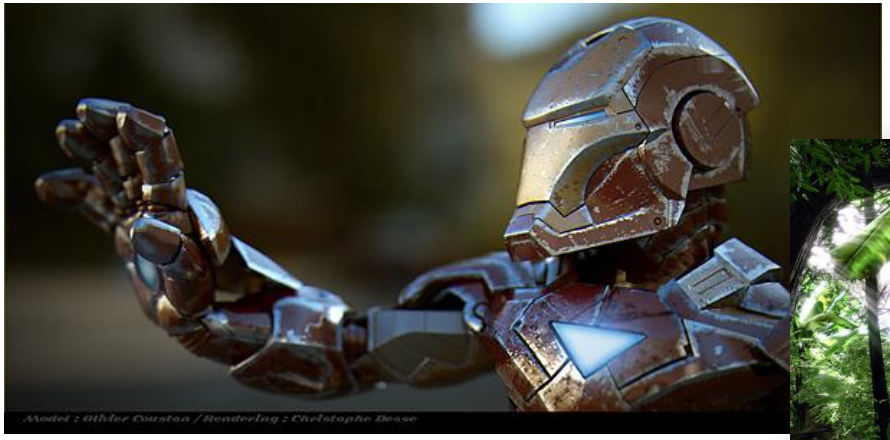
VII. Rendering

VII. Rendering

- Was ist ein Rendering?
  - Das Zusammenspiel von Modellen, Material und Licht wird erst in einem Rendering sichtbar. Man spricht hierbei von der ***Rasterung*** oder ***Bildsynthese***. Dreidimensionale Vektor-Daten werden auf ein Zweidimensionales Raster (Bitmap) projiziert. Hierbei wird für jeden Pixel einer Bitmap ein Farbwert (meist im RGB-Farbraum) ermittelt.
  - Man spricht beim Rendern auch von der ***Umwandlung von Rohdaten in Mediendaten***.
  - Eine Computeranimation besteht immer aus einer Einzelbildfolge von Renderings.

VII. Rendering

- Echtzeit oder Nicht Echtzeit?
  - Schematisches Rendern (z.B. Drahtgitter)
  - Konturenverstärktes Rendern (z.B. Cartoon)
    - Vektor-Renderings (z.B. als Export für Flash)
  - Realistisches Rendern (z.B. für Architektur, Filme, …)



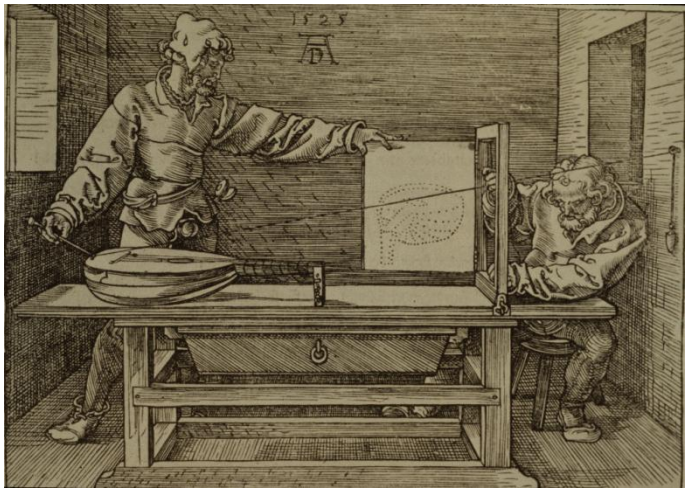
VII. Rendering

- Wie kommt ein Rendering zustande?
  - Kamera/Ansicht
  - 3D Objekte/Shader
  - Beleuchtung

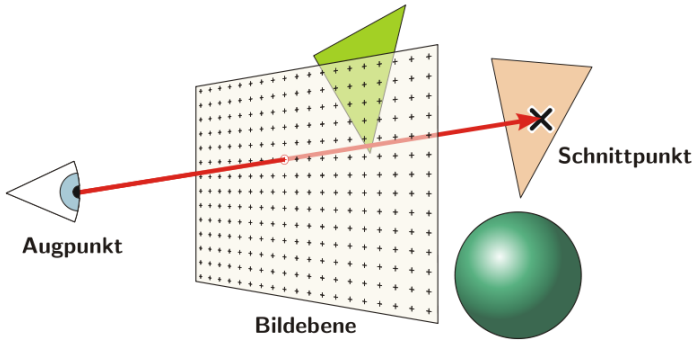
- Echtzeit
  - Eine Grafikkarte berechnet ein Bild (GPU Rendering)
  - Mehrere Grafikkarten berechnen ein Bild (Multi GPU Rendering)

- Nicht Echtzeit
  - Ein Computer berechnet ein Bild (Still)
  - Ein Computer berechnet mehrere Bilder (Animation)
  - Mehrere Computer berechnen mehrere Bilder (Netzwerkrendering)
  - Mehrere Computer berechnen ein Bild (Distributed Rendering)

VII. Rendering



- Was macht ein Renderer?
  - Projektive Abbildungen
    - Perspektivisch/Orthogonal
    - physikalische Projektionen
  - Sichtbarkeitsberechnungen
    - Culling (Frustum Culling, Backface Culling, …)
  - Verdeckungberechungen
    - Raytracing
  - Shading (Berechnung von Materialeigenschaften)
  - Licht- und Schattenberechnung
    - Direkte/Indirekte Beleuchtung
    - Globale Beleuchtung
  - Filterung
    - Kantengättung (Antialias)
    - Texturfilterung (Super Sampling, Anisotrope Filterung)



VII. Rendering

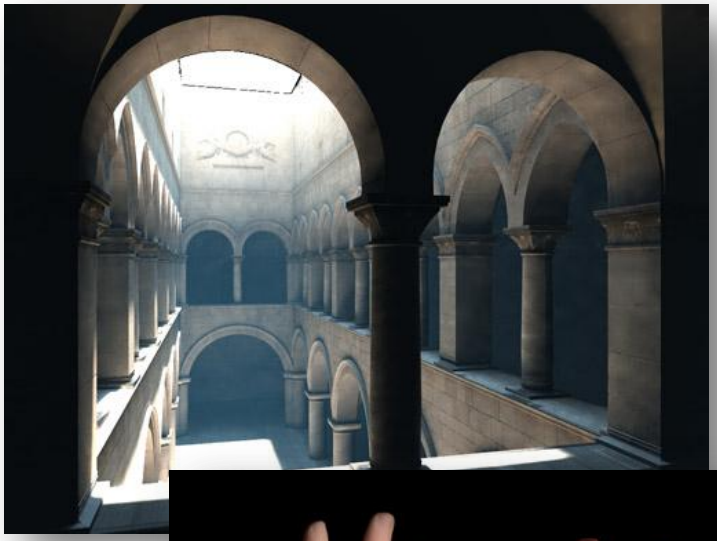
- Ein simples Raytracing-Beispiel

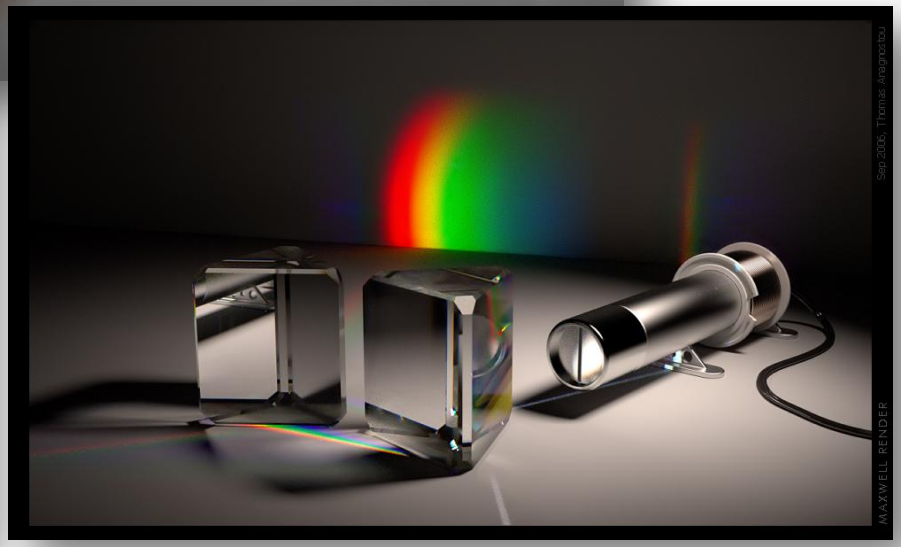
```
Prozedur Bild_Rendern
    Strahl.Ursprung = Augpunkt
    Für jedes (x,y)-Pixel der Rastergrafik
        Strahl.Richtung = [3D-Koordinaten des Pixels der Bildebene] – Augpunkt
        Farbe des (x,y)-Pixels = Farbe_aus_Richtung(Strahl)

Funktion Farbe_aus_Richtung(Strahl)
    Schnittpunkt = Nächster_Schnittpunkt(Strahl)
    Wenn Schnittpunkt.Gewinner ≠ (kein) dann
        Farbe_aus_Richtung = Farbe_am_Schnittpunkt(Strahl, Schnittpunkt)

Funktion Nächster_Schnittpunkt(Strahl)
    MaxDistanz = ∞
    Schnittpunkt.Gewinner = (kein)
    Für jedes Primitiv der Szene
        Schnittpunkt = Teste_Primitiv(Primitiv, Strahl)
        Wenn Schnittpunkt.Distanz < MaxDistanz dann
            MaxDistanz = Schnittpunkt.Distanz
            Schnittpunkt.Gewinner = Primitiv
    Nächster_Schnittpunkt = Schnittpunkt
```

VII. Rendering

- Welche Techniken beherrschen moderne Renderer?
  - Global Illumination
    - Path Tracing, Bidirektionales Path Tracing, Light Transport & Light Tracing
    - Photon Mapping (Speicherung der Global Illumination in einer Datei)
  - Volumenstreuung (Sub Surface Scattering)
  - Tiefenunschärfe
  - Bewegungsunschärfe
  - Kaustiken
  - Spektrales Rendering
    - Polarisationseffekte (Berücksichtigung von Lichtwellenvektoren)
    - Dispersion (Berücksichtigung von Lichtwellen bei der Lichtbrechung, z.B. im Prisma)
    - Metamerie (optische Verschiebung von Farbräumen durch Umgebungslichter)
  - High Dynamic Range (Farbwerte als Gleitkommazahlen für höchsten Kontrastumfang)
  - Relativistisches Raytracing
    - Farb- und Formverzerrungen durch gekrümmte Raumzeit

VII. Rendering





VII. Rendering

- Gammakorrektur
  - Die Helligkeit vieler Bildschirme ist nicht abhängig von der Bildhelligkeit, sondern von der Stärke der verarbeitenden Chips, Signalwandler, Röhren, LEDs, Backlights, etc.
    - Jedes Gerät besitzt seine eigene Funktion zur Korrektur seiner Helligkeit
  - Eine Potenzfunktion mit einem Exponenten ist die sogenannte ***Gammakorrektur*** mit dem sogenannten ***Gammawert***
    - Spezielle Helligkeitsprofile werden mithilfe von ***LUT-Tabellen*** vorgenommen
  - Typische Gammawerte:
    - RGB-Monitor unter Windows: 2,2
    - Mac Standardprofil vor Mac OS X 10.6 liegt bei: 1,8

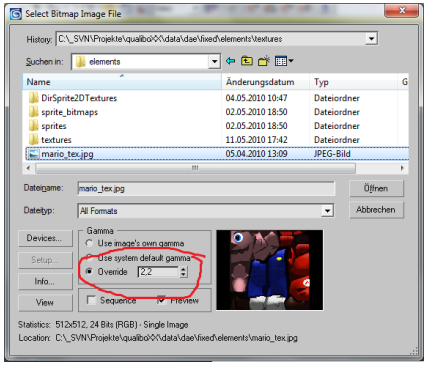
VII. Rendering

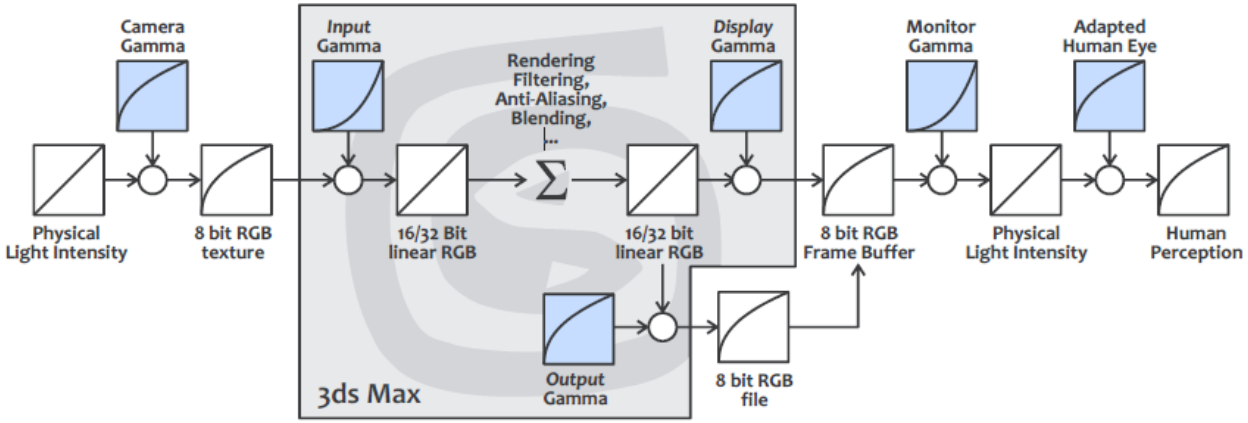
- Linear Workflow
  - Notwendig zum Rendern und Bearbeiten unter der Berücksichtigung der natürlichen Farbgebung
    - Unabhängig von der Helligkeit des Wiedergabegerätes, an dem gerade gearbeitet wird
    - Kein Informationsverlust bei nachträglicher Helligkeits- oder Kontrastveränderungen
  - Aufnahmen und Renderings sollten unabhängig von der Helligkeit des Bildschirms bearbeitet werden. Man spricht dabei vom ***Linearen Workflow.***
    - Der Gammawert wird in das Farbprofil eines Bildes integriert (ICC-Profil im ***sRGB***)
  - Beispiel: Hat ein Bild den Gammawert 0,454 (1/2,2), wird es an einem Monitor mit einer Gamma von 2,2 für unsere Augen richtig dargestellt.
  - Welche Auswirkungen hat der Lineare Workflow auf die Arbeit in der Computergrafik?
    - Farben & Texturen?
    - Bearbeitung?
    - Wiedergabe?

VII. Rendering

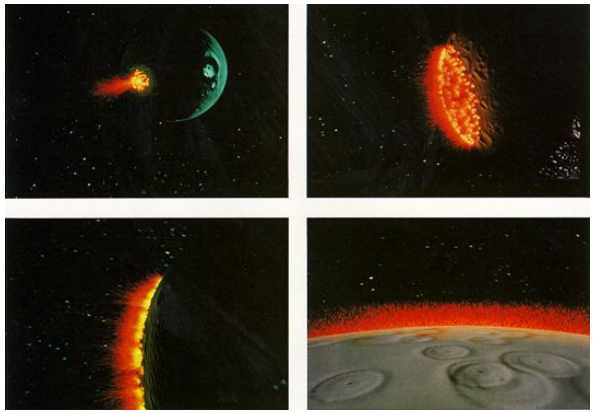
- Linear Workflow
  - Input-Dateien (Texturen und Farben) müssen mit dem Gammawert 2,2 skaliert werden
  - Das Rendering erfolgt linear, d.h. die Gammakorrekturist separat (in diesen Fällen ist das Rendering immer etwas dunkel)
  - Das Datenformat muss Gamma-Korrektur unterstützen (OpenEXR, Adobe sRGB,…)

  → Nicht notwendig mit der Arbeit im 32-Bit-Float-Raum (HDR)





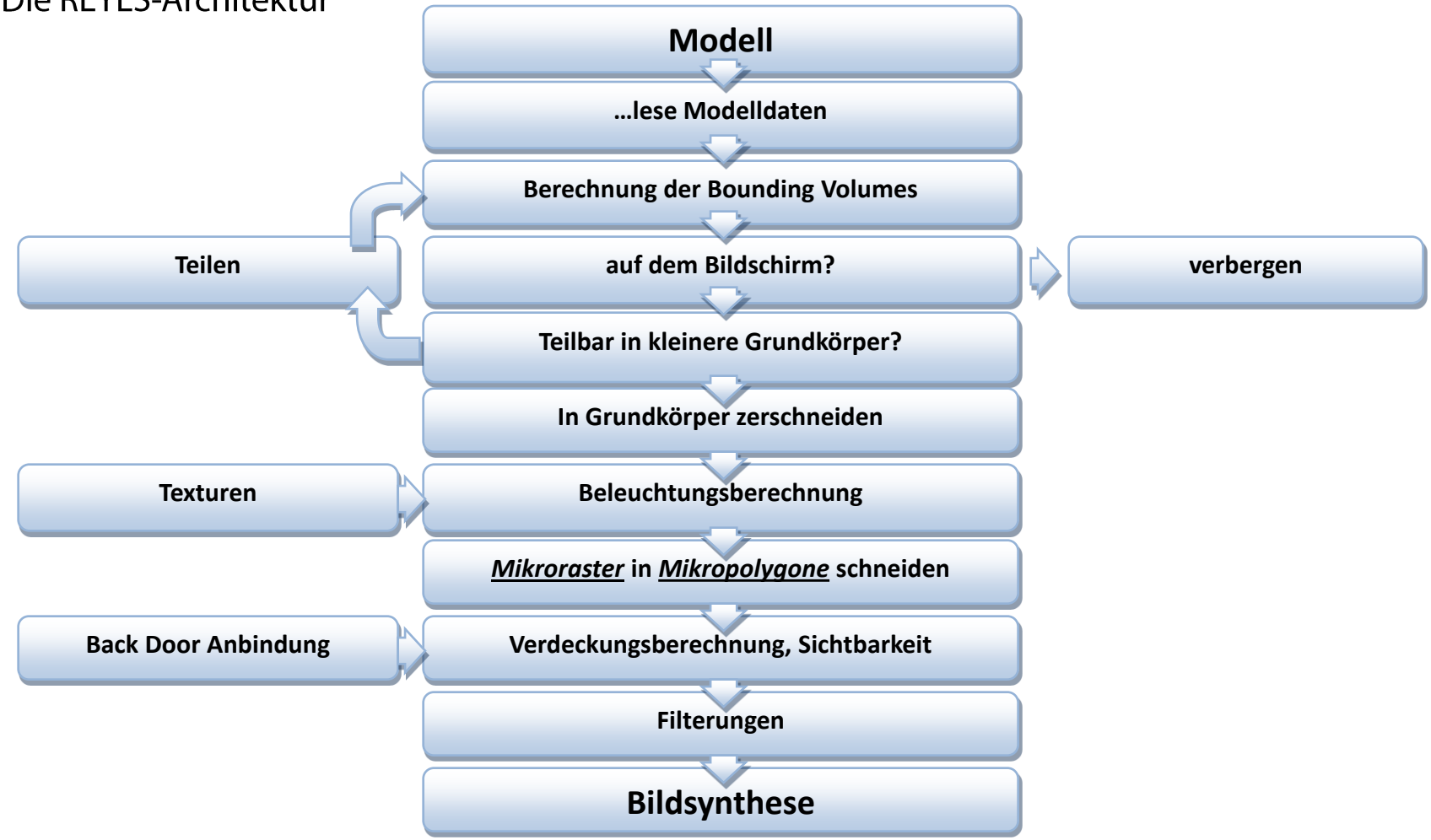
VII. Rendering



- Die REYES-Architektur
  - “Renders Everything You Ever Saw “ ist eine Rendermethodik, die von Lucasfilm LTD, ILM und Pixar entwickelt wurde um große CG-Produktion rendern zu können.
  - Rendering-Architektur unter Berücksichtigung
    - Großer Modellkomplexität
    - Ausfallssicherheit
    - Vollständige Shadervielfalt
    - Minimales Raytracing (Approximation von GI mit Mapping-Technologien)
      - Kein Schwerpunkt auf Radiosity
    - Höchste Rendergeschwindigkeit!
    - Bildqualität
      - Vermeidung von Artefakten (Gezackte Ecken, Schatten-Flackern)
      - Vermeidung von Moirée-Effekten
  - Erweiterbar/konfigurierbar auf zukünftige Technologien ohne Reprogrammierung
  - Wurde zur Renderman-Implementation (Renderman, 3delight, Arnold)

VII. Rendering

- Die REYES-Architektur

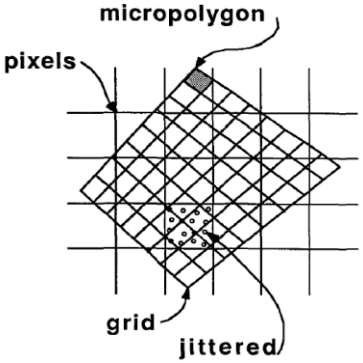


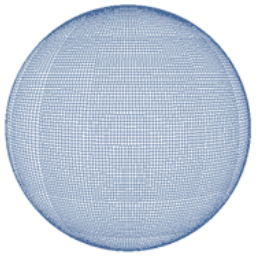
VII. Rendering

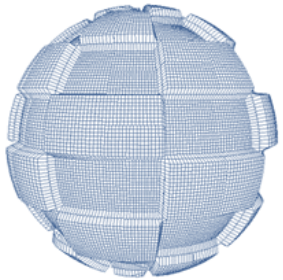
```
Initialize the z buffer.
For each geometric primitive in the model,
    Read the primitive from the model file
    If the primitive can be bounded,
        Bound the primitive in eye space.
        If the primitive is completely outside of the hither-yon z range, cull it.
        If the primitive spans the ε plane and can be split,
            Mark the primitive undiceable.
        Else
            Convert the bounds to screen space.
            If the bounds are completely outside the viewing frustum, cull the primitive.
    If the primitive can be diced,
        Dice the primitive into a grid of micropolygons.
        Compute normals and tangent vectors for the micropolygons in the grid.
        Shade the micropolygons in the grid.
        Break the grid into micropolygons.
        For each micropolygon,
            Bound the micropolygon in eye space.
            If the micropolygon is outside the hither-yon range, cull it.
            Convert the micropolygon to screen space.
            Bound the micropolygon in screen space.
            For each sample point inside the screen space bound,
                If the sample point is inside the micropolygon,
                    Calculate the z of the micropolygon at the sample point by interpolation.
                    If the z at the sample point is less than the z in the buffer,
                        Replace the sample in the buffer with this sample.
    Else
        Split the primitive into other geometric primitives.
        Put the new primitives at the head of the unread portion of the model file.
Filter the visible sample hits to produce pixels.
Output the pixels.
```

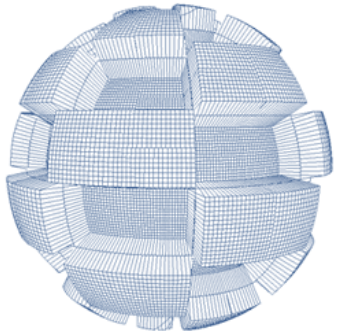
VII. Rendering

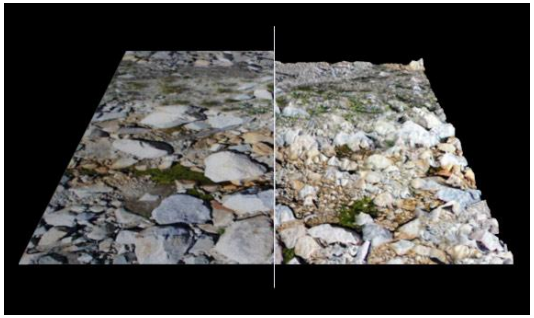
- Micro-Displacement

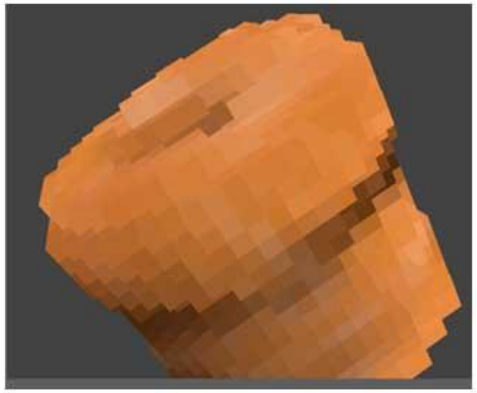






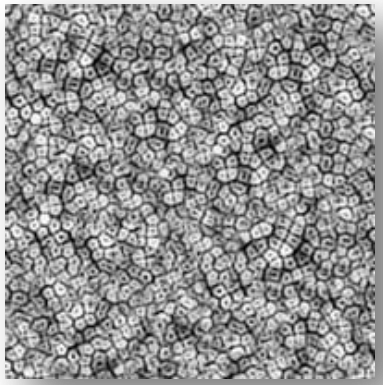






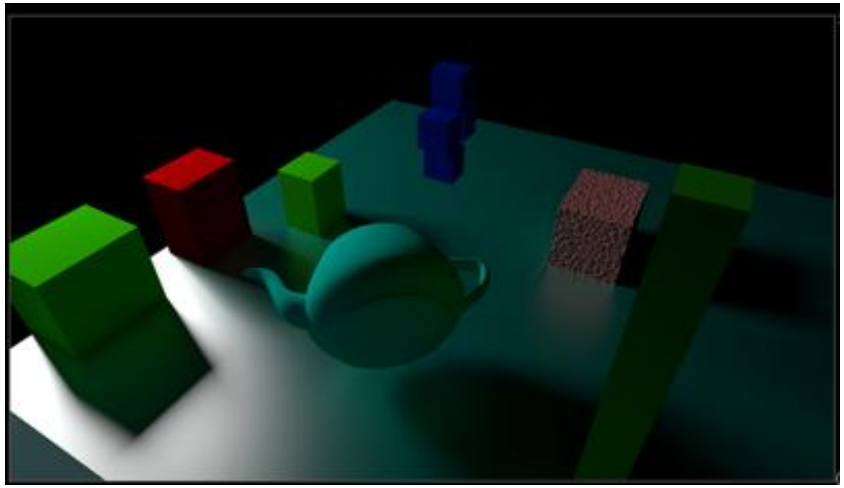
VII. Rendering

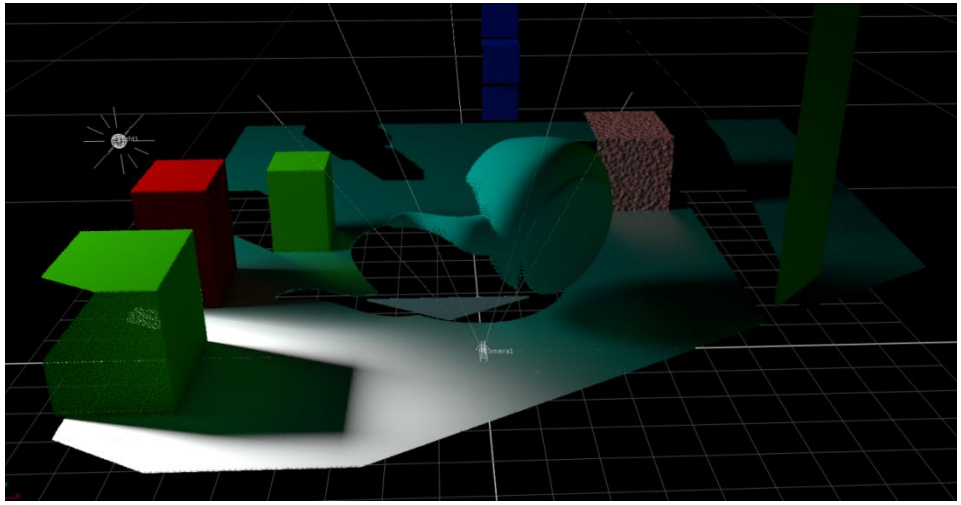
- Microshader

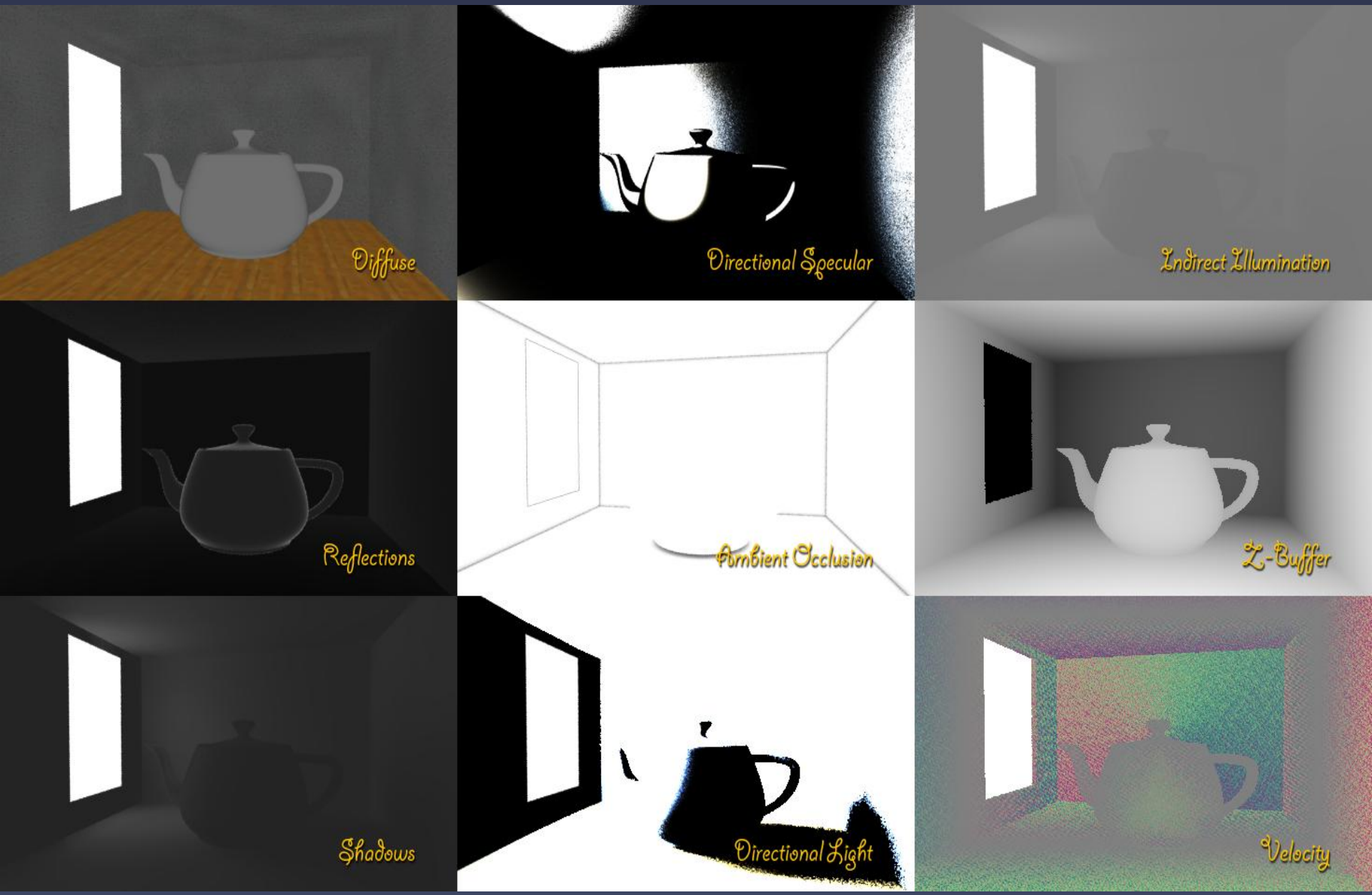


VII. Rendering

- Aus was besteht ein Rendering?
  - Diffuse (Farbwerte)
  - Specular (Glanzwerte)
  - Normals (Normalen)
  - Z-Buffer (Tiefenwerte)
  - Position (räumliche Pixel)
  - Ambient Occlusion (Schatten)
  - Velocity (Geschwindigkeit)
  - Reflections (Reflexionen)
  - Direct & Indirect Illumination
  - Transparency (Durchsichtigkeit)
  - Translucency (Lichtdurchlässigkeit)
  - uvm...






Diffuse
Directional Specular
Indirect Illumination
Reflections
Ambient Occlusion
Z-Buffer
Shadows
Directional Light
Velocity

VII. Rendering

- Warum werden die Elemente eines Rendering separate Dateien gespeichert?
  - Nachträgliche Kontrolle/Korrektur über viele Parameter
    - Ohne neues Rendering (Zeitersparnis)
    - Mehr Möglichkeiten bei der Nachbearbeitung, als mit fertigem Rendering
  - Sehr wichtig bei:
    - Bildverarbeitung
    - Compositing/Nachbearbeitung
    - Farbkorrektur (Grading)
    - für die Verwendung spezieller visueller Effekte

- Was kann alles in Dateien gespeichert werden?
  - Sämtliche sichtbaren Oberflächeninformationen einer Szene können in separate Einheiten abgelegt und später wieder zusammen gefügt werden.
    - Ausnahme:
      - Refraktion

VII. Rendering

- Wie können Layers wieder zusammengefügt werden?

  → Im Composting! Siehe nächstes Kapitel