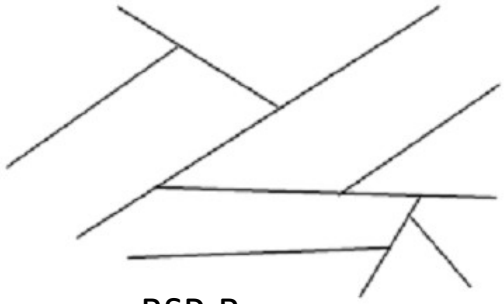
OPTIMIERUNGSSTRUKTUREN

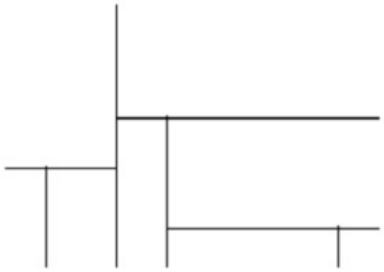
Prof. Dr. Elke Hergenröther

Optimierung: BSP- Baum und kd-Baum

- Verfahren zur Verwaltung räumlicher Daten
- Binäre hierarchische Unterteilung
  - *Szene wird rekursiv in zwei Teilräume geteilt*
  - *Teilung erfolgt durch eine orientierte Ebene*
  - *BSP-Baum (**B**inary **S**pace **P**artition):*
    - Teilung erfolgt durch eine beliebige Ebene im Raum
  - *kd-Baum (**k** **d**imensional)*
    - Teilung erfolgt durch achsenparallel Ebenen im Raum

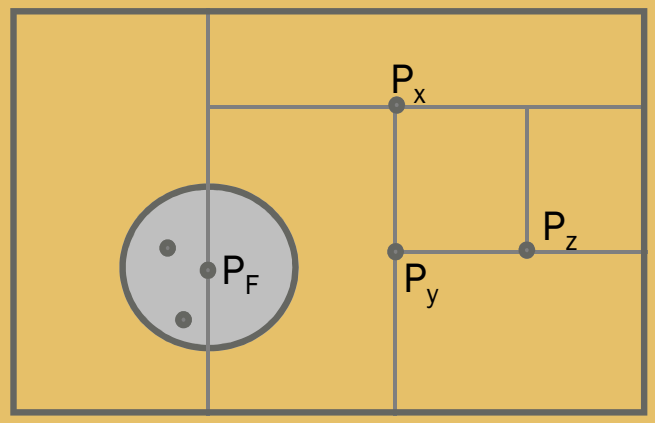


BSP-Baum



kd-Baum

Frage zum Abbruchkriterium:



Links sehen Sie die Struktur eine kd-Baums. Es wurden bereits die m nächsten Punkte zu P_F gefunden, die sich in dem grauen Umkreis verstecken. Der Umkreis hat als Radius mdist. Eigentlich könnten wir die Suche beenden.

Aber erlaubt uns das Abbruchkriterium auch die Suche zu beenden obwohl P_x noch nicht untersucht wurde?

Frage: Müssen wir damit beginnen den Bound von P_x, also den Teilbaum, der durch P_x aufgespannt wird, zu durchsuchen?

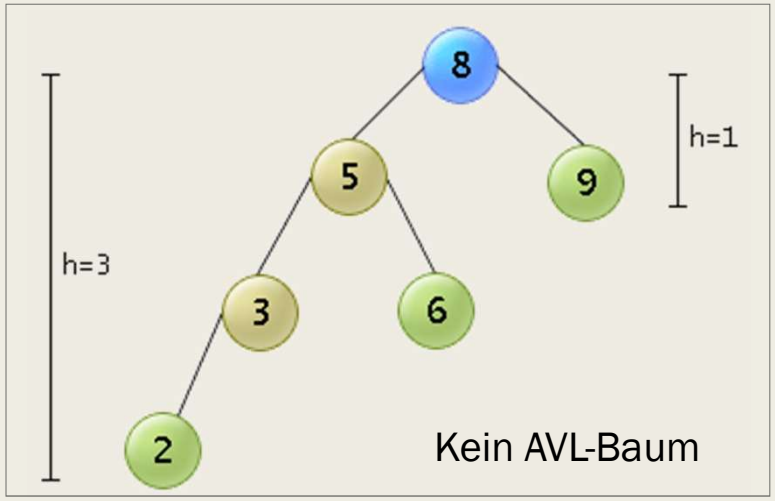
Ja oder nein?

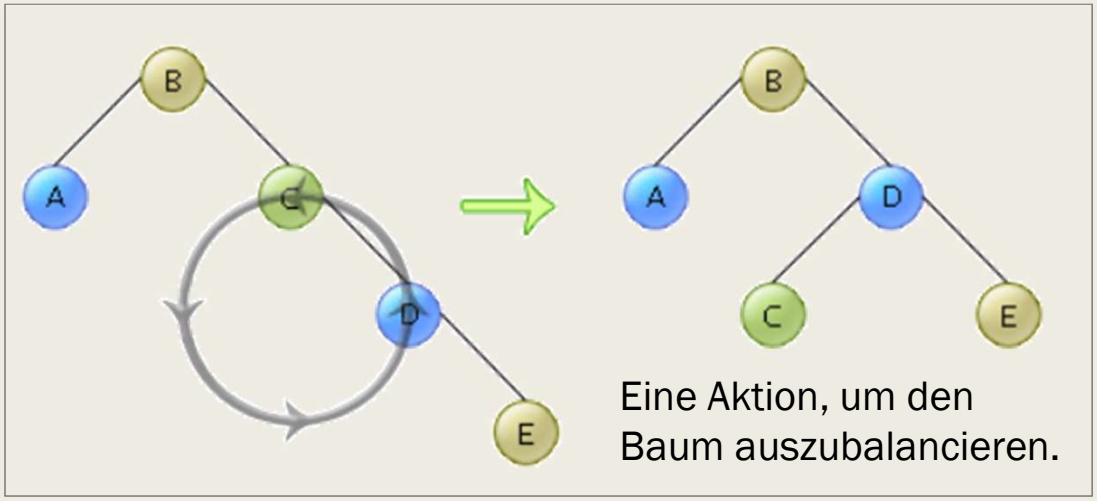
Wie sieht es mit P_y und P_z aus?

Kd-Baum im Photon Mapping Verfahren:

- kd-Baum speichert die Photon Map.
- Während des Ray-Tracing-Teils des Photon Mapping Verfahren wird nur noch im kd-Baum gesucht und keine neuen Photonen mehr eingefügt.
- In einem nicht ausbalancierten Baum kann die Suche im worst-case-Fall eine quadratische Laufzeit annehmen.
- Daher sollte der Baum ausbalanciert werden (AVL-Baum)
- Laufzeit beträgt dann : $O(\log_2 n)$

AVL-Bäume





Bilder aus: https://me-lrt.de/binarbaum-avl-tree-baum-algorithmus-datenstruktur

AVL -Bäume sind ausbalancierte Bäume, die einen maximalen Höhenunterschied (h) von 1 haben.
Der Baum in der Abbildung oben hat einen Höhenunterschied von 2, weil
$h = abs(h_{links} - h_{rechts}) \Leftrightarrow 2 = abs(3 - 1)$

Das Ausbalancieren des Baums wird während der Erstellung des Baums vorgenommen. Eine Eindruck davon, wie man dabei vorgeht, können Sie über den Link oben oder auch über Youtube-Videos bekommen.

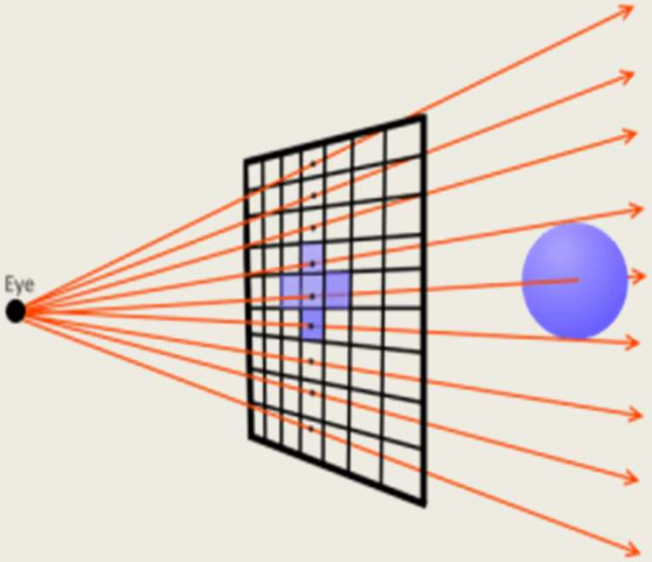
Optimierungsstrukturen sind nicht nur für Raytracer nützlich:

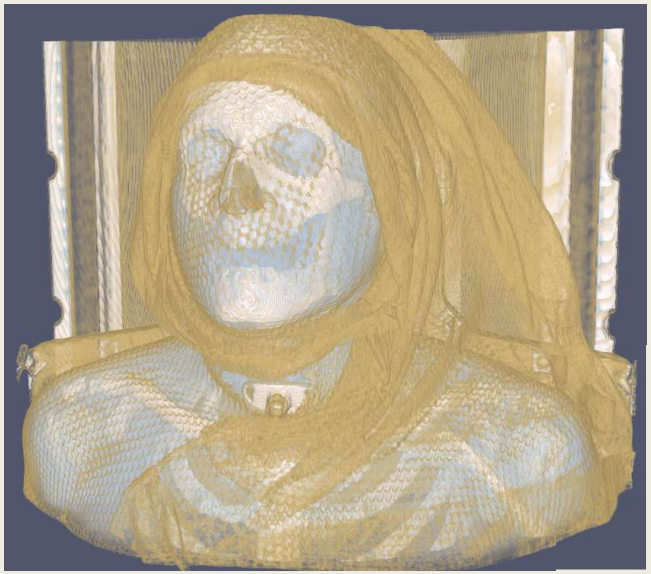
- Optimierungsstrukturen sind zur Beschleunigung der Berechnungen für Raytracing notwendig, aber auch für nicht Raytracing Anwendungen sind sie oft sinnvoll.
- Die Daten werden räumlich unterteilt, um nur auf die Daten zugreifen zu müssen, die gerade gebraucht werden, weil sie zum Beispiel sichtbar sind.
- Für Computer Graphik ist es sinnvoll, die Daten räumlich zu unterteilen, da die Datenbasis meist aus 2D- und 3D-Geometrien, Volumendaten (Voxeln), Texturen, also Bildern oder Point-Clouds (bsp. aus Scans) besteht.
- Die meisten Optimierungsstrukturen arbeiten hierarchisch. Was bedeutet, dass der obere Level die Daten der darunter liegenden Leveln umfassen. Der oberste Level beinhaltet also die gesamte Szene.
- Optimierungsstrukturen werden überall dort genutzt, wo „gesucht“ werden muss, ob eine Geometrie oder bestimmte Elemente, wie beispielsweise die Photonen, an einer Aktion beteiligt sind oder nicht.

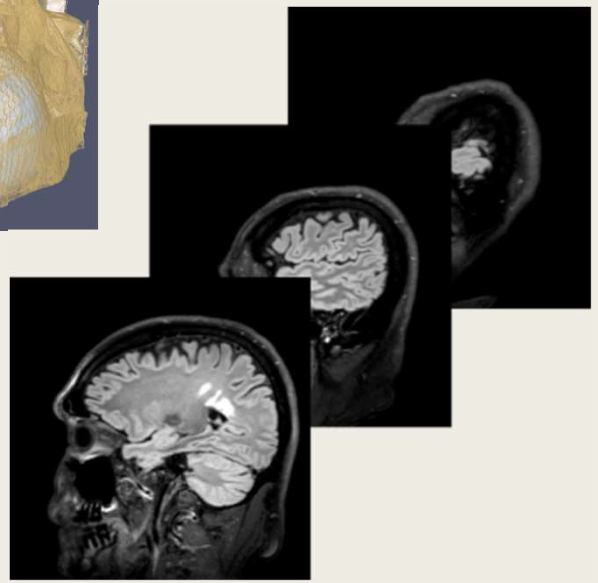
Beispielanwendungen in denen die Optimierungsstrukturen genutzt werden:

- **Raytracing:** Prüfen, ob der Sehstrahl oder der Schattenfühler eine Geometrie schneidet und wenn ja, an welcher Stelle?
- **Raycasting:** Prüfen welche Voxel vom Sehstrahl durchdrungen werden
- **Kollissioneerkennung:** Prüfen, welche Geometrien oder Partikel mit dem aktuellen, sich in Bewegung befindlichen Geometrien oder Partikel kollidieren
- **Sichtbarkeitsprüfung bei polygonalen Renderern:** Prüfen, welche Geometrien im Viewfrustum und welche außerhalb liegen

Raycasting

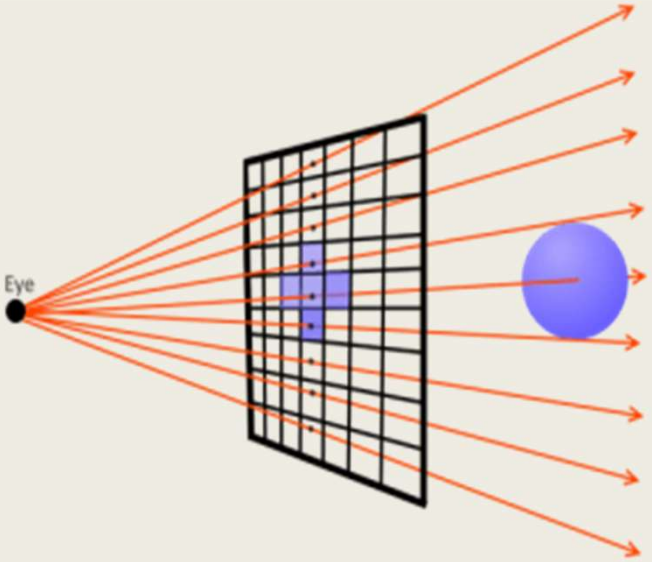


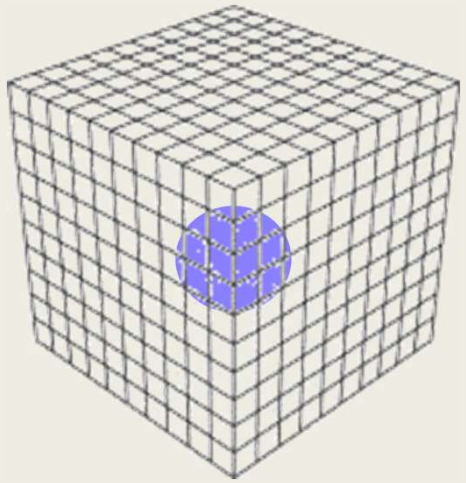


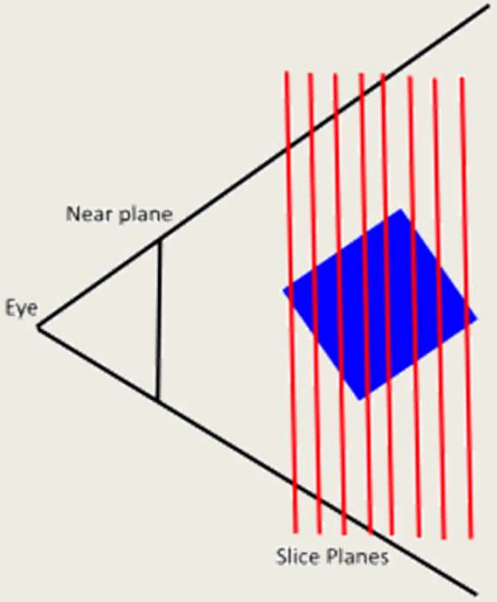


Bilder aus: [Ray Cast Volume Rendering – Andrew Wilson]

Raycasting

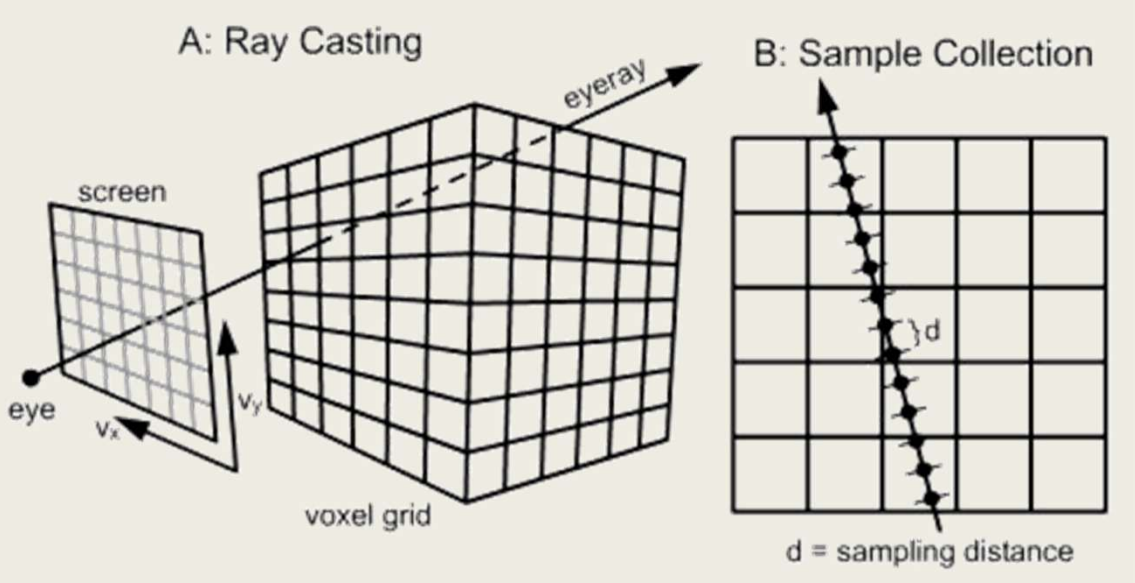


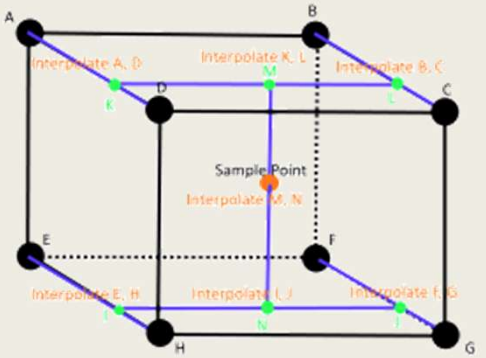




Bilder aus: Ray Cast Volume Rendering – Andrew Wilson

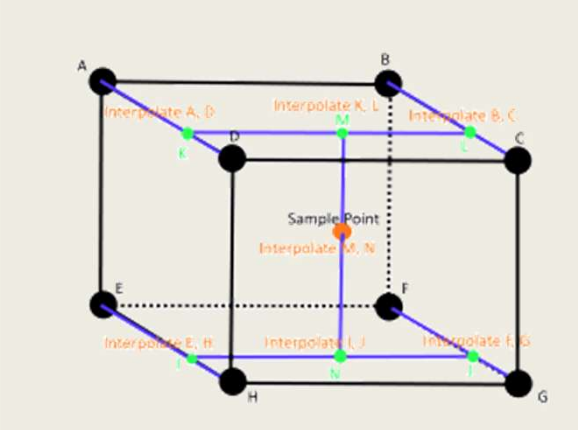
Raycasting

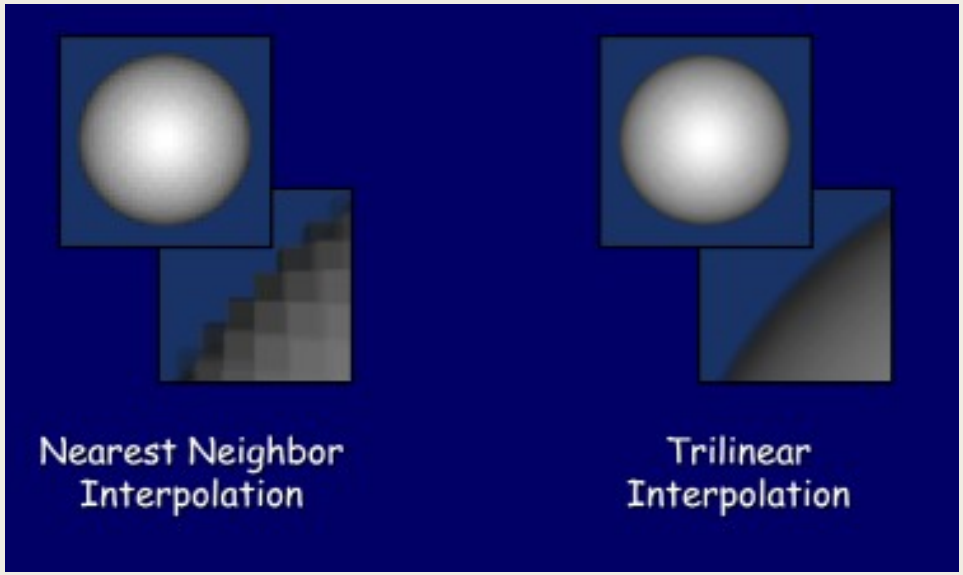




Bilder aus: [Ray Cast Volume Rendering – Andrew Wilson]

Raycasting





Bilder aus: Ray Cast Volume Rendering – Andrew Wilson

Weitere „Spatial Data Structures“ zur Optimierung (siehe [1])

- Bounding Volume Hierarchies
- BSP-Trees
  - *Axis-Aligned BSP Trees (AABB)* – *kd-Bäume sind eine Variante davon*
  - *Polygon-Aligned BSP Trees*
- Quadtrees & Octrees

Welche Optimierungsstruktur teilt den Raum regulär, also in regelmäßigen Abständen auf:

a.) BSP-Baum

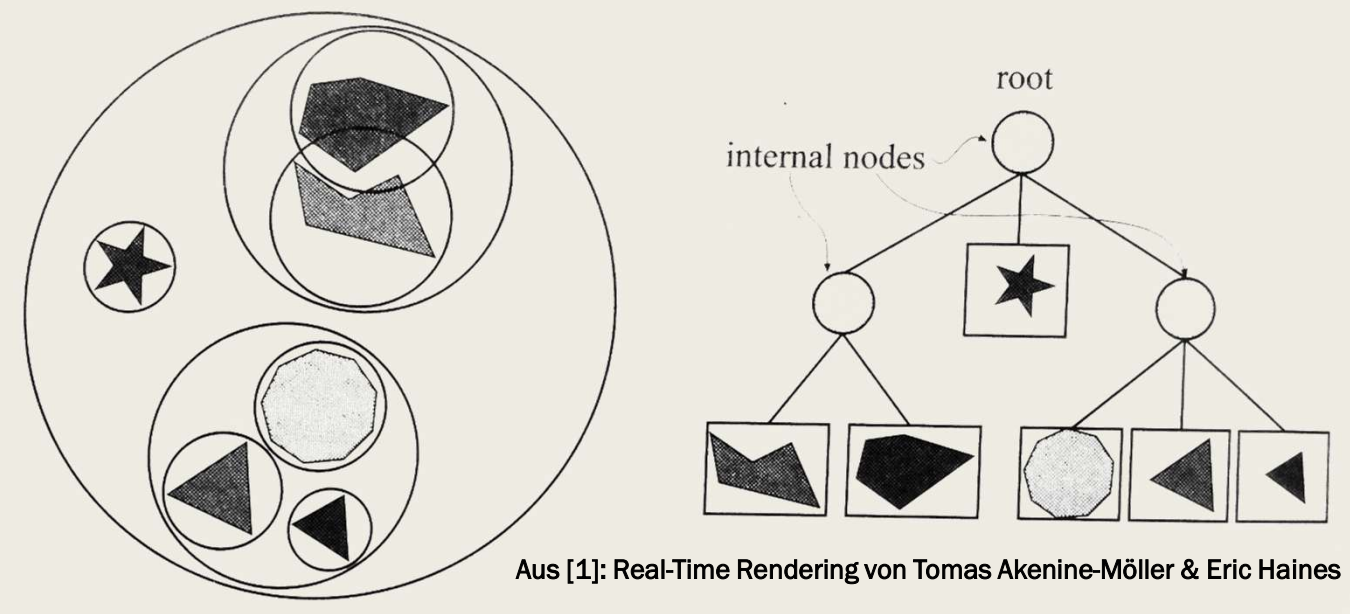
b.) Octree

Weitere „Spatial Data Structures“ zur Optimierung (siehe [1])

Bounding Volume Hierarchies (BVH)

- sie sind weder regulär, wie die Octrees, noch irregulär, wie die BSP-Trees aufgebaut
- sie unterteilen auch nicht den Raum
- sie umschließen den Raum in dem sich die Geometrien befinden
- es kann also durchaus so sein, dass sie nicht die gesamte Szene abdecken
- ihr Aufbau ist aber genau, wie bei den BSP- oder Octrees hierarchisch strukturiert

Bounding Volume Hierarchies (BVH)



Aus [1]: Real-Time Rendering von Tomas Akenine-Möller & Eric Haines

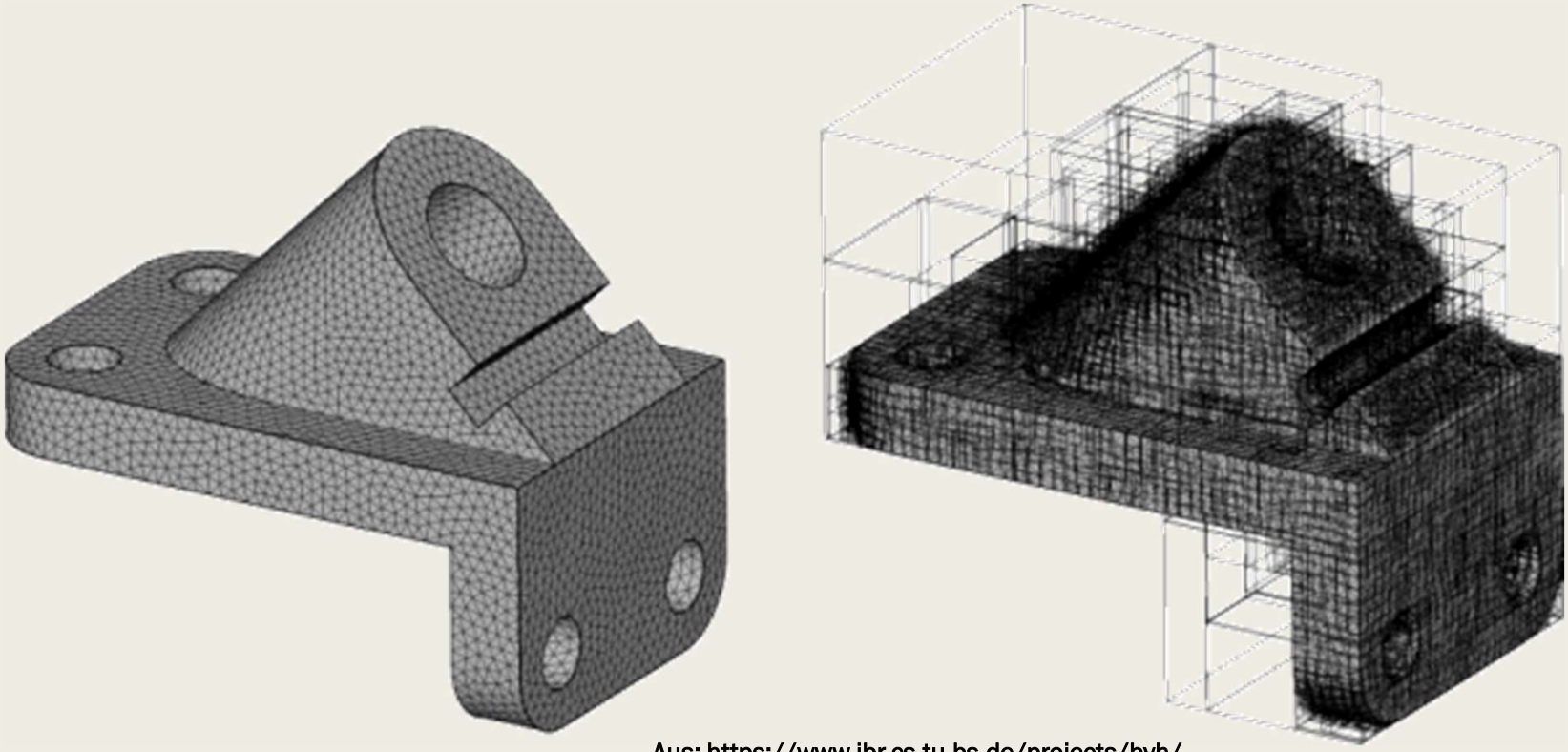
- **Links:** Die einzelnen Geometrien der Szene werden jeweils mit einer Bounding-Sphere umfasst. Nahe beieinander liegende Bounding-Spheres werden wiederum mit einer Bounding-Sphere umfasst.
- **Rechts:** Die BVH der gesamten Szene als Baumstruktur

Bounding Volume Hierarchies (BVH)

Punkte, die man berücksichtigen sollte:

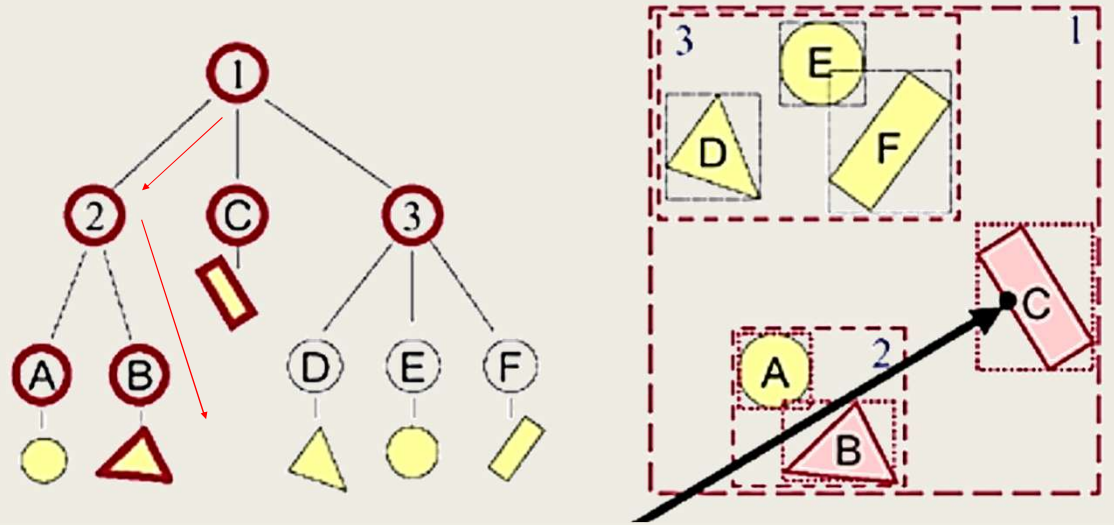
- Die Bounding Volumes (BV), die die Geometrien umschließen, sollten wesentlich einfacher sein, als die darin befindlichen Geometrien: Kugeln, Würfel oder Quader, die zum Weltkoordinaten-system achsenparallel angeordnet sind, erfüllen diese Voraussetzungen beispielsweise.
- Die Berechnung mit den BV sollte wesentlich schneller durchgeführt werden können als mit den darin enthaltenen Geometrien. Nur so kann die gewünschte Beschleunigung erzielt werden.
- Die BV selbst wird nicht gerendert und leistet auch sonst keinen visuellen Beitrag zum Rendern. Sie ist eine Struktur, die ausschließlich zur Beschleunigung der Berechnungsgeschwindigkeit beim Rendern und sonstigen Aufgaben genutzt wird.
- Mehr dazu bspw. in „Introduction to Algorithms (MIT Press)" von Cormen, Leiserson und Rivest.

Beispiel für eine Bounding Volume Hierarchie (BVH)



Aus: https://www.ibr.cs.tu-bs.de/projects/bvh/

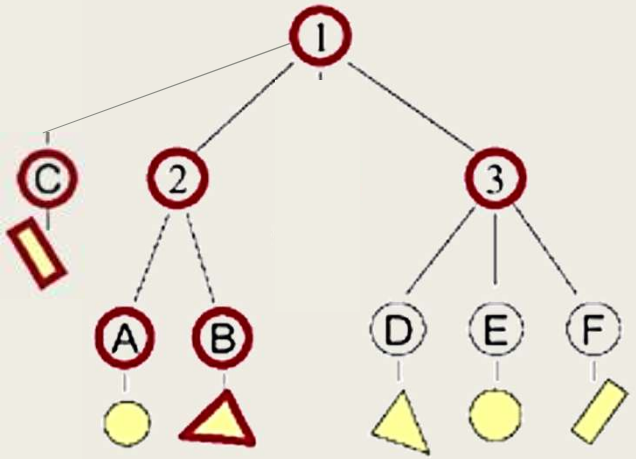
Ray-Casting am Beispiel einer Bounding Volume Hierarchie (BVH)

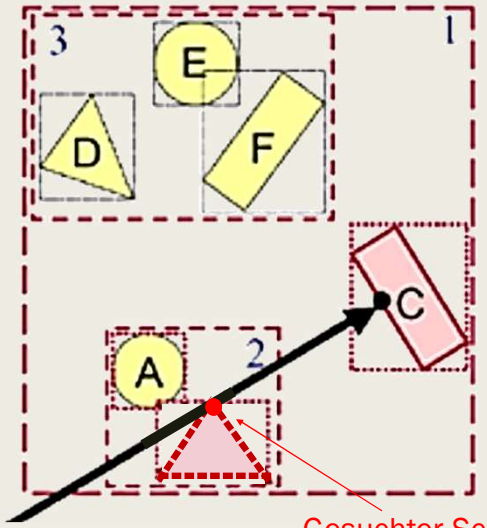


Bilder aus:
https://cseweb.ucsd.edu/~viscomp/classes/cse167/wi19/slides/lecture16.pdf

Traversierung der BVH bis zum Dreieck. Jetzt muss getestet werden, ob der Strahl einen Schnittpunkt mit dem Dreieck hat. Wenn nicht (wie in unserem Fall), muss die BVH weiter traversiert werden, um den Schnittpunkt mit C zu finden.

Ray-Casting am Beispiel einer Bounding Volume Hierarchie (BVH)





Bilder aus: https://cseweb.ucsd.edu/~viscomp/classes/cse167/wi19/slides/lecture16.pdf

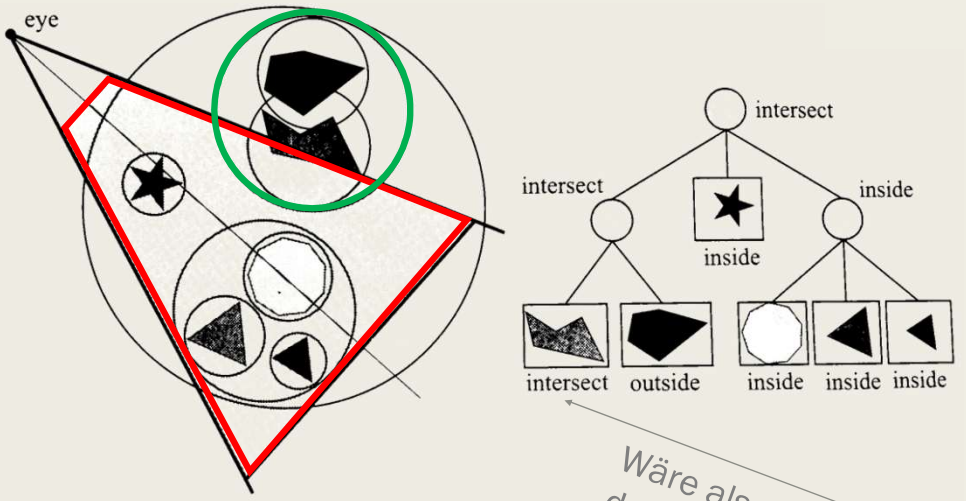
Gesuchter Schnittpunkt

Stellen Sie sich jetzt das folgende Szenario vor: Der Strahl schneidet das Dreieck B und das Quadrat C. Aber in der BVH wird das Quadrat C vor dem Dreieck B traversiert. Damit wird auch zuerst der Schnittpunkt mit C berechnet. Gesucht wird aber der Schnittpunkt mit B, weil er näher am Betrachter liegt.

Beachten Sie, dass der erste gefundene Schnittpunkt in einer BVH nicht unbedingt, der am nächsten zum Betrachter liegende Schnittpunkt sein muss!

Hierarchical View Frustum Culling mit

Bounding Volume Hierarchie (BVH)



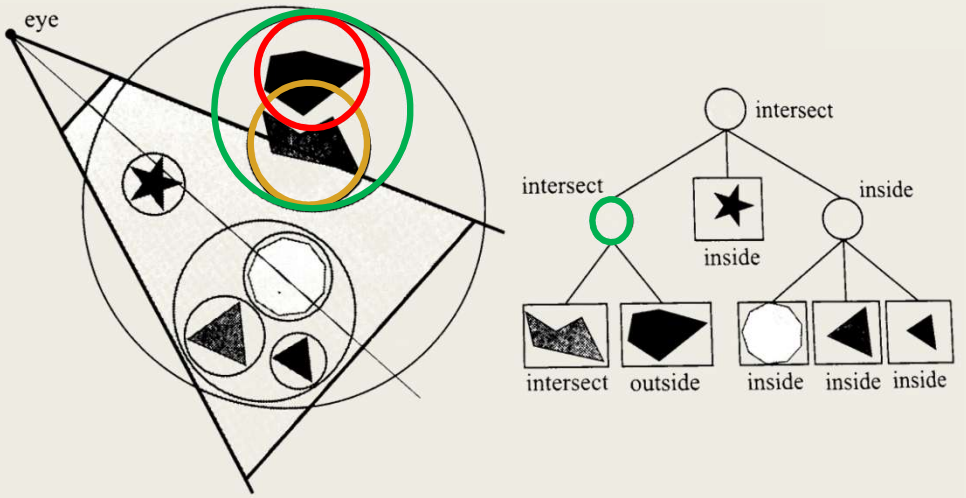
Bilder aus:
https://cseweb.ucsd.edu/~viscomp/classes/cse167/wi19/slides/lecture16.pdf

Wäre also innerhalb des Viewfrustums

- Nur die Geometrien, die im View Frustum (rot markiert) enthalten sind, werden gerendert. Alle anderen Geometrien sind außerhalb des Sichtbereichs der Virtuellen Kamera.
- Wird festgestellt, dass ein Bounding Volume (BV) innerhalb oder außerhalb des View Frustums ist, wird die Traversierung dieses Zweigs eingestellt.
- Wenn ein BV das View Frustum schneidet (hier grüne BV), dann müssen die Kinder weiter getestet werden.
- Ist man bei der Geometrie (Blatt im Baum) angekommen und stellt fest, dass eine dieser Geometrie geschnitten wird, gibt es verschiedene Strategien. Die einfachste ist die geschnittene Geometrie insgesamt als „innerhalb“ zu klassifizieren.

Hierarchical View Frustum Culling mit

Bounding Volume Hierarchie (BVH)

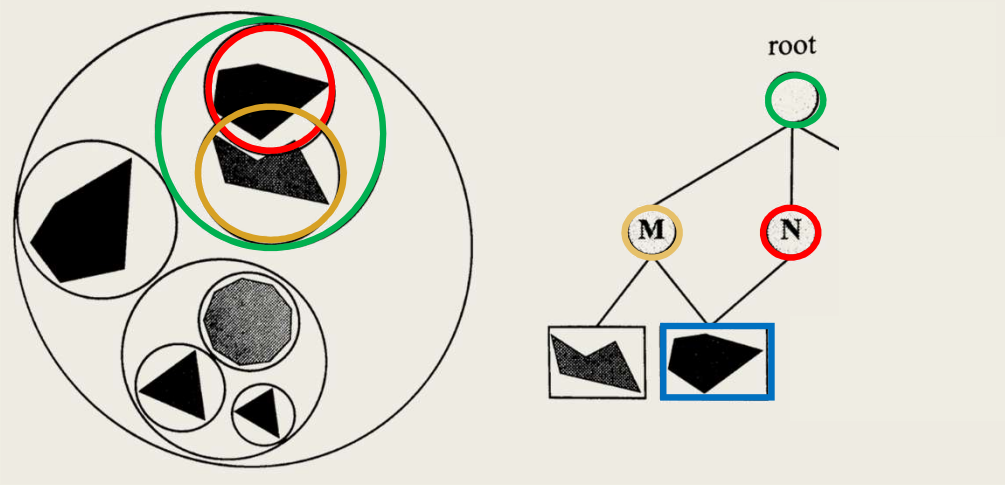


Bilder aus:
https://cseweb.ucsd.edu/~viscomp/classes/cse167/wi19/slides/lecture16.pdf

- Während das grüne Bounding Volume (BV) in der BVH als Gruppierungsknoten abgebildet wurde, fehlen die beiden, rot und ockergelb gekennzeichneten BVs in der Baumstruktur. In den Baum werden die Geometrien (Kästchen) ohne Boundingsphere dargestellt.
- Während im roten BV nur eine Geometrie enthalten ist, sind im ockergelben BV zwei Geometrien enthalten, wobei eine davon die Geometrie ist, die bereits im roten BV umschlossen wurde.
- Die Frage ist, wie die rote und ockergelbe BV nun in die Hierarchie integriert werden kann?

Hierarchical View Frustum Culling mit

Bounding Volume Hierarchie (BVH)



Bilder aus:
https://cseweb.ucsd.edu/~viscomp/classes/cse167/wi19/slides/lecture16.pdf

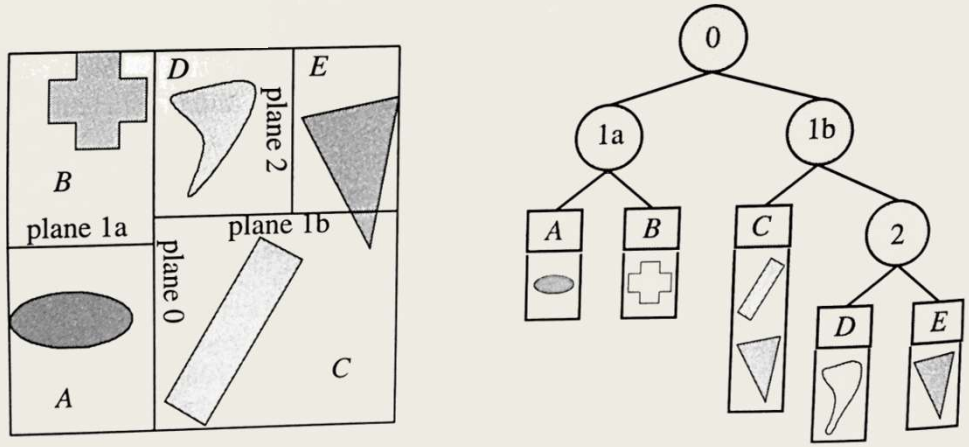
Die Frage ist, wie die rote und ockergelbe BV nun in die Bounding Volume Hierarchie integriert werden kann?

- Ziel ist alle Geometrien nur ein einziges Mal vorzuhalten.
- Auf die blau markierte Geometrie würden demnach zwei BVs verweisen.
- In der praktischen Anwendung bedeutet dass, das sowohl von der BV M als auch vom BV N Anfragen zum Test der blau umrahmten Geometrie kommen könnte.

Binary Space Partitioning Trees (BSP)

- Diese Bäume werden erzeugt indem man den 3D-Raum mit Hilfe einer Ebene oder den 2D-Raum mit einer Geraden in zwei Hälften teilt.
- Danach sortiert man die Geometrien in die zwei Hälften ein.
- Die Unterteilung wird rekursiv weitergeführt. Das Prinzip kennen Sie vom kd-Baum.
- Eine wichtige Eigenschaft ist, dass die Geometrien „räumlich sortiert“ im BSP vorliegen, so dass Nachbarschaften erkannt werden können (siehe kd-Baum zu Anfang des Kapitels). Das unterscheidet sie von den BVH in denen die Geometrien zwar gebündelt, aber die Bündel selbst unsortiert angeordnet sind.
- BSP-Bäume können so aufgebaut werden, dass sie sehr schnell, die am nächsten zum Betrachter liegenden Polygone ermitteln können. Die BVH können das nicht leisten, da sie nicht garantieren, dass die Geometrien räumlich sortieren angeordnet sind.

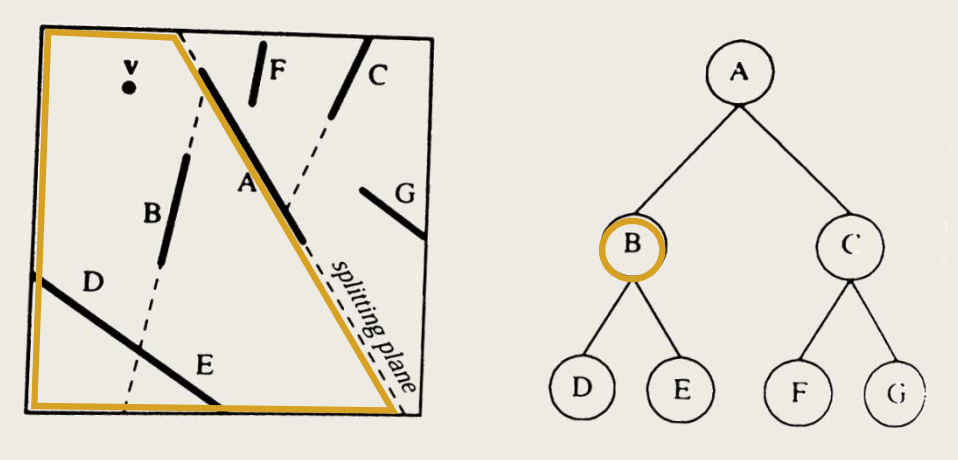
Axis-Aligned BSP Trees (AABB)
kd-Bäume sind eine Variante davon



Bilder aus:
https://cseweb.ucsd.edu/~viscomp/classes/cse167/wi19/slides/lecture16.pdf

- Achsenparallele Unterteilung des Raums ähnlich, der bereits vorgestellte kd-Baum Unterteilung. Nur geben jetzt nicht die Photonen sondern die Geometrien vor, wie der Raum geteilt wird.
- Es gibt viele mögliche Strategien, um die beste Aufteilung der Räume zu finden.
- Das vorgestellte kd-Baum Verfahren erzeugt auch ein Variante von AABB. Aber selbst zur Erzeugung von kd-Bäumen gibt es diverse Verfahren mit bestimmten Vor- und Nachteilen.
- **Wichtig beim AABB ist, dass die Unterteilung durch Ebenen oder Geraden passiert, die an den Koordinatenachsen des Weltkoordinatensystems ausgerichtet sind – axis-aligned.**

Polygon-Aligned BSP Trees

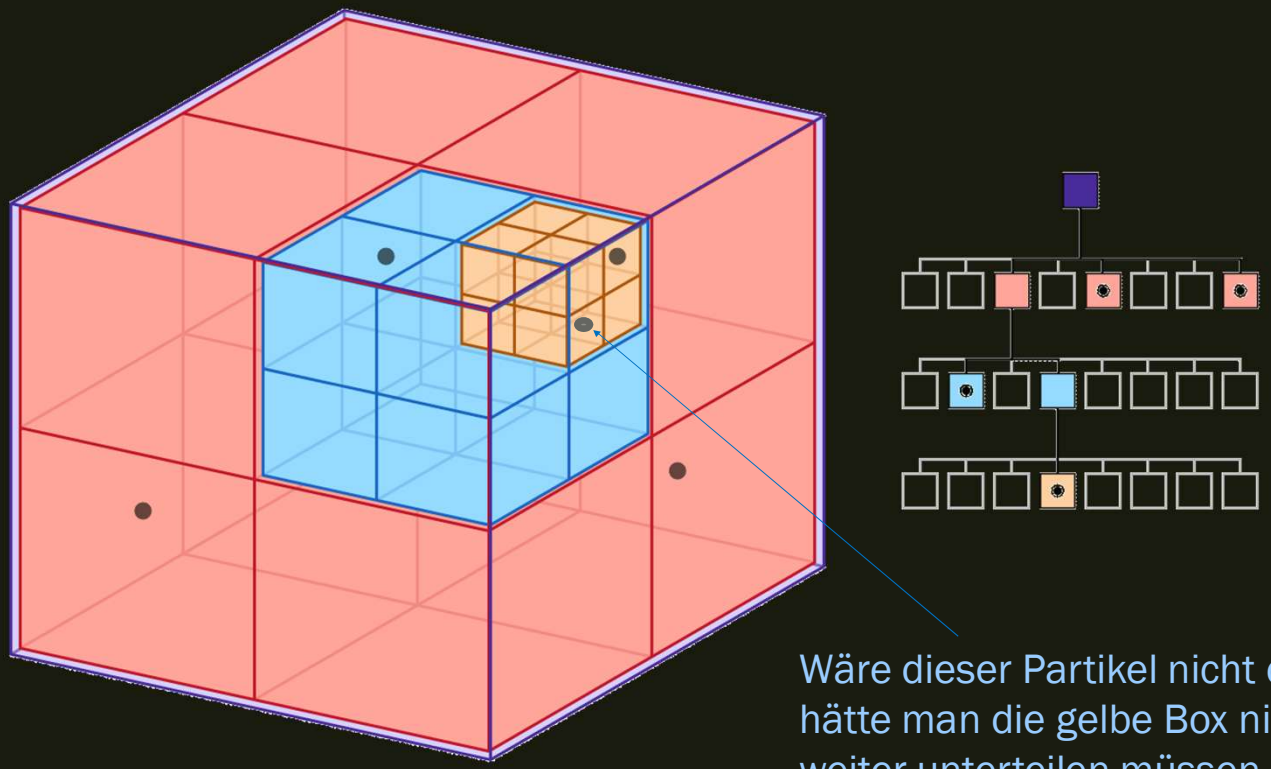


Für den Teilungsraum unten im Bild (ocker-farben eingerahmt) wurde B als Teilungs-gerade gewählt, da die Gerade, die durch B geht, die einzige ist, die den Raum in zwei Räume aufteilt, welche Teilungsgerade A teilt. B zerschneidet, das Polygon in der Ecke in zwei Teile nämlich in D und E.

Bilder aus:
https://cseweb.ucsd.edu/~viscomp/classes/cse167/wi19/slides/lecture16.pdf

- Die Raumunterteilung geschieht im 3D-Raum anhand der Ebene eines ausgewählten Polygons (eines geometrischen Objekts). Im Beispiel oben ist das A. Im nächsten Schritt sucht man in jeder Raumhälfte ein Polygon, das den Raum so teilt, das sich danach möglichst gleich viele Polygone in den beiden Teilräumen befinden. Im Beispiel oben trifft das auf C zu. Das Verfahren wird auf diese Weise rekursiv für die so entstandenen Raumhälften fortgesetzt.
- Eines der einfachsten Strategien zum Teilen beruht auf dem „least cross-Kriterium“. Hier wird eine Anzahl an Polygonen im zu teilenden Raum zufällig ausgewählt. Von den ausgewählten Polygonen, wird das genommen, welches die wenigsten Polygone zerschneidet, da diese Berechnung der Zerschneidung teuer ist.

Octrees



Wäre dieser Partikel nicht da, hätte man die gelbe Box nicht weiter unterteilen müssen.

Bild aus: https://developer.apple.com/documentation/gameplaykit/gkoctree

Grundidee: Reguläres Gitter, das den 3D-Raum gleichmäßig unterteilt.

Nur die Bereiche in denen sich Geometrien befinden, werden weiter unterteilt.

In diesem Beispiel sind es keine Geometrien sondern graue Partikel, die der Octree-unterteilung zugrunde liegen.

Ziel ist immer ein Element pro Box. Gibt es mehr Element muss weiter geteilt werden.

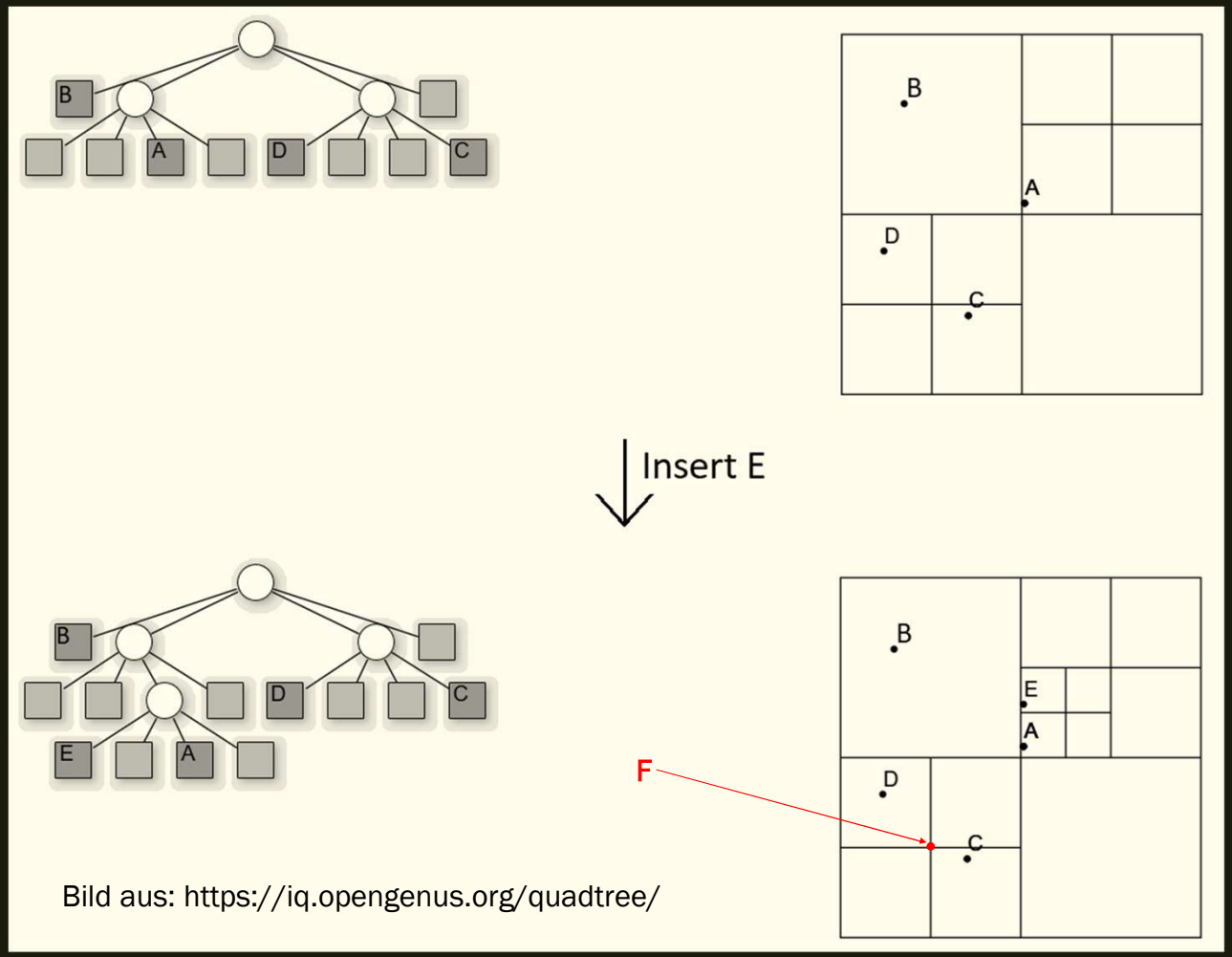


Bild aus: https://iq.opengenus.org/quadtree/

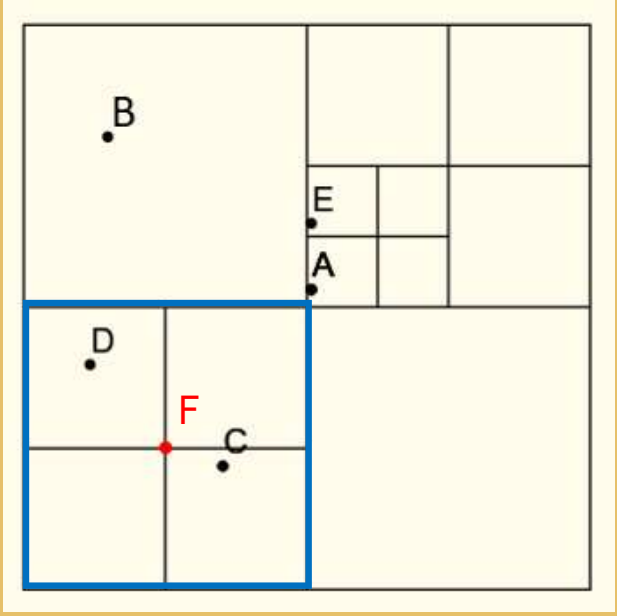
Quadtree

Äquivalent zum Octrees im 2D-Raum

Hier sieht man die sich der Quadtree weiter entwickelt wenn das zusätzliche Element E eingefügt wird.

Was passiert, wenn ich F einfüge und F genau auf der Mitte des Teilungs-kreuzes liegt?

Was passiert, wenn ich F einfüge und das F genau auf der Mitte des Teilungskreuzes liegt?



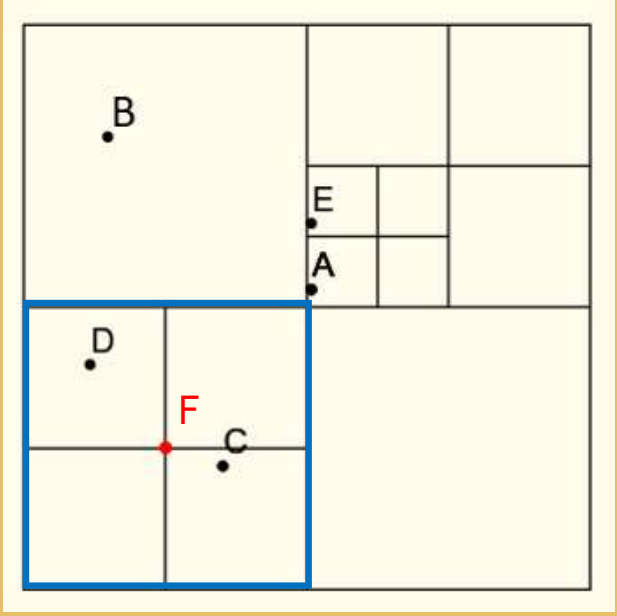
A.) **Nichts.** Der Octree bleibt wie er ist.

B) Das Kästchen mit dem **D** drin wird in vier Teilkästchen aufgeteilt. Alle anderen werden nicht unterteilt.

C) Die Kästchen mit dem **D** und **C** drin, werden unterteilt.

D) Alle **vier Kästchen**, im blauen Rahmen, werden geteilt.

Was passiert, wenn ich F einfüge und das F genau auf der Mitte des Teilungskreuzes liegt?



A.) **Nichts.** Der Octree bleibt wie er ist.

B) Das Kästchen mit dem **D** drin wird in vier Teilkästchen aufgeteilt. Alle anderen werden nicht unterteilt.

C) Die Kästchen mit dem **D** und **C** drin, werden unterteilt.

D) Alle **vier Kästchen**, im blauen Rahmen, werden geteilt.

Erstellung eines Quadtrees:

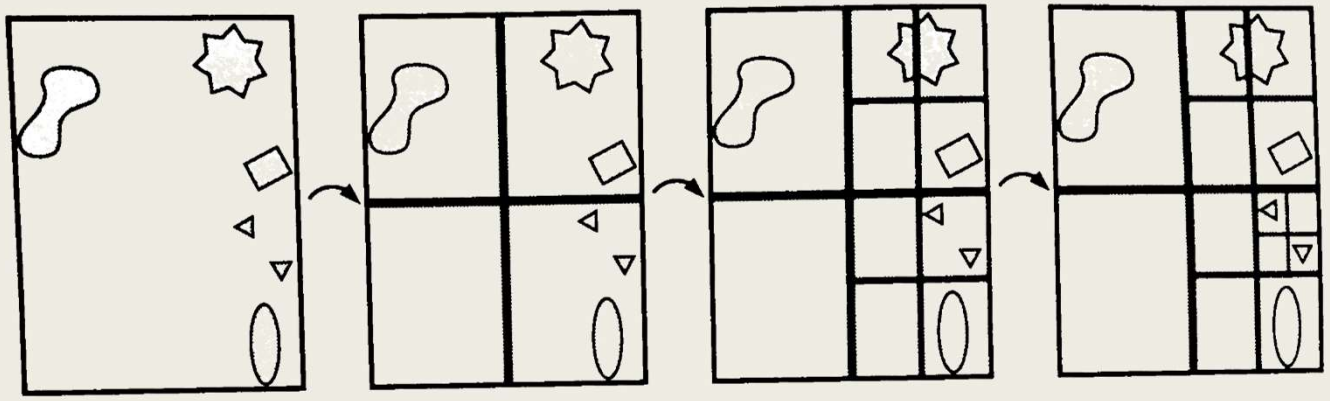
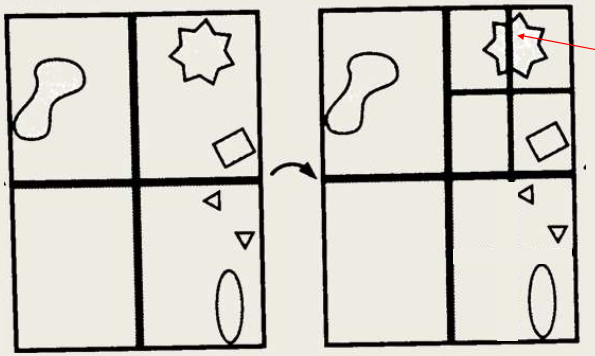


Bild aus: https://cseweb.ucsd.edu/~viscomp/classes/cse167/wi19/slides/lecture16.pdf

- Die Aufteilung eines Raums als Quadtree erfolgt immer von Mittelpunkt des 2D-Raums ausgehend. Vom Mittelpunkt aus wird der Raum in vier achsenparallele Teilräume unterteilt.
- Die Unterteilung erfolgt solange, bis es nur noch ein Element (Geometrie, Partikel oder was auch immer) pro Teilraum gibt. (Alternativ kann nach x-Teilungen abgebrochen werden.)
- Dabei kann es vorkommen, dass durch die Teilung Geometrien zerschnitten werden, wie man am Stern oben rechts sehen kann. Dies möchte man wenn möglich vermeiden.
- Die Erstellung eines Octrees erfolgt im 3D-Raum äquivalent zum Quadtree.

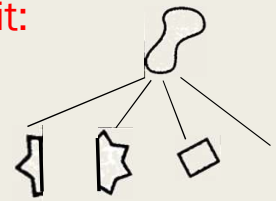
Erstellung eines Quadtrees:



Problemfall für dessen Lösung es mindestens drei Alternativen gibt:

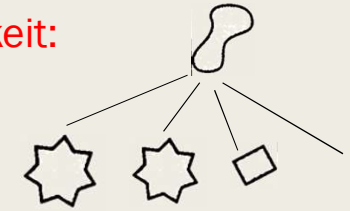
Bildaus:
https://cseweb.ucsd.edu/~viscomp/classes/cse167/wi19/slides/lecture16.pdf

1. Lösungsmöglichkeit:



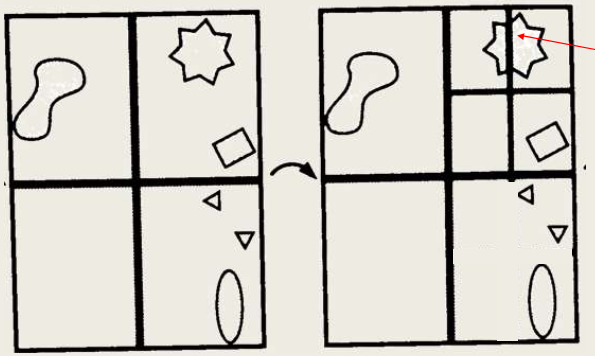
- Betroffenes Element zerschneiden.
- Erzeugt mehr Geometrien, die erzeugt und anschließend verwaltet werden müssen, was Einbußen in der Effizienz mit sich bringt.

2. Lösungsmöglichkeit:



- Betroffenes Element beiden Teilräumen zuordnen.
- Verkompliziert die Konstruktion eines Quadtrees und geht zu Lasten der Effizienz.

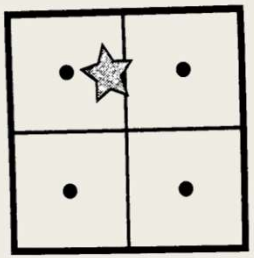
Erstellung eines Quadtrees:

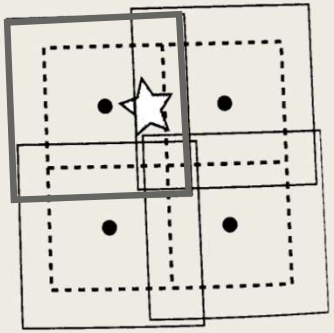


Problemfall für dessen Lösung es mindestens drei Alternativen gibt:

Bild oben und unten aus:
https://cseweb.ucsd.edu/~viscomp/classes/cse167/wi19/slides/lecture16.pdf

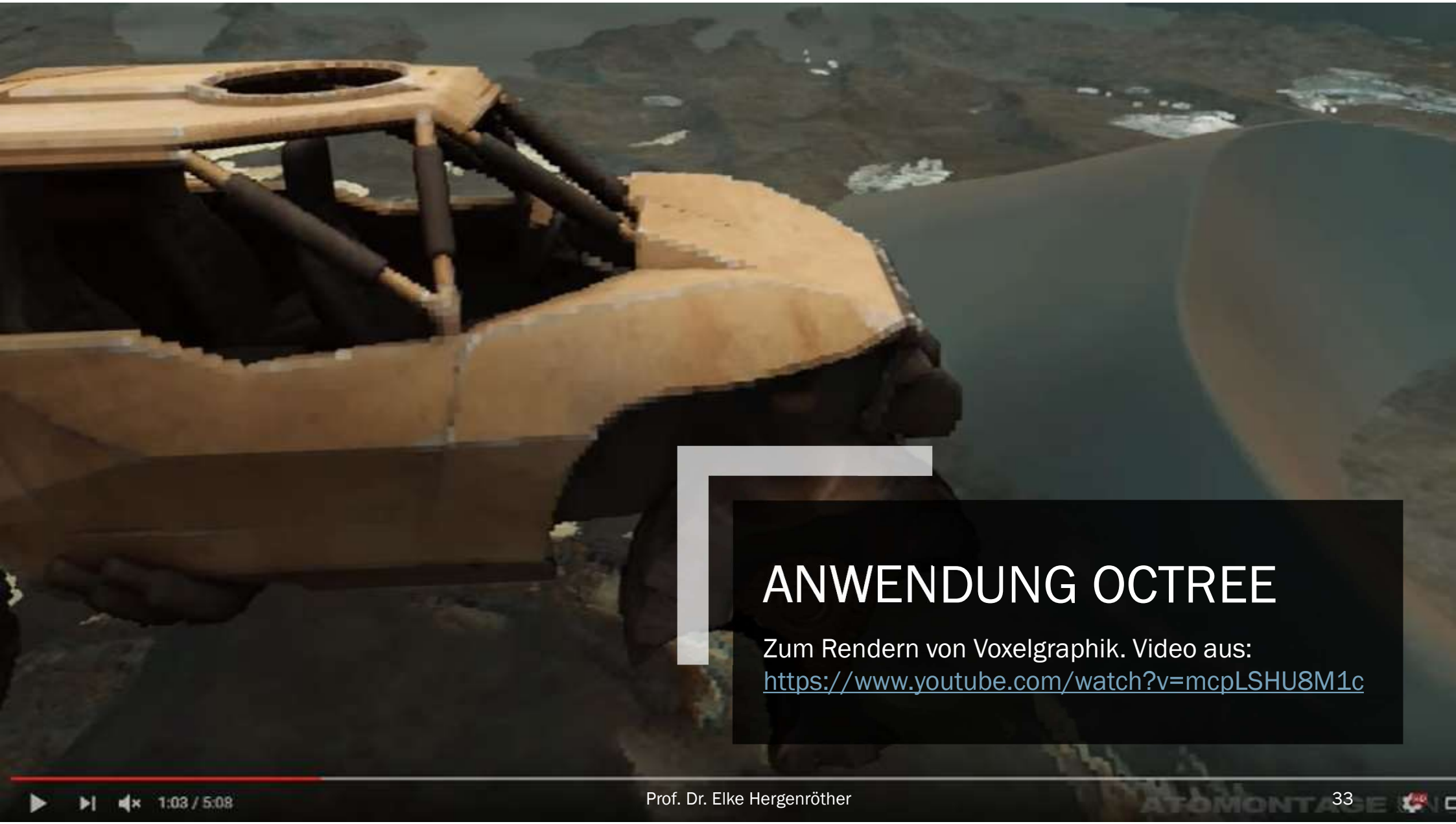
3. Lösungsmöglichkeit:





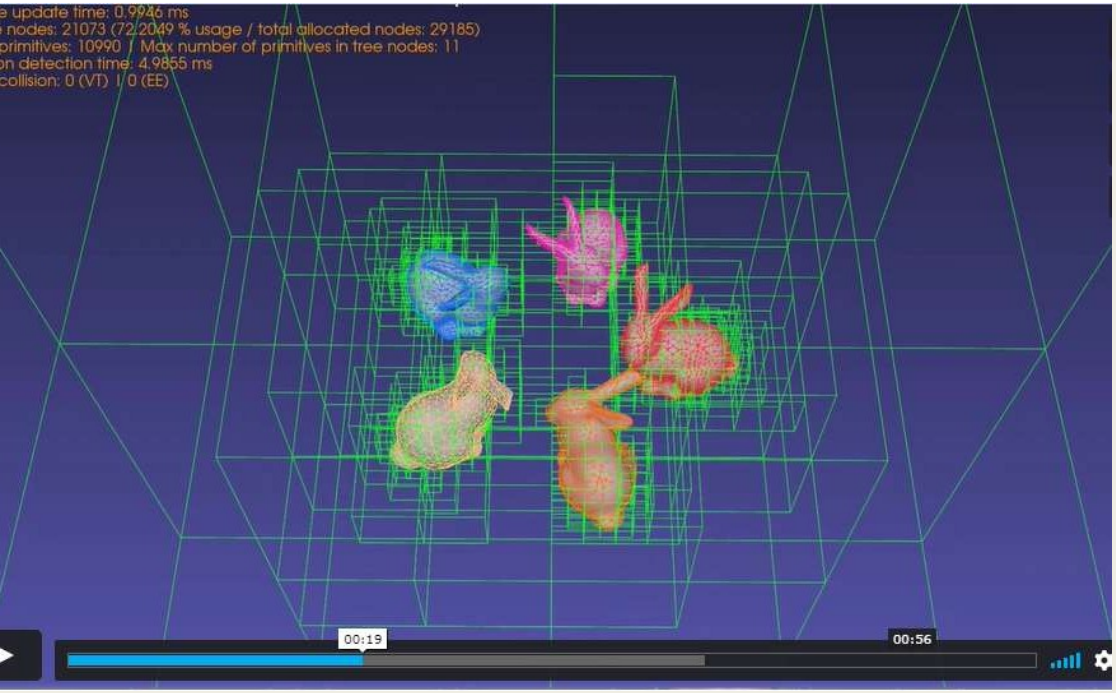
Der „Loose Octree“ *) : Das Vorgehen zur Erstellung des Octrees ist das gleiche, wie beim normalen Octree. Nur die Größe der Boxen „relaxed“. Wenn die Seitenlänge einer Box normalerweise l beträgt, wird nun mit l mit einen Faktor k>1 multipliziert. Links ist der Fall k=1,5 aufgezeigt. Die einzelnen Kästen des Quadtrees werden also größer, so dass nun kleinere Objekte, die in der Nähe einer Teilungsgrenze liegen, einem Kästchen zugeordnet werden können und Lösung 1 und 2 nicht benötigt werden.

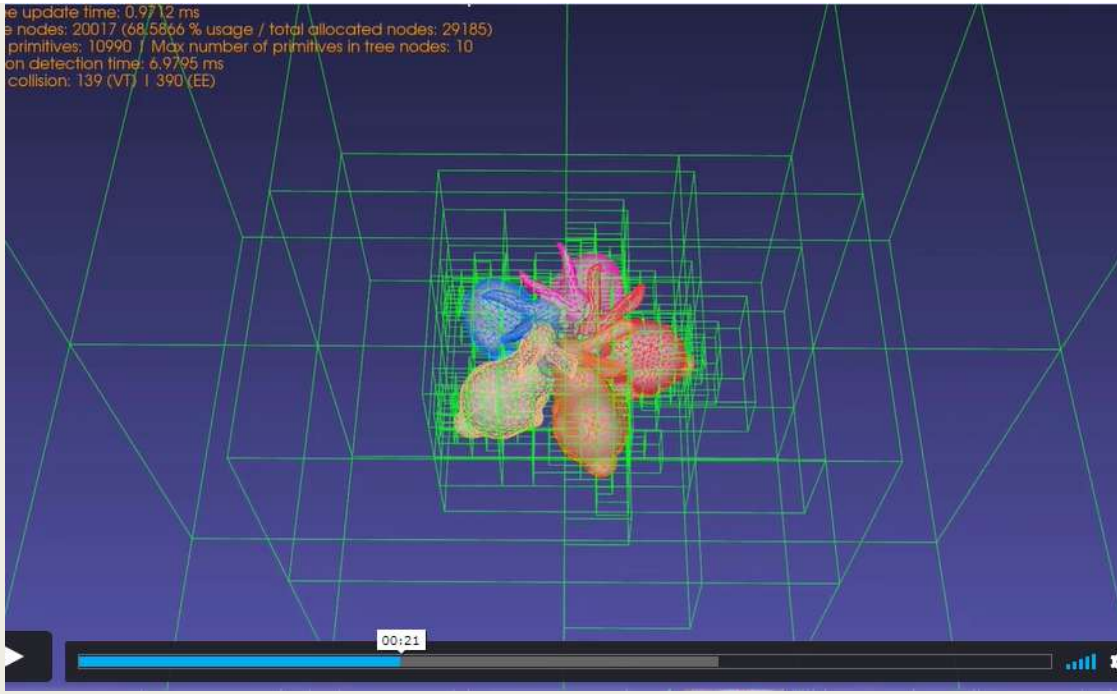
*) „Loose Octree“ U. Thatcher in Mark DeLoura, ed., Game Programming Gems, Charles River Media, pp. 444-453, 2000



ANWENDUNG OCTREE

Zum Rendern von Voxelgraphik. Video aus:
https://www.youtube.com/watch?v=mcpLSHU8M1c

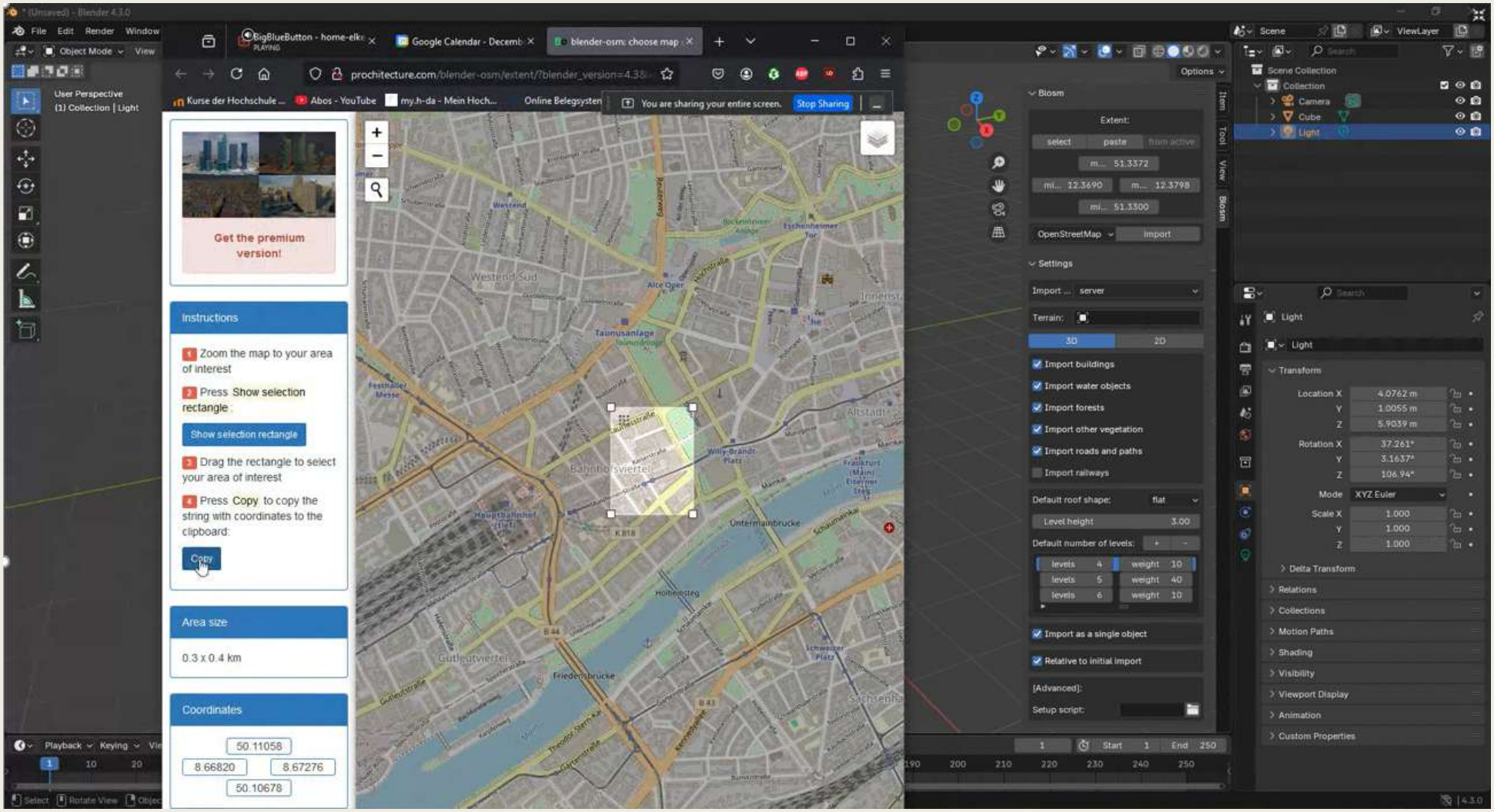




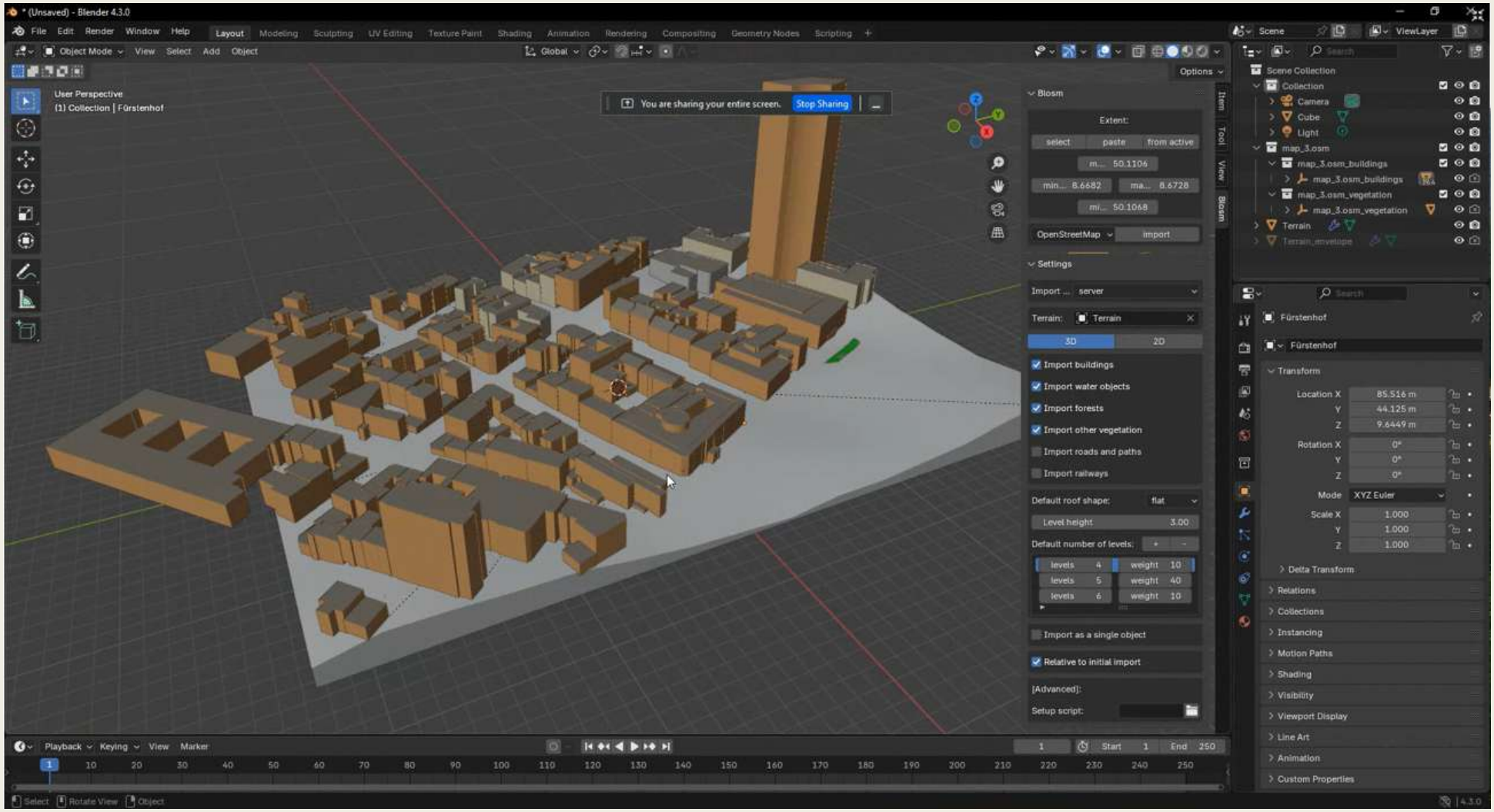
Anwendung Octree

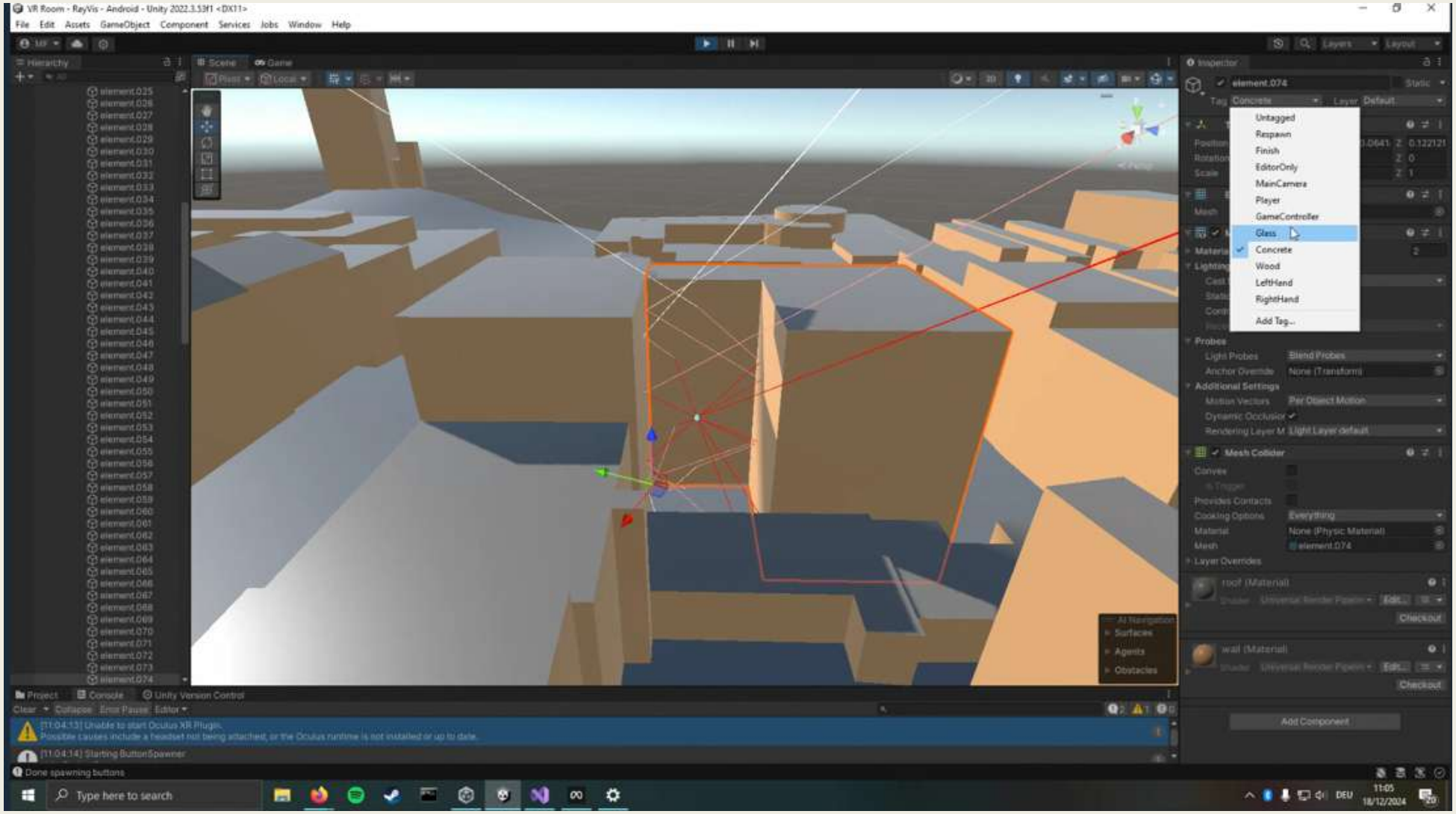
- Kollisionsdetektion
- Bilder aus Video: https://vimeo.com/358097791

Ausschnitt aus OpenStreetMap



Ausschnitt aus OpenStreetMap




VR Room - RayVis - Android - Unity 2022.3.53f1 <DX11>
File Edit Assets GameObject Component Services Jobs Window Help
Hierarchy
Scene
Game
Inspector
element.074
Tag Concrete
Layer Default
Untagged
Respawn
Finish
EditorOnly
MainCamera
Player
GameController
Glass
Concrete
Wood
LeftHand
RightHand
Add Tag...
Probes
Light Probes
Blend Probes
Anchor Override
None (Transform)
Additional Settings
Motion Vectors
Per Object Motion
Dynamic Occlusion
Rendering Layer M
Light Layer default
Mesh Collider
Convex
Is Trigger
Provides Contacts
Cooking Options
Everything
Material
None (Physic Material)
Mesh
element.074
Layer Overrides
roof (Material)
Checkout
wall (Material)
Checkout
Add Component
AI Navigation
Surfaces
Agents
Obstacles
Project
Console
Unity Version Control
Clear
Collapse
Error Pause
Editor
[11:04:13] Unable to start Oculus XR Plugin.
Possible causes include a headset not being attached, or the Oculus runtime is not installed or up to date.
[11:04:14] Starting ButtonSpawner
Done spawning buttons
Type here to search
DEU
11:05
18/12/2024

Literatur:

1. Real-Time Rendering, Tomas Akenine-Möller & Eric Haines (Hauptquelle)
2. M. Pharr, G. Humpherys: „ *Physically Based Rendering*“, Morgan Kaufmann Verlag, 2004
3. J. Foley, A. van Dam, St. Feiner and J. Hughes: *„Computer Graphics: Principles and Practice“*, 2nd Ed.,Addison-Wesley, 1990
4. P. Dutre, K. Bala, P. Bekaert: „Advanced Gobal Illumination“, 2nd Ed., A K Peters, 2006
5. H. W. Jensen: *„Realistic Image Synthesis Using Photon Mapping*“, A K Peters, 2001
6. P. Tipler: „*Physik*“, Spektrum Akademischer Verlag, 1998
7. R. L. Cook and K. E. Torrance „A Reflectance Model for Computer Graphics“, 1981