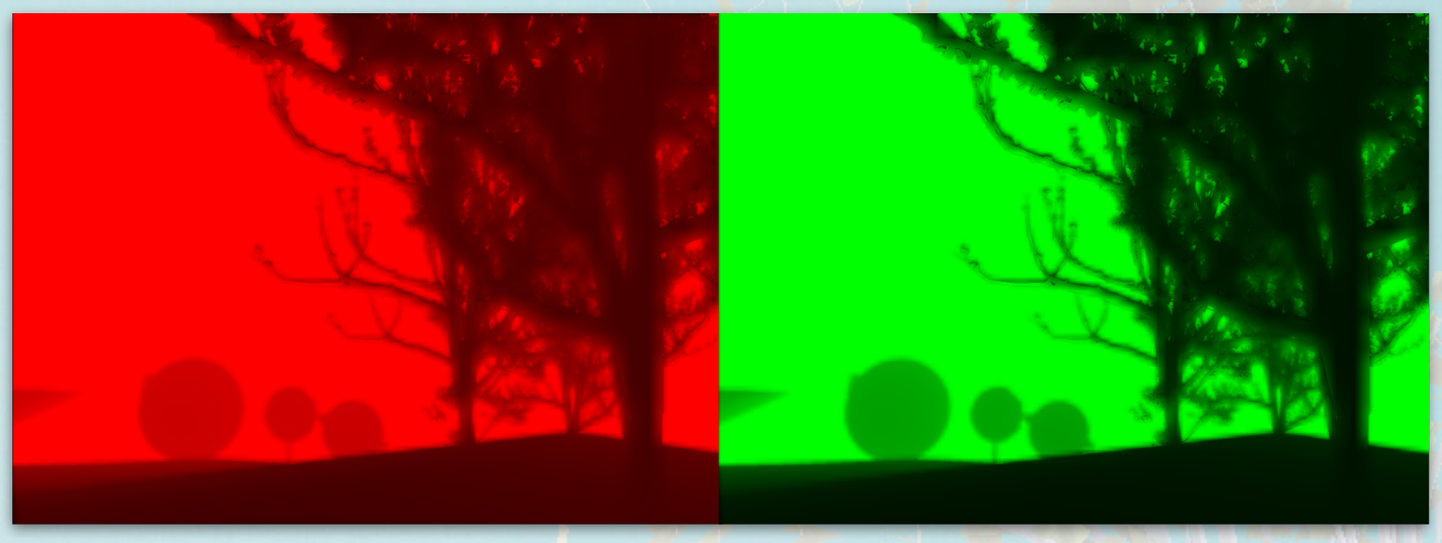


z^1

z^2



Mipmaps

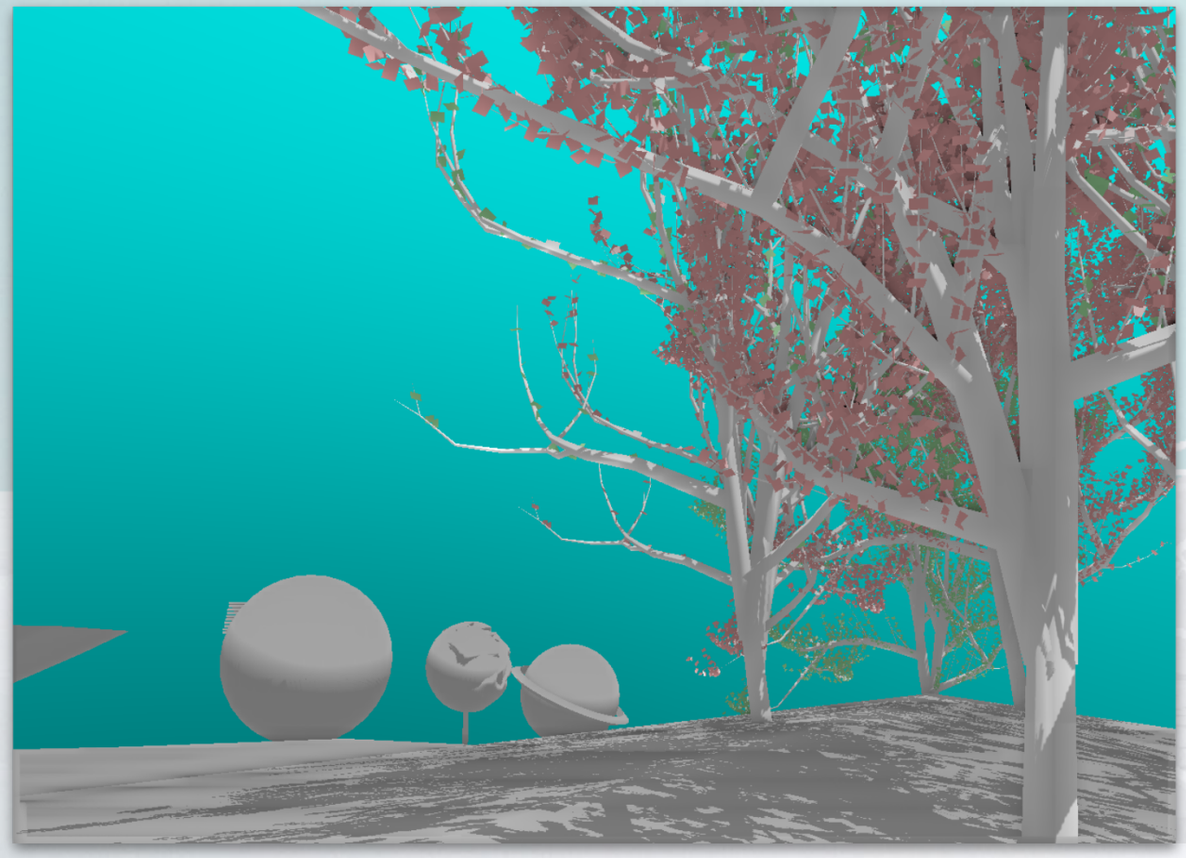


obscurance Term

mean

variance

obscurance Term



verrechnetes Bild

4. MOMENT SHADOW MAPS

- hohe Schattenqualität
- filterbare Shadow Map
- Momente aus Tiefe generieren (ähnlich VSM)
- MSAA und Gauss Pre-Processing
- Moment Based Volumetric Obscurance

4.1 MSM: ALGORITHMUS

- Hamburger Moment Shadow Mapping
- Ähnlich VSM
- untere Schranke , Schattenintensität wird nicht überschätzt
- H4MSM liefert Schattenwert durch 4 Momente z-z^4

4.2 MSM: MBVO ALGORITHMUS

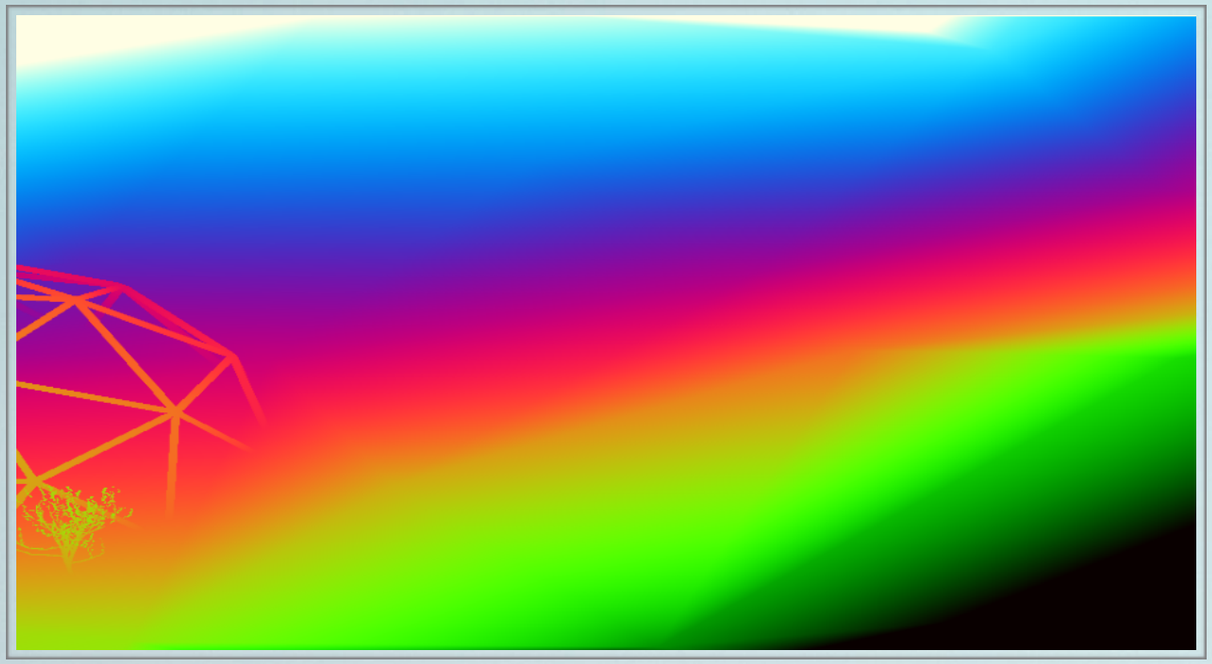
- Moment Based Volumetric Obscurance
- Nutzbar durch MSM Technik
- Algorithmus liefert Term für VO

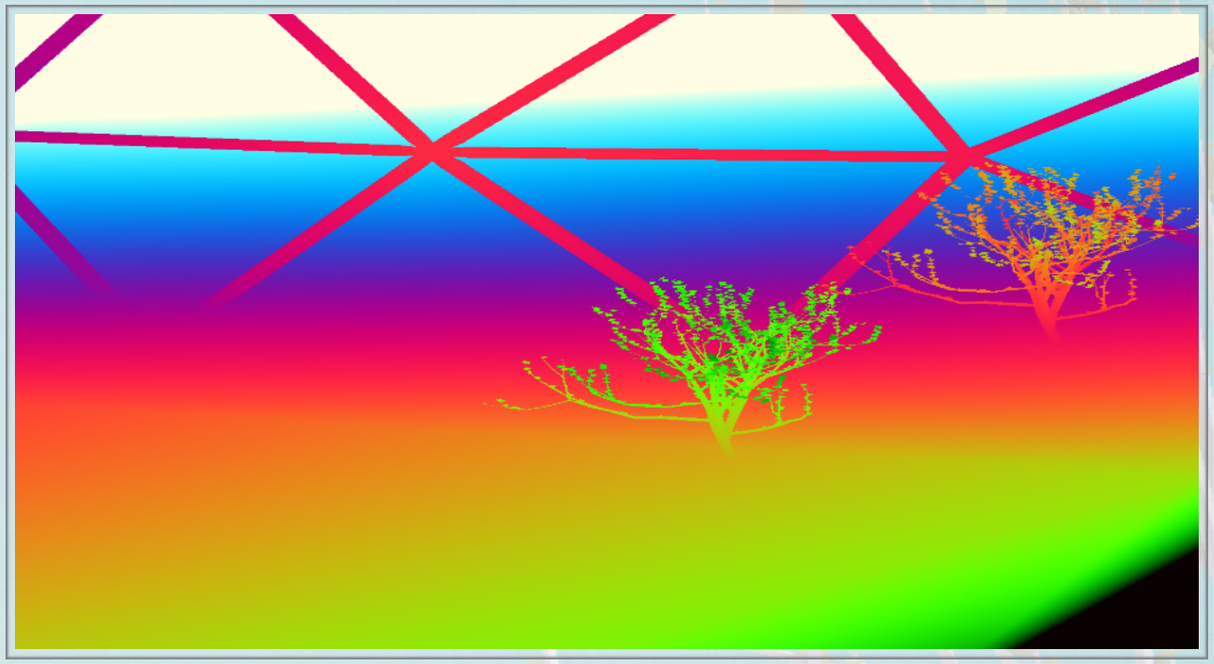
4.3 MSM: IMPLEMENTIERUNG

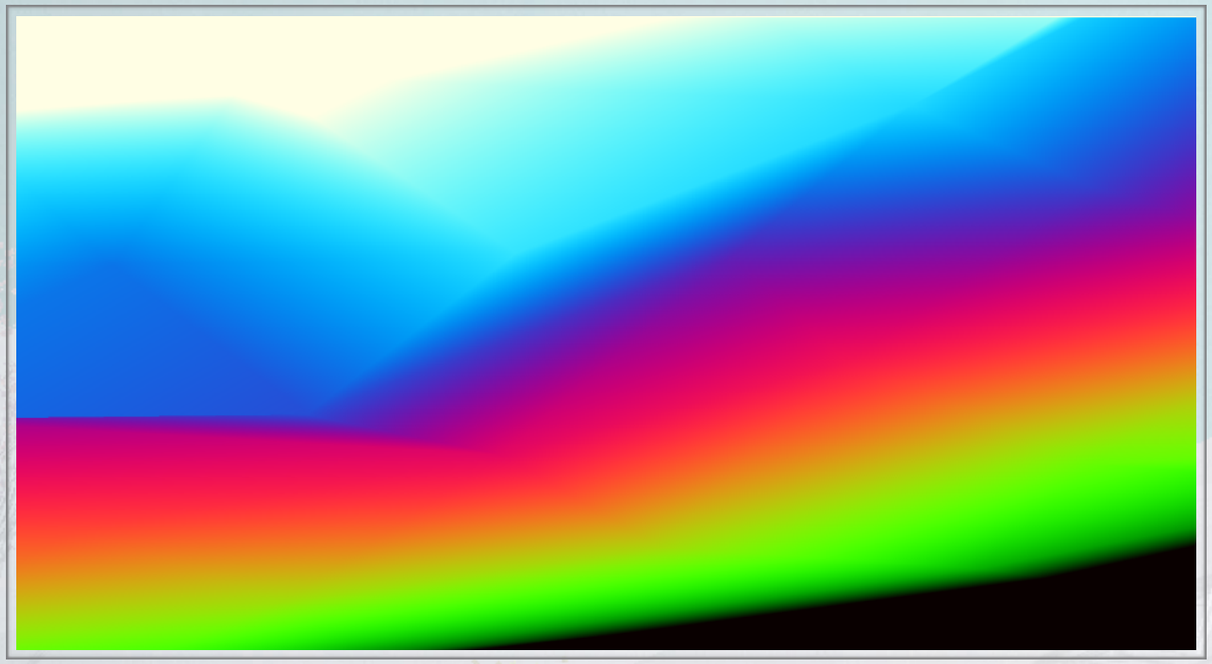
- Generieren der 4 Momente durch Tiefe z aus Tiefenbuffer
- speichern in RGBA Textur: z, z^2, z^3, z^4
- Matrixtransformation für 16-bit Quantifikation
- Rückrechnen zum speichern in Textur
- Berechnen der Schattenintensität

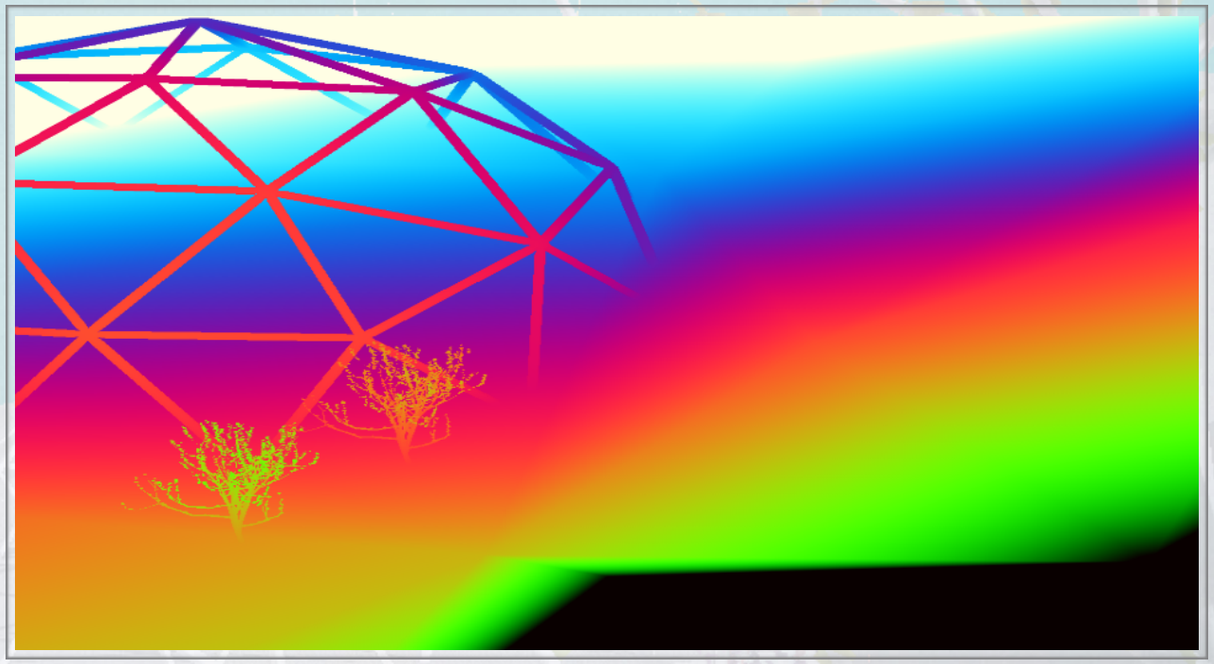
4.4 MSM: MOMENT TEXTUR

- Speichern in Textur



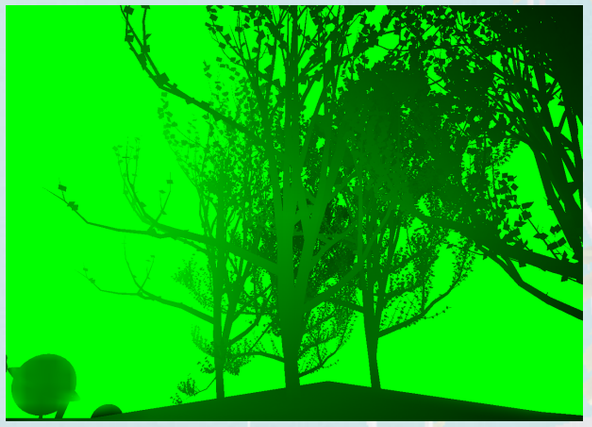


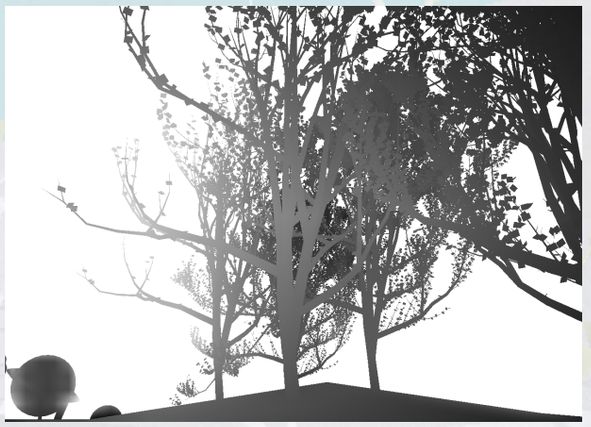




4.5 MBVO: IMPLEMENTIERUNG

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





4.6 MSM: ERGEBNISSE

1xMSAA

2xMSAA

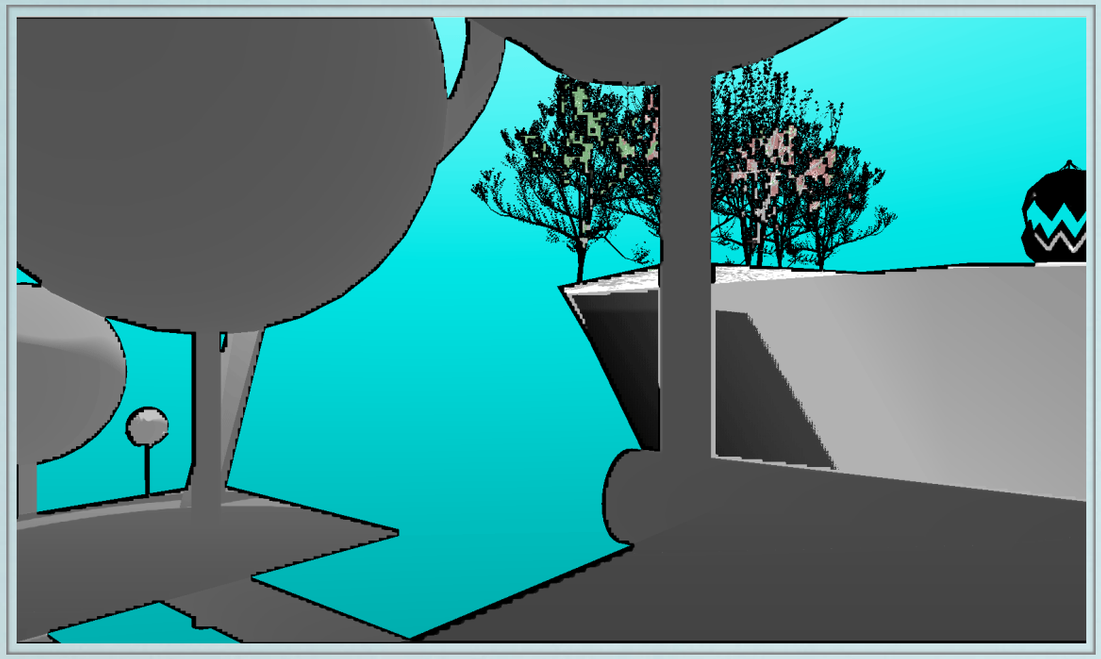
16xMSAA

GAUSS

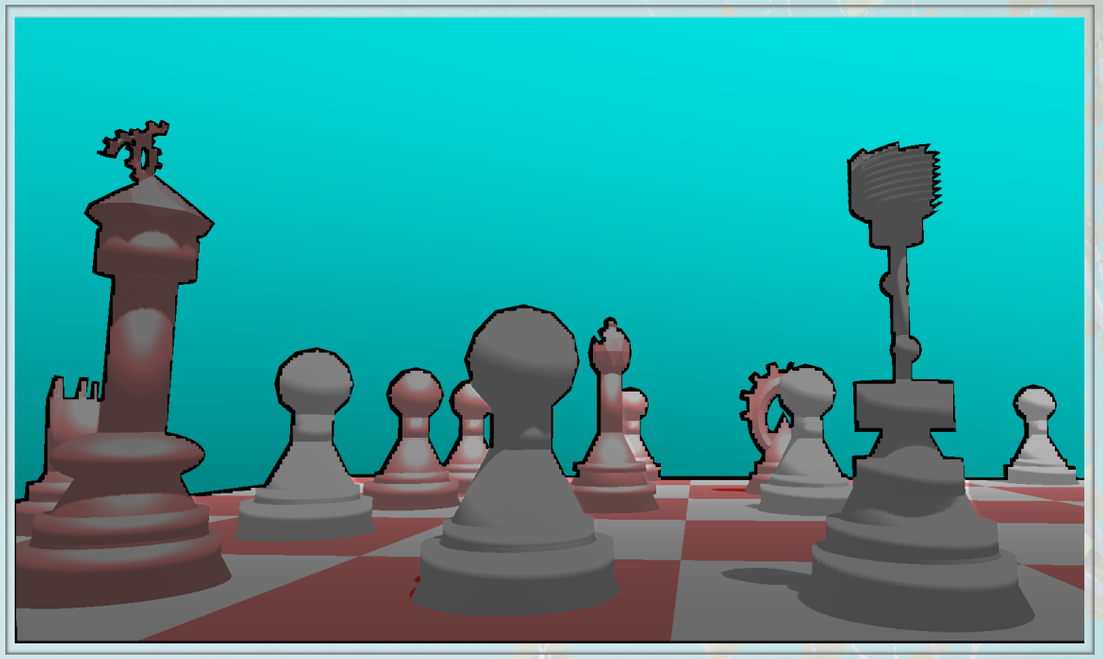
4.6 MSM: ERGEBNISSE

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

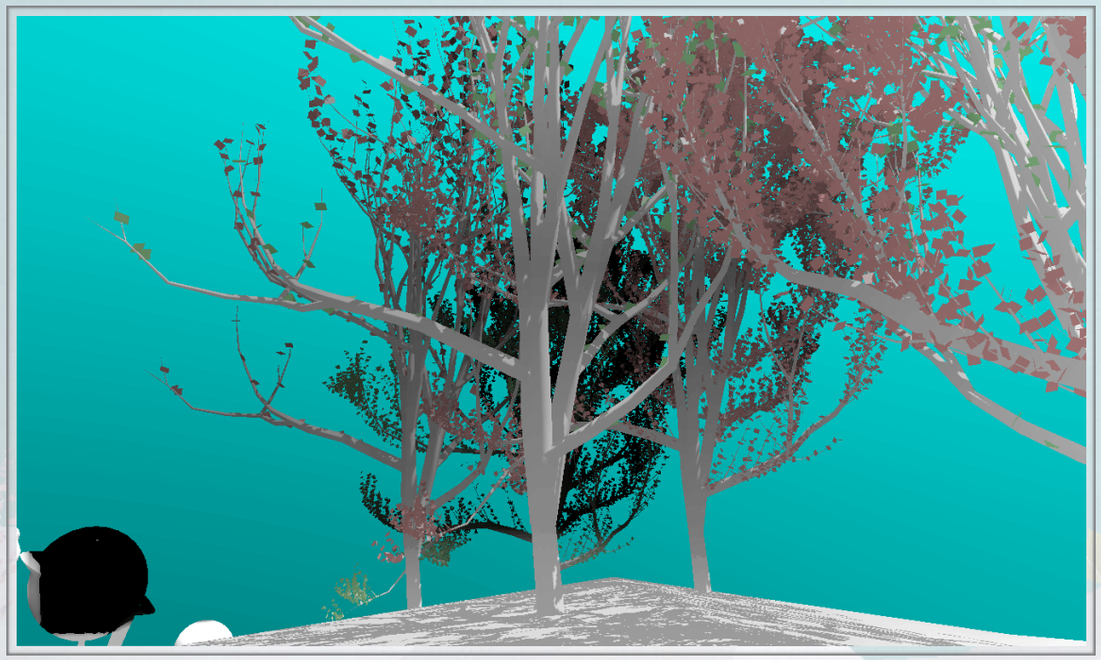
4.7 MBVO: ERGEBNISSE



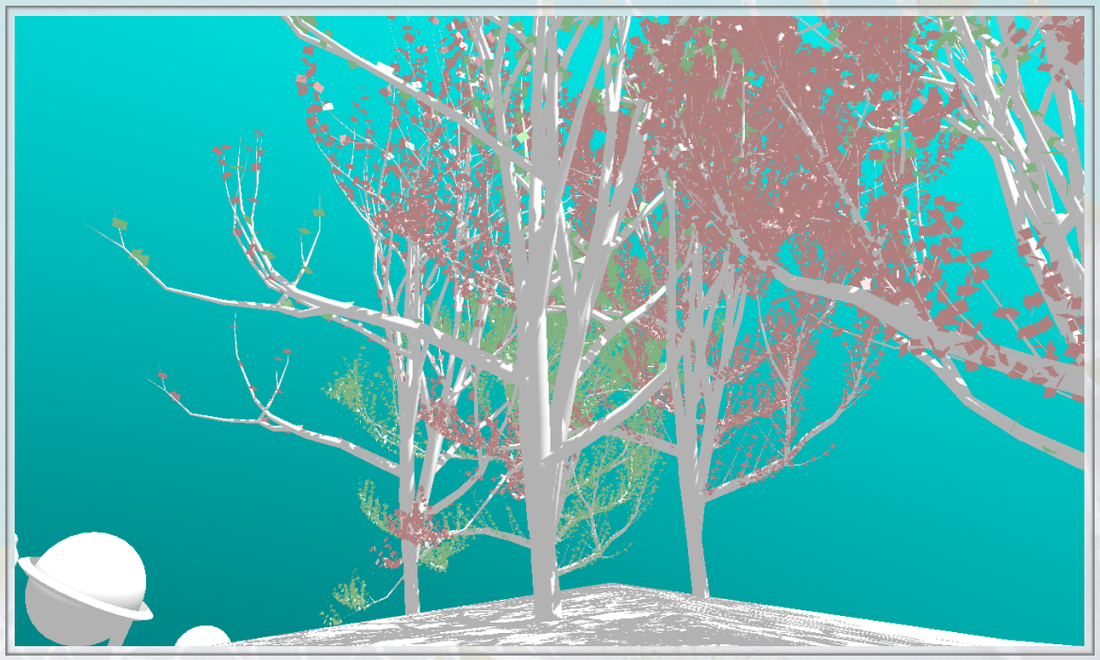
MBVO 1



MBVO 2



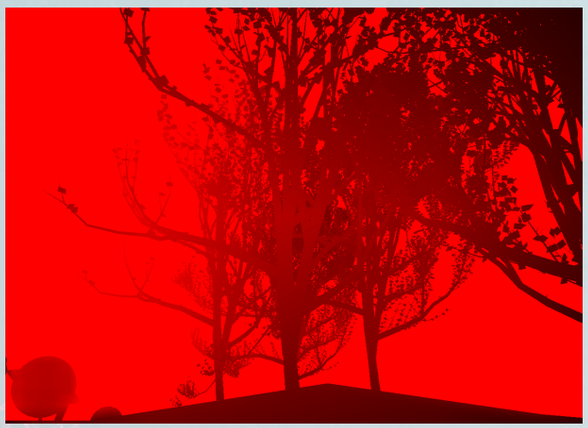
MBVO ON

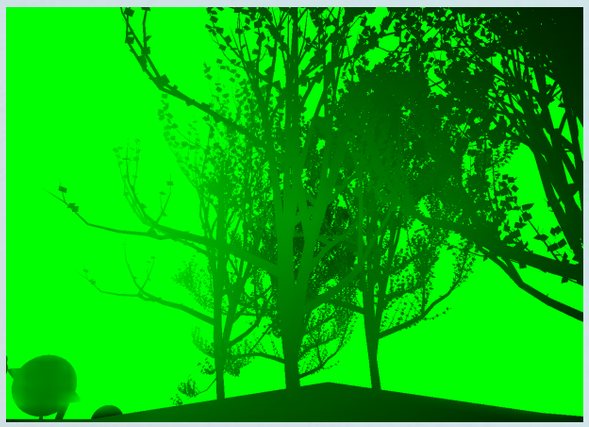


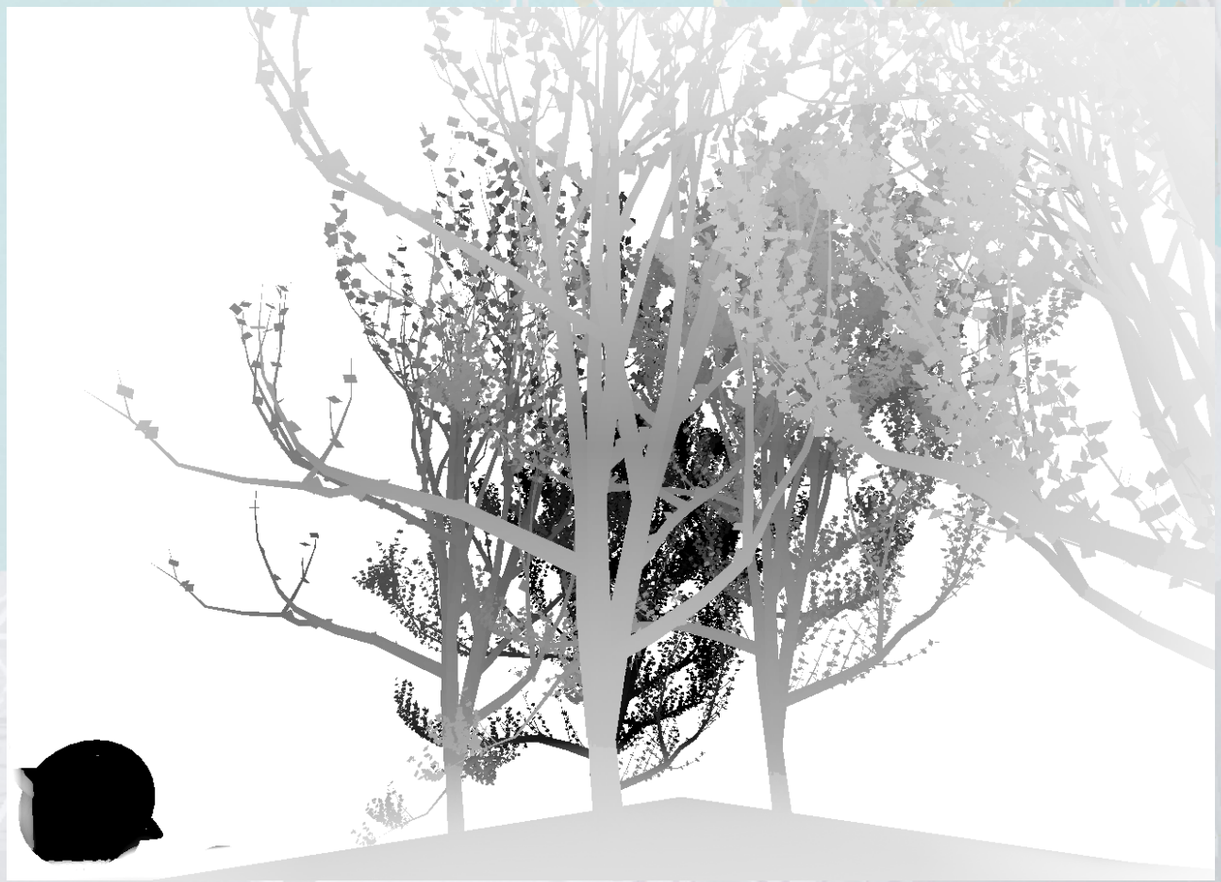
MBVO OFF

4.7 MBVO: ERGEBNISSE

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

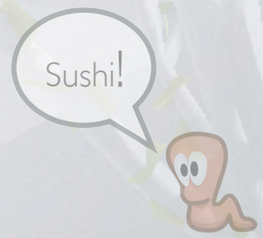






5. SUSHI! WITH R.I.C.E.

- **S**imple **U**nified **S**hader **H**elper **I**nterface
- C++, Qt, OpenGL
- Linux und Windows Support
- Sushi! Game Demo



5. SUSHI!

PRESS START

7. FAZIT UND AUSSICHTEN

Erfolg kommt nur durch Arbeit