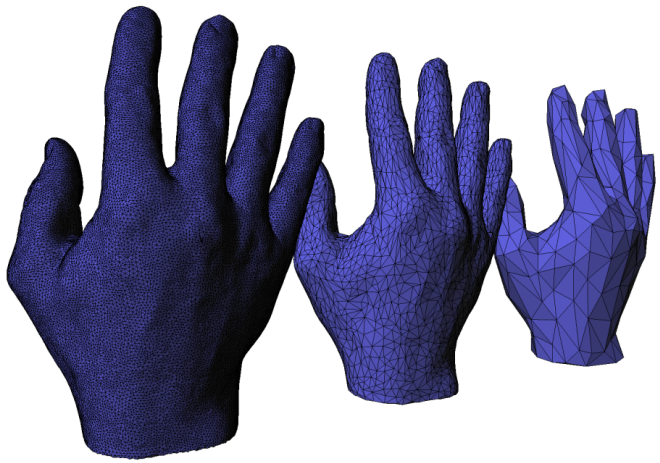
Freie Universität Berlin



Surface Simplification

Michael Thomas
Institut für Informatik
Freie Universität Berlin

Proseminar Computational Geometry 2010

[1] Titelbild: http://www.cgal.org/

Gliederung

Was wollen wir erreichen?

Was wollen wir erreichen?

- **ganz einfach:** Oberflächen vereinfachen!

Was wollen wir erreichen?

- **ganz einfach:** Oberflächen vereinfachen!
- wir haben sehr komplexe Polygonnetze

Was wollen wir erreichen?

- **ganz einfach:** Oberflächen vereinfachen!
- wir haben sehr komplexe Polygonnetze
- ...und wollen daraus einfachere Polygonnetze erzeugen

Etwas genauer

Etwas genauer

- Oberflächen werden durch triangulierte Polygonnetze dargestellt

Etwas genauer

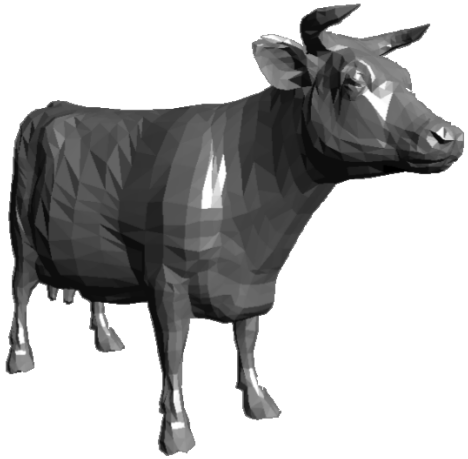
- Oberflächen werden durch triangulierte Polygonnetze dargestellt
- Sie bestehen aus vielen Vertices, Kanten und Dreiecken, sogenannten Facetten

Etwas genauer

- Oberflächen werden durch triangulierte Polygonnetze dargestellt
- Sie bestehen aus vielen Vertices, Kanten und Dreiecken, sogenannten Facetten
- Eine Surface Simplification reduziert die Anzahl dieser Facetten

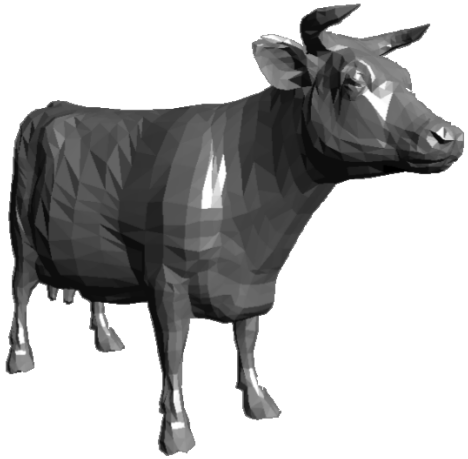
Ein Beispiel

Ein Beispiel

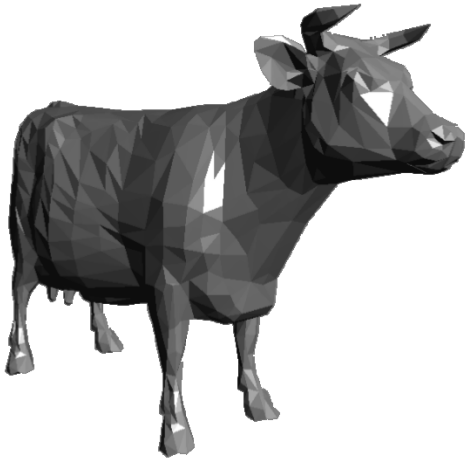


- Original

Ein Beispiel

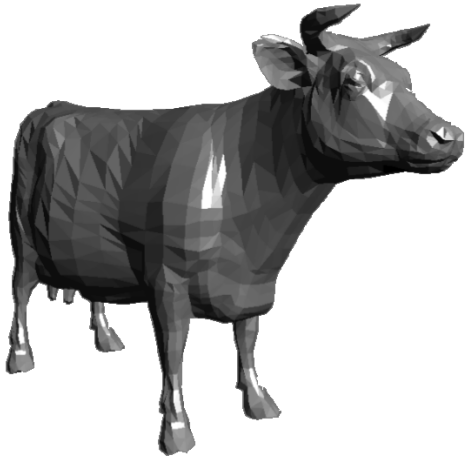


- Original

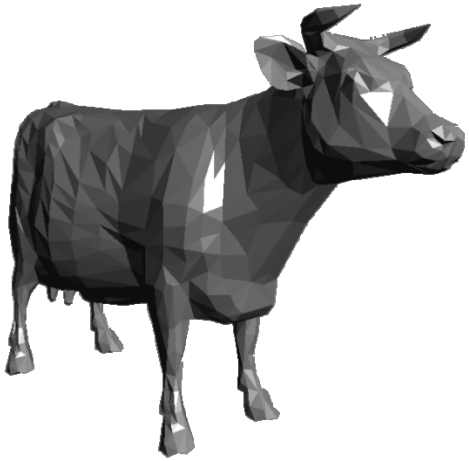


- 2500 Facetten

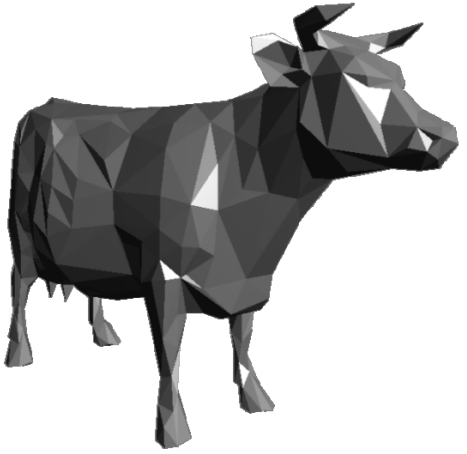
Ein Beispiel



- Original

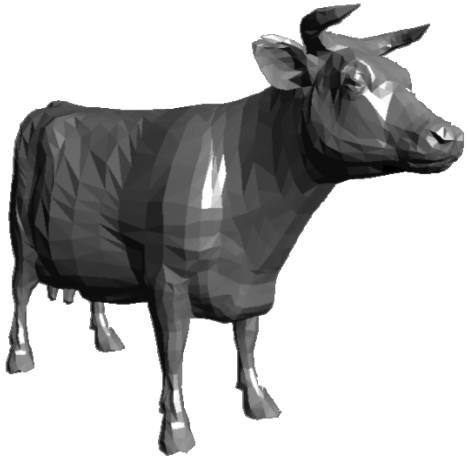


- 2500 Facetten

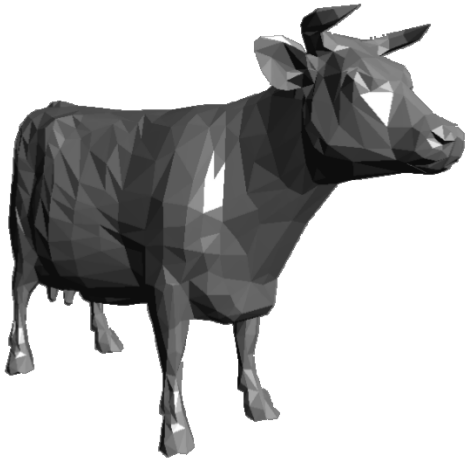


- 1000 Facetten

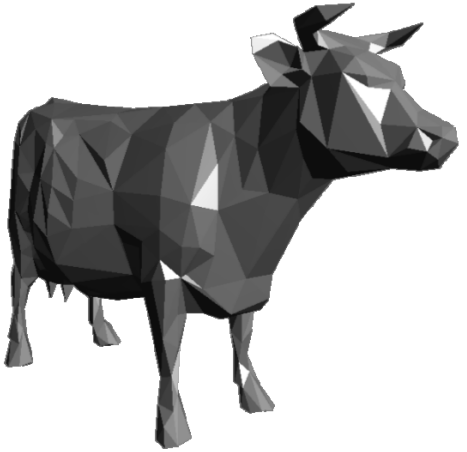
Ein Beispiel



- Original



- 2500 Facetten



- 1000 Facetten

- **Quelle:**
  `http://www.gustavgahm.com/wp-content/upload/moa/Gustav_Gahm_Mesh_Decimation.pdf`

Motivation

Warum sollen überhaupt Oberflächen vereinfacht werden?

Motivation

Warum sollen überhaupt Oberflächen vereinfacht werden?

- Erzeugung von Polygonnetzen oft nur in hohen Auflösungen möglich

Motivation

Warum sollen überhaupt Oberflächen vereinfacht werden?

- Erzeugung von Polygonnetzen oft nur in hohen Auflösungen möglich
- hohe Auflösung sieht auch besser aus!

Motivation

Warum sollen überhaupt Oberflächen vereinfacht werden?

- Erzeugung von Polygonnetzen oft nur in hohen Auflösungen möglich
- hohe Auflösung sieht auch besser aus!
- aber: Laufzeit der Algorithmen auf Polygonnetzen ist abhängig von deren Komplexität

Motivation

Warum sollen überhaupt Oberflächen vereinfacht werden?

- Erzeugung von Polygonnetzen oft nur in hohen Auflösungen möglich
- hohe Auflösung sieht auch besser aus!
- aber: Laufzeit der Algorithmen auf Polygonnetzen ist abhängig von deren Komplexität
- d.h. je kleiner die Netze, desto schneller die Verarbeitung

Motivation

Warum sollen überhaupt Oberflächen vereinfacht werden?

- Erzeugung von Polygonnetzen oft nur in hohen Auflösungen möglich
- hohe Auflösung sieht auch besser aus!
- aber: Laufzeit der Algorithmen auf Polygonnetzen ist abhängig von deren Komplexität
- d.h. je kleiner die Netze, desto schneller die Verarbeitung
- ...und benötigen weniger Speicherplatz!

Motivation

Warum sollen überhaupt Oberflächen vereinfacht werden?

- Erzeugung von Polygonnetzen oft nur in hohen Auflösungen möglich
- hohe Auflösung sieht auch besser aus!
- aber: Laufzeit der Algorithmen auf Polygonnetzen ist abhängig von deren Komplexität
- d.h. je kleiner die Netze, desto schneller die Verarbeitung
- ...und benötigen weniger Speicherplatz!

Also:

Wir brauchen ein Kompromiss zwischen Auflösung der Oberflächen und Geschwindigkeit in der Berechnung

Szenario

Computerspiel

- Rendern von Objekten aus Millionen von Dreiecken ist nicht möglich in Echtzeit

Szenario

Computerspiel

- Rendern von Objekten aus Millionen von Dreiecken ist nicht möglich in Echtzeit
- Man will aber trotzdem eine möglichst detaillierte Szene

Szenario

Computerspiel

- Rendern von Objekten aus Millionen von Dreiecken ist nicht möglich in Echtzeit
- Man will aber trotzdem eine möglichst detaillierte Szene
- **Beobachtung:** Detailgrad der Objekte ist auch Abhängig von der Größe ihrer Darstellung!

Szenario

Computerspiel

- Rendern von Objekten aus Millionen von Dreiecken ist nicht möglich in Echtzeit
- Man will aber trotzdem eine möglichst detaillierte Szene
- **Beobachtung:** Detailgrad der Objekte ist auch Abhängig von der Größe ihrer Darstellung!
- **Lösung:**

Szenario

Computerspiel

- Rendern von Objekten aus Millionen von Dreiecken ist nicht möglich in Echtzeit
- Man will aber trotzdem eine möglichst detaillierte Szene
- **Beobachtung:** Detailgrad der Objekte ist auch Abhängig von der Größe ihrer Darstellung!
- **Lösung:**
  - Für Objekte die näher an der Kamara sind werden Modelle mit höheren Auflösungen benutzt

Szenario

Computerspiel

- Rendern von Objekten aus Millionen von Dreiecken ist nicht möglich in Echtzeit
- Man will aber trotzdem eine möglichst detaillierte Szene
- **Beobachtung:** Detailgrad der Objekte ist auch Abhängig von der Größe ihrer Darstellung!
- **Lösung:**
  - Für Objekte die näher an der Kamara sind werden Modelle mit höheren Auflösungen benutzt
  - für entfernte Objekte entsprechend Modelle mit kleinerer Auflösung

Szenario

Computerspiel

- Rendern von Objekten aus Millionen von Dreiecken ist nicht möglich in Echtzeit
- Man will aber trotzdem eine möglichst detaillierte Szene
- **Beobachtung:** Detailgrad der Objekte ist auch Abhängig von der Größe ihrer Darstellung!
- **Lösung:**
  - Für Objekte die näher an der Kamara sind werden Modelle mit höheren Auflösungen benutzt
  - für entfernte Objekte entsprechend Modelle mit kleinerer Auflösung

Fazit:

Szenario

Computerspiel

- Rendern von Objekten aus Millionen von Dreiecken ist nicht möglich in Echtzeit
- Man will aber trotzdem eine möglichst detaillierte Szene
- **Beobachtung:** Detailgrad der Objekte ist auch Abhängig von der Größe ihrer Darstellung!
- **Lösung:**
  - Für Objekte die näher an der Kamara sind werden Modelle mit höheren Auflösungen benutzt
  - für entfernte Objekte entsprechend Modelle mit kleinerer Auflösung

Fazit:

- Surface Simplification macht Anwendung in Echtzeit oft erst möglich!

Szenario

Computerspiel

- Rendern von Objekten aus Millionen von Dreiecken ist nicht möglich in Echtzeit
- Man will aber trotzdem eine möglichst detaillierte Szene
- **Beobachtung:** Detailgrad der Objekte ist auch Abhängig von der Größe ihrer Darstellung!
- **Lösung:**
  - Für Objekte die näher an der Kamara sind werden Modelle mit höheren Auflösungen benutzt
  - für entfernte Objekte entsprechend Modelle mit kleinerer Auflösung

Fazit:

- Surface Simplification macht Anwendung in Echtzeit oft erst möglich!
- Die Veränderung der Auflösung soll möglichst kontinuierlich erfolgen

Zusammenfassung

- Detailgrad von Oberflächen reduzieren um Anwendungen zu beschleunigen
- dabei soll die geometrische Struktur der Modelle so gut wie möglich erhalten bleiben
- Kontrolle über den Grad der Vereinfachung
- kontinuierliche Veränderung der Auflösung (multiresolutional modelling)
- Effizienz

Methoden

Vertex Decimation

Vertex Decimation

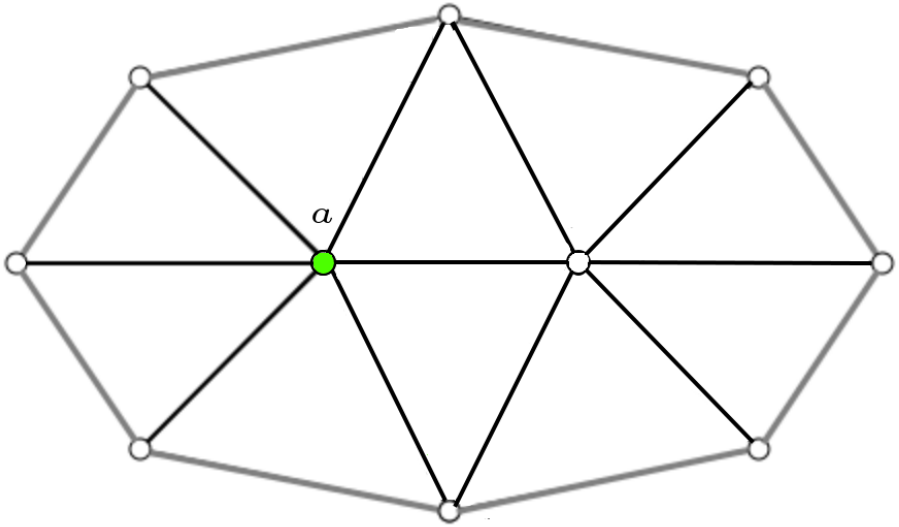
- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke

Vertex Decimation

- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke
- Triangulierung des enstandenen Lochs

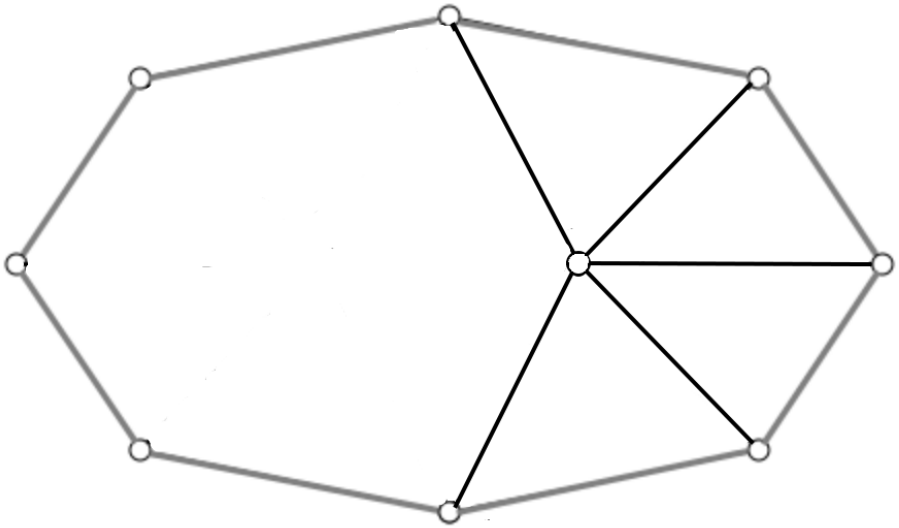
Vertex Decimation

- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke
- Triangulierung des enstandenen Lochs



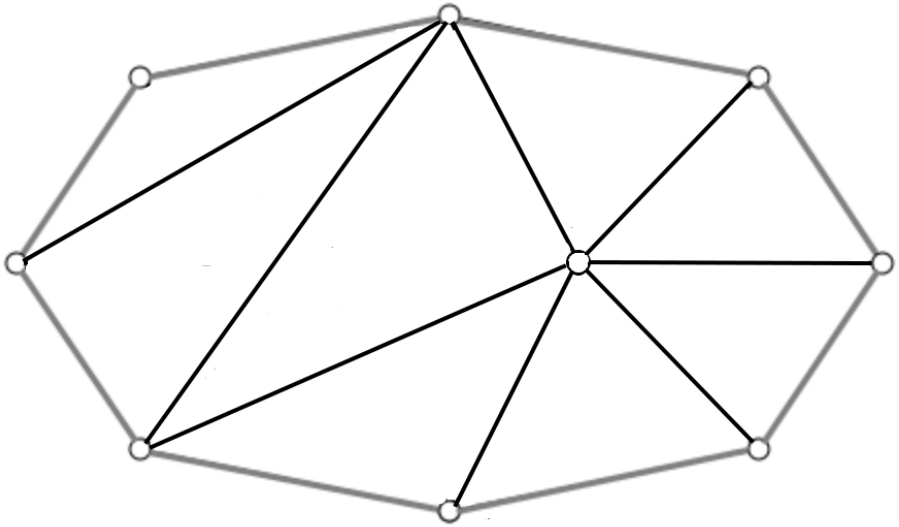
Vertex Decimation

- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke
- Triangulierung des enstandenen Lochs



Vertex Decimation

- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke
- Triangulierung des enstandenen Lochs



Methoden

Vertex Decimation

- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke
- Triangulierung des enstandenen Lochs

Edge Contraction

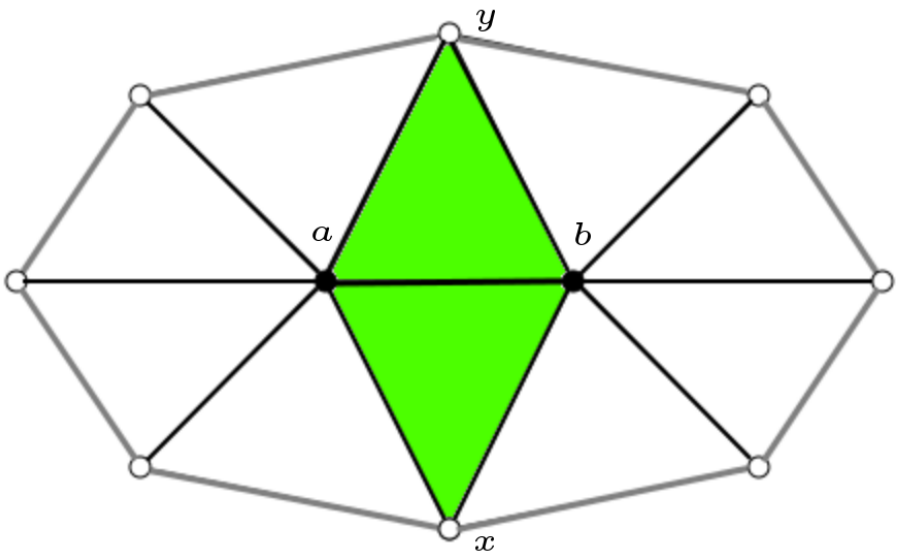
Methoden

Vertex Decimation

- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke
- Triangulierung des enstandenen Lochs

Edge Contraction

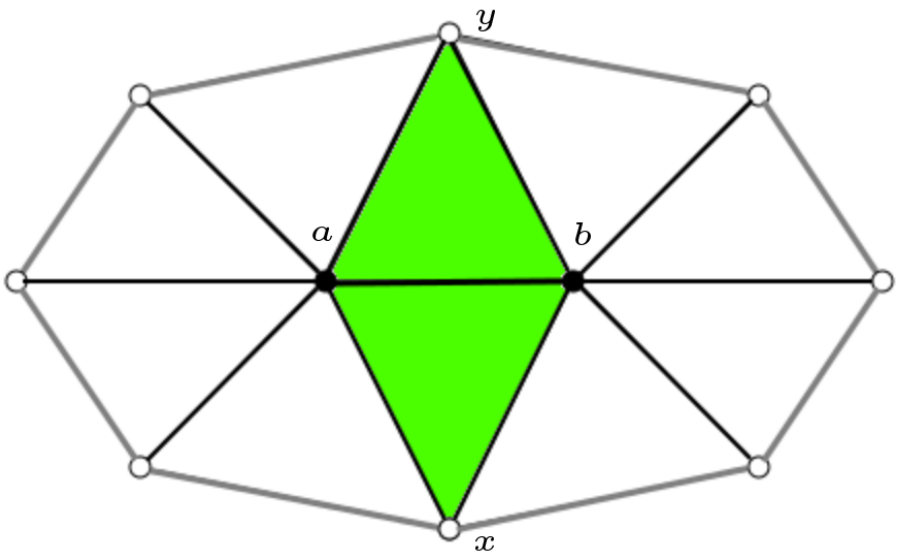
- Zusammenziehen einer Kante zu einem Vertex



Methoden

Vertex Decimation

- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke
- Triangulierung des enstandenen Lochs



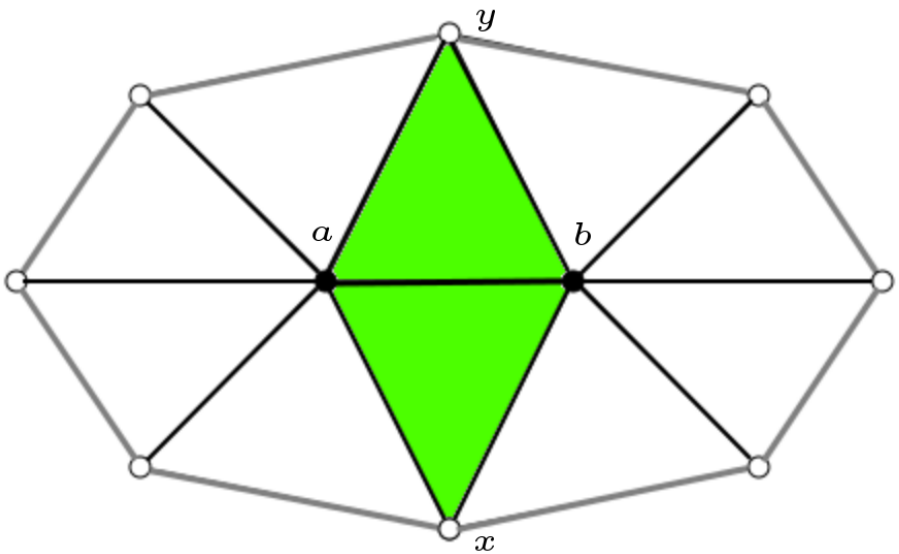
Edge Contraction

- Zusammenziehen einer Kante zu einem Vertex
- dabei verschwinden zwei Dreiecke

Methoden

Vertex Decimation

- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke
- Triangulierung des enstandenen Lochs



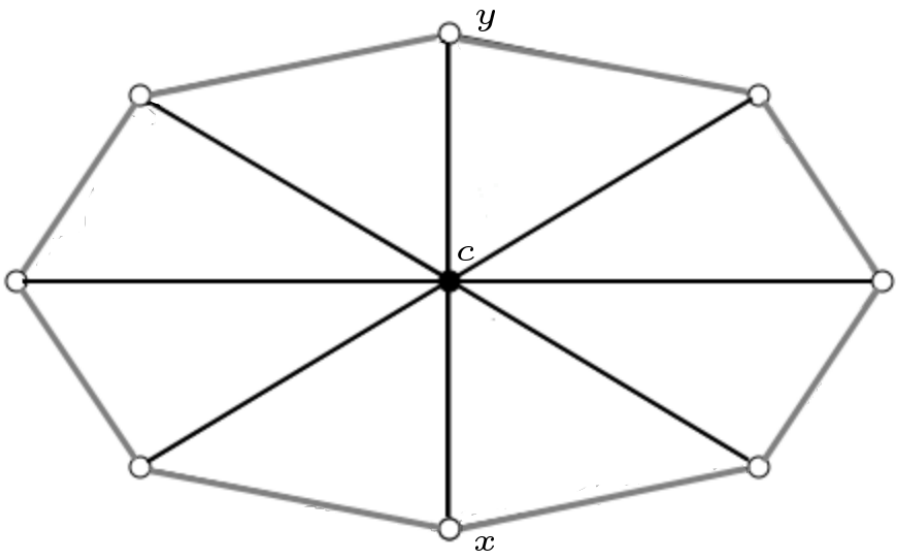
Edge Contraction

- Zusammenziehen einer Kante zu einem Vertex
- dabei verschwinden zwei Dreiecke

Methoden

Vertex Decimation

- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke
- Triangulierung des enstandenen Lochs



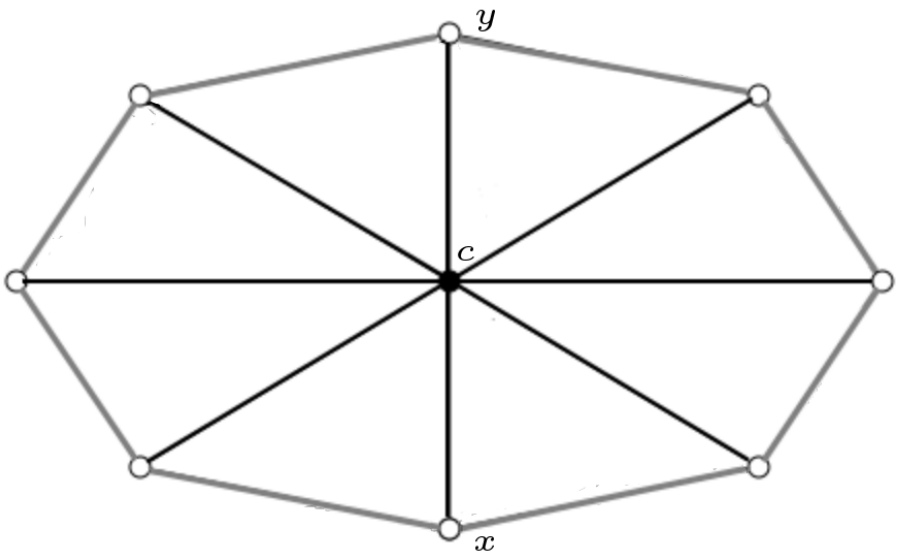
Edge Contraction

- Zusammenziehen einer Kante zu einem Vertex
- dabei verschwinden zwei Dreiecke

Methoden

Vertex Decimation

- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke
- Triangulierung des enstandenen Lochs



Edge Contraction

- Zusammenziehen einer Kante zu einem Vertex
- dabei verschwinden zwei Dreiecke

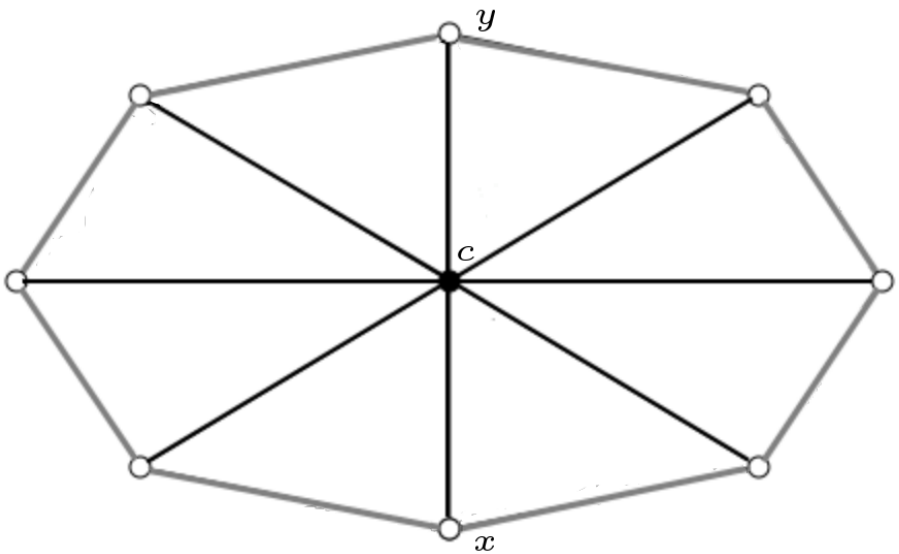
Beobachtung

- beide Algorithmen arbeiten iterativ
- d.h. in jedem Schritt wird eine Dezimierungsoperation ausgeführt

Methoden

Vertex Decimation

- Entfernung von einzelnen Vertices, und die mit ihm verbundenen Kanten und Dreiecke
- Triangulierung des enstandenen Lochs



Edge Contraction

- Zusammenziehen einer Kante zu einem Vertex
- dabei verschwinden zwei Dreiecke

Beobachtung

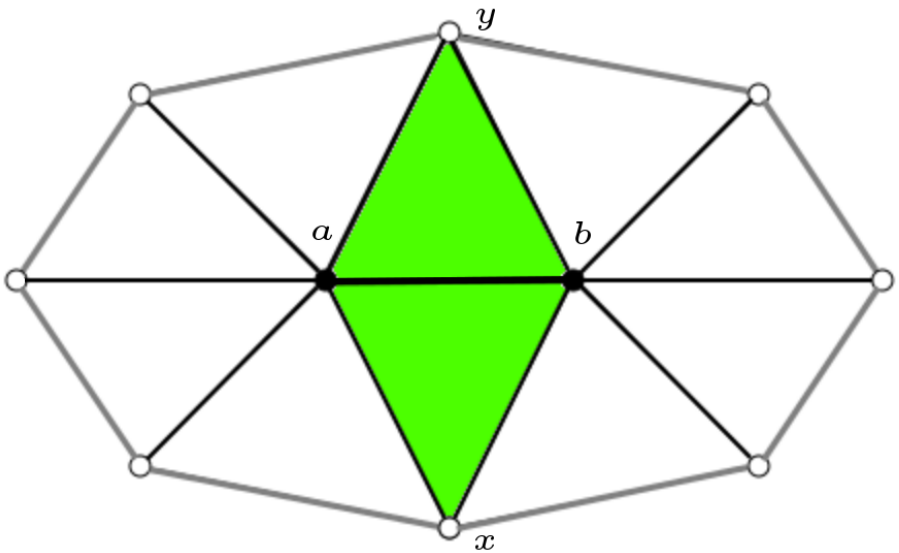
- beide Algorithmen arbeiten iterativ
- d.h. in jedem Schritt wird eine Dezimierungsoperation ausgeführt

Was passiert genau?

Formale Beschreibung

Was passiert genau?

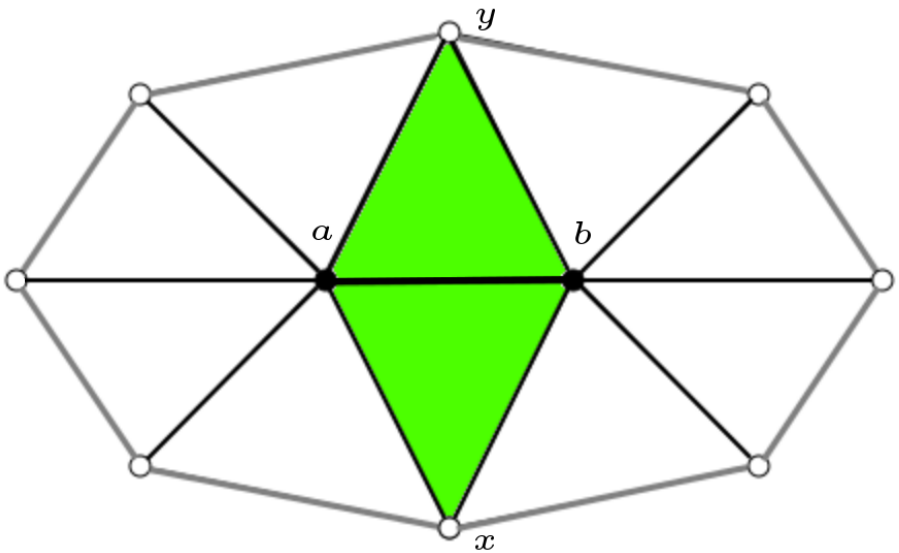
- Die Kante ab und alle angrenzenden Kanten und Dreiecke werden entfernt



Formale Beschreibung

Was passiert genau?

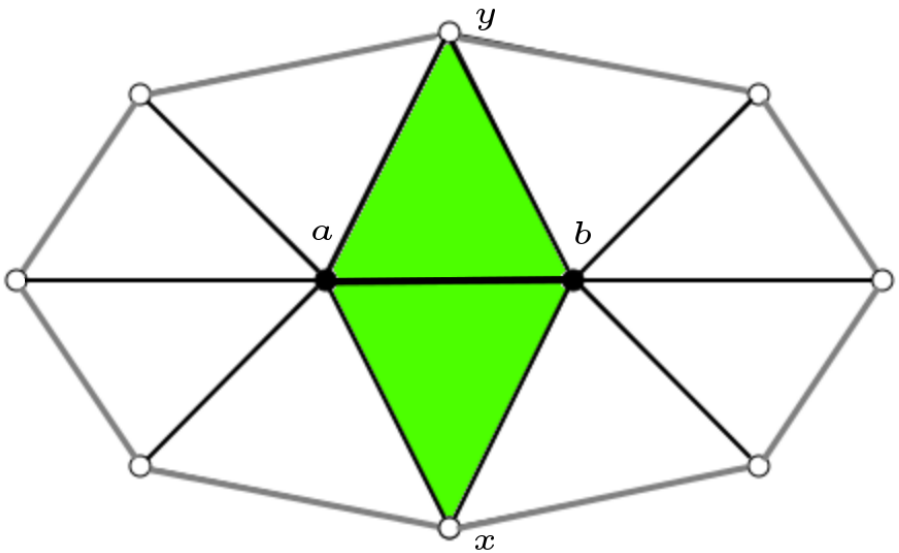
- Die Kante ab und alle angrenzenden Kanten und Dreiecke werden entfernt
- Es wird ein neuer Vertex c eingefügt



Formale Beschreibung

Was passiert genau?

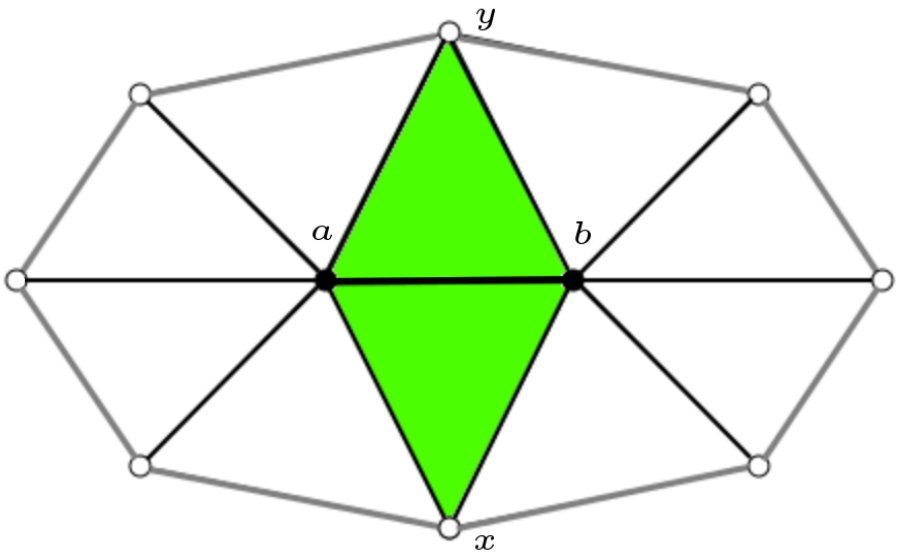
- Die Kante ab und alle angrenzenden Kanten und Dreiecke werden entfernt
- Es wird ein neuer Vertex c eingefügt
- Alle Vertices im Link von $\overline{ab}$ werden mit c verbunden



Formale Beschreibung

Was passiert genau?

- Die Kante ab und alle angrenzenden Kanten und Dreiecke werden entfernt
- Es wird ein neuer Vertex c eingefügt
- Alle Vertices im Link von $\overline{ab}$ werden mit c verbunden



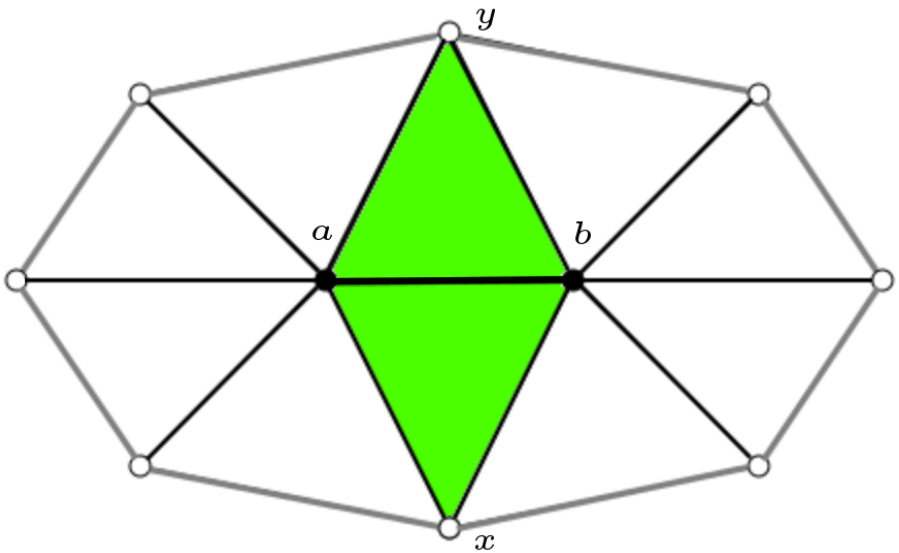
Lemma

Formale Definition: $L = K - St\,\overline{ab} \cup c \cdot Lk\,\overline{ab}$
wobei $c \cdot T = \{conv(\{c\} \cup \tau) \mid \tau \in T\}$ *die Kegel der Menge T darstellt.*

Formale Beschreibung

Was passiert genau?

- Die Kante ab und alle angrenzenden Kanten und Dreiecke werden entfernt
- Es wird ein neuer Vertex c eingefügt
- Alle Vertices im Link von $\overline{ab}$ werden mit c verbunden



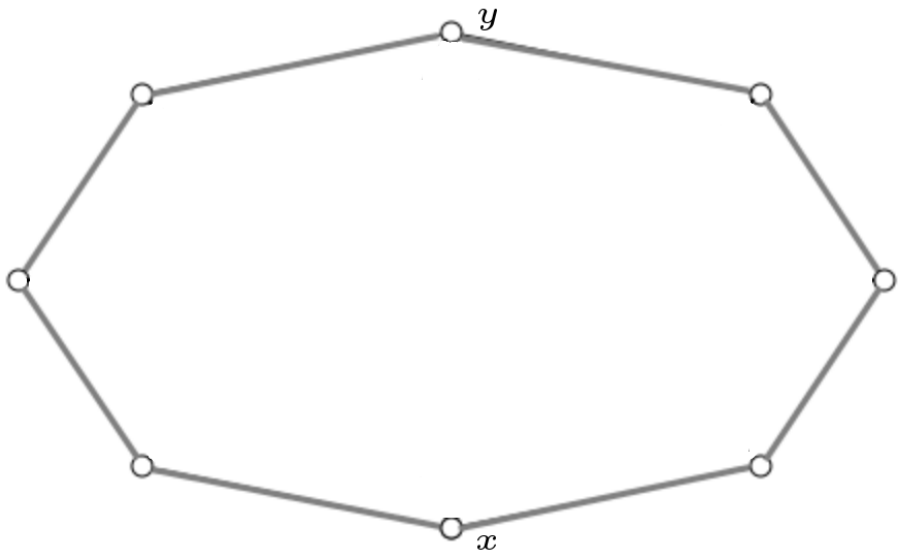
Lemma

Formale Definition: $L = K - St\,\overline{ab} \cup c \cdot Lk\,\overline{ab}$
wobei $c \cdot T = \{conv(\{c\} \cup \tau) \mid \tau \in T\}$ *die Kegel der Menge T darstellt.*

Formale Beschreibung

Was passiert genau?

- Die Kante ab und alle angrenzenden Kanten und Dreiecke werden entfernt
- Es wird ein neuer Vertex c eingefügt
- Alle Vertices im Link von $\overline{ab}$ werden mit c verbunden



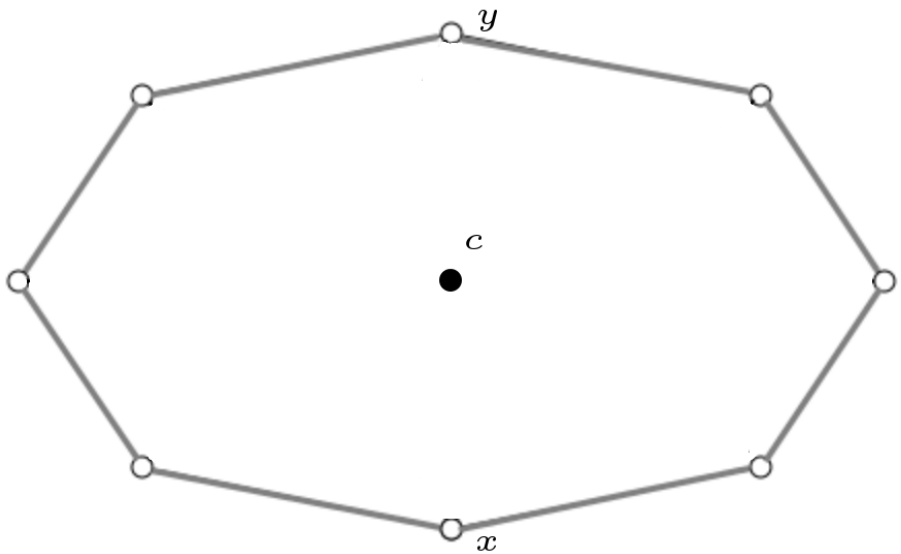
Lemma

Formale Definition: $L = K - St\,\overline{ab} \cup c \cdot Lk\,\overline{ab}$
wobei $c \cdot T = \{conv(\{c\} \cup \tau) \mid \tau \in T\}$ *die Kegel der Menge T darstellt.*

Formale Beschreibung

Was passiert genau?

- Die Kante ab und alle angrenzenden Kanten und Dreiecke werden entfernt
- Es wird ein neuer Vertex c eingefügt
- Alle Vertices im Link von $\overline{ab}$ werden mit c verbunden



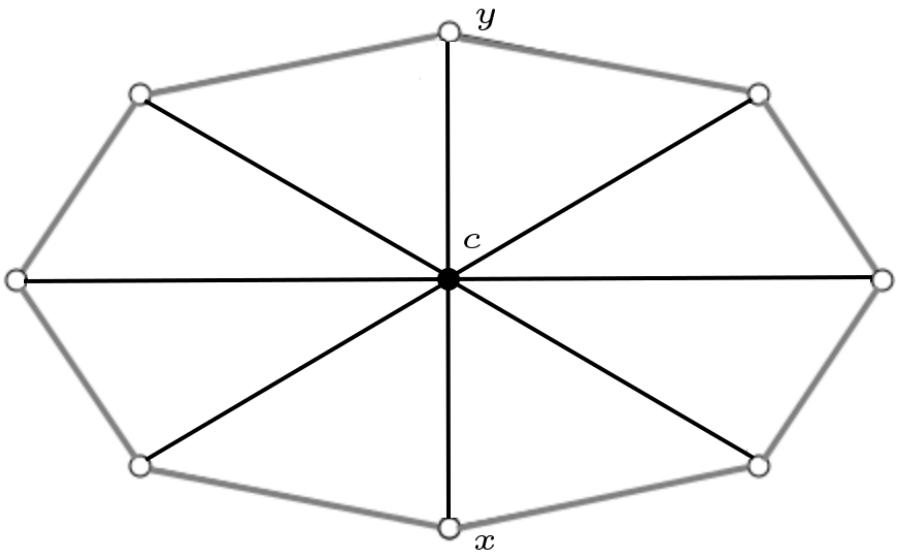
Lemma

Formale Definition: $L = K - St\,\overline{ab} \cup c \cdot Lk\,\overline{ab}$
wobei $c \cdot T = \{conv(\{c\} \cup \tau) \mid \tau \in T\}$ *die Kegel der Menge T darstellt.*

Formale Beschreibung

Was passiert genau?

- Die Kante ab und alle angrenzenden Kanten und Dreiecke werden entfernt
- Es wird ein neuer Vertex c eingefügt
- Alle Vertices im Link von $\overline{ab}$ werden mit c verbunden



Lemma

Formale Definition: $L = K - St\,\overline{ab} \cup c \cdot Lk\,\overline{ab}$
wobei $c \cdot T = \{conv(\{c\} \cup \tau) \mid \tau \in T\}$ *die Kegel der Menge T darstellt.*

Freie Universität Berlin

mehrere Auflösungen?

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c

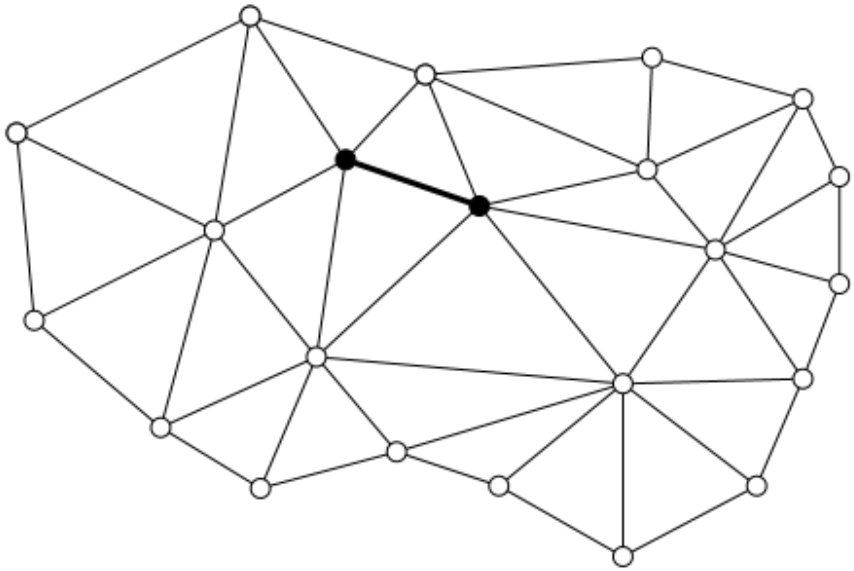
Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

Freie Universität Berlin

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c



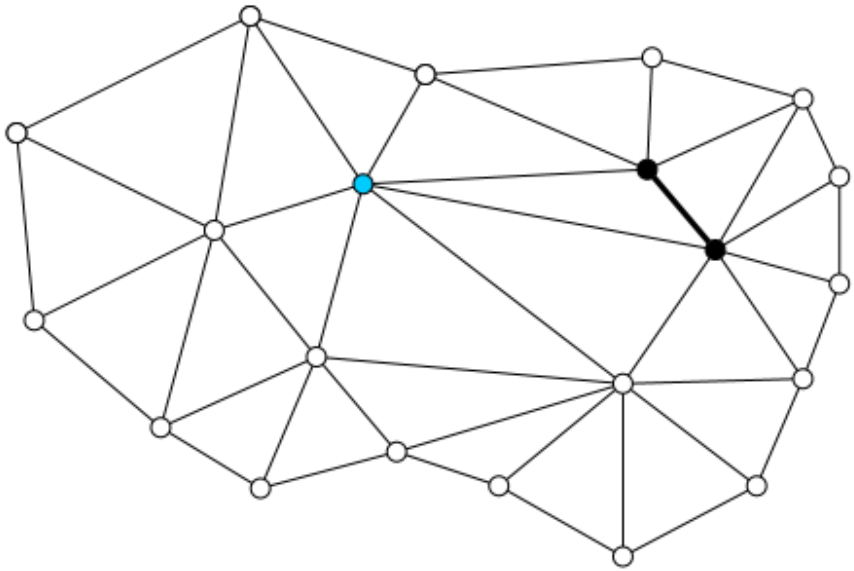
Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

Freie Universität Berlin

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c



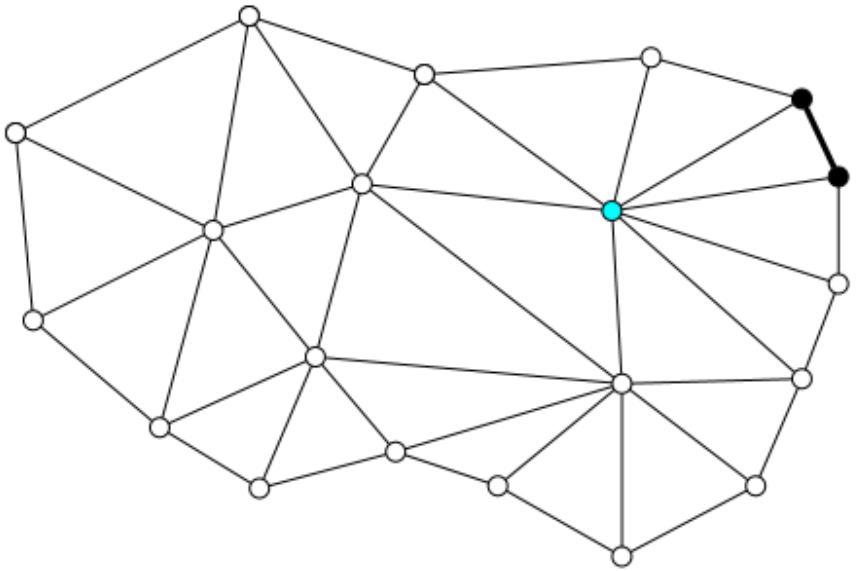
Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

Freie Universität Berlin

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c



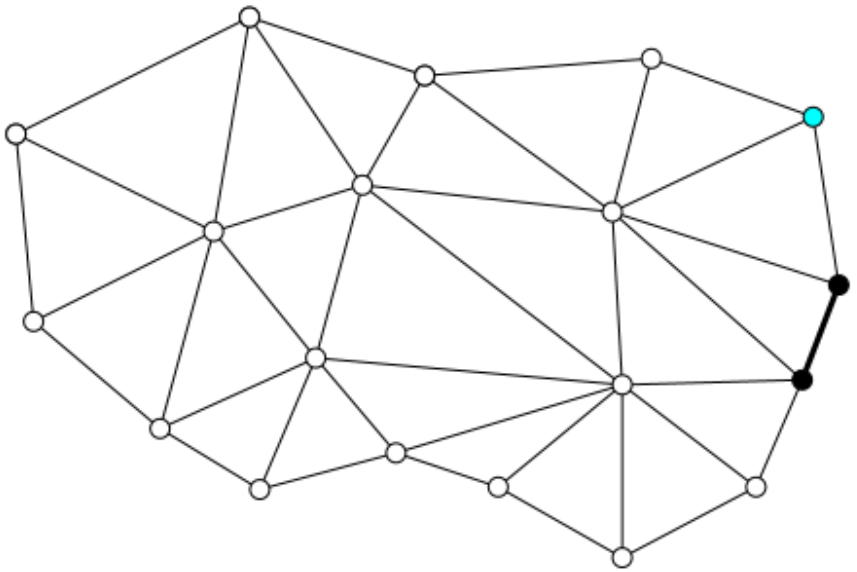
Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

Freie Universität Berlin

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c

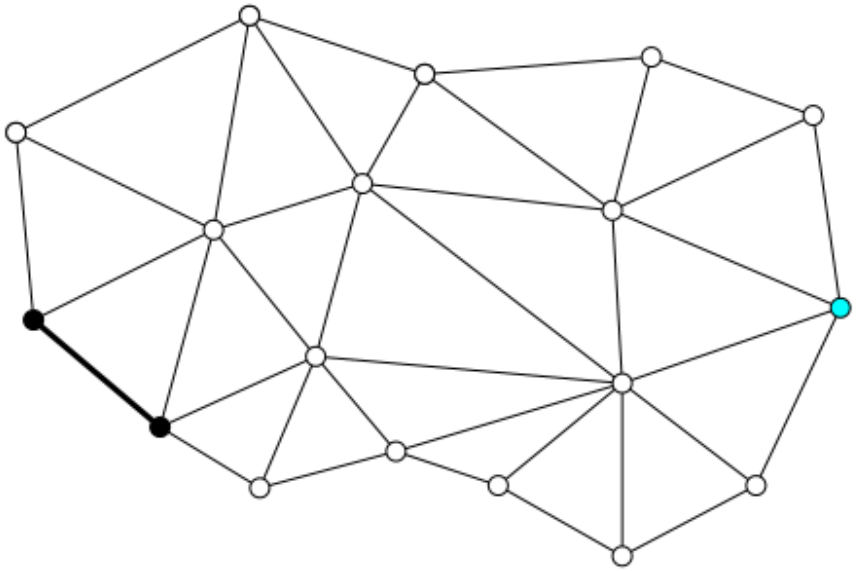


Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c

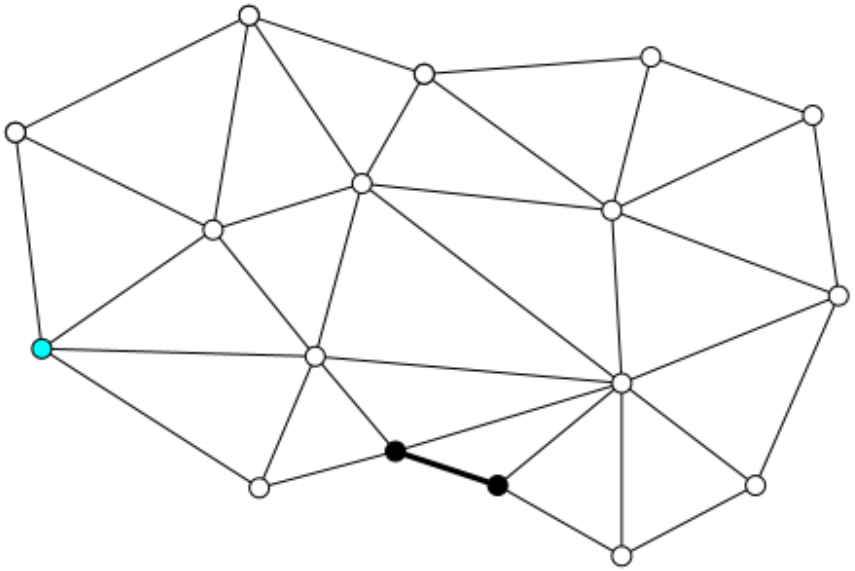


Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c



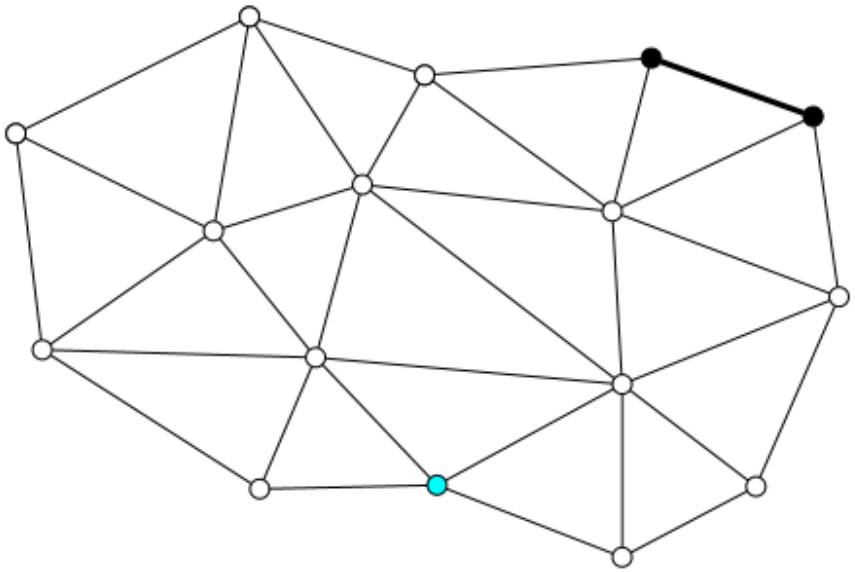
Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

Freie Universität Berlin

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c



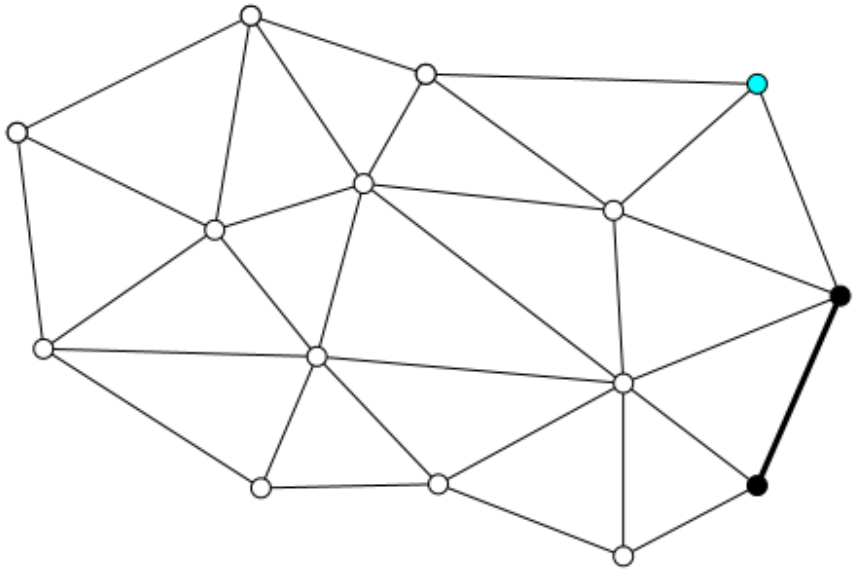
Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

Freie Universität Berlin

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c

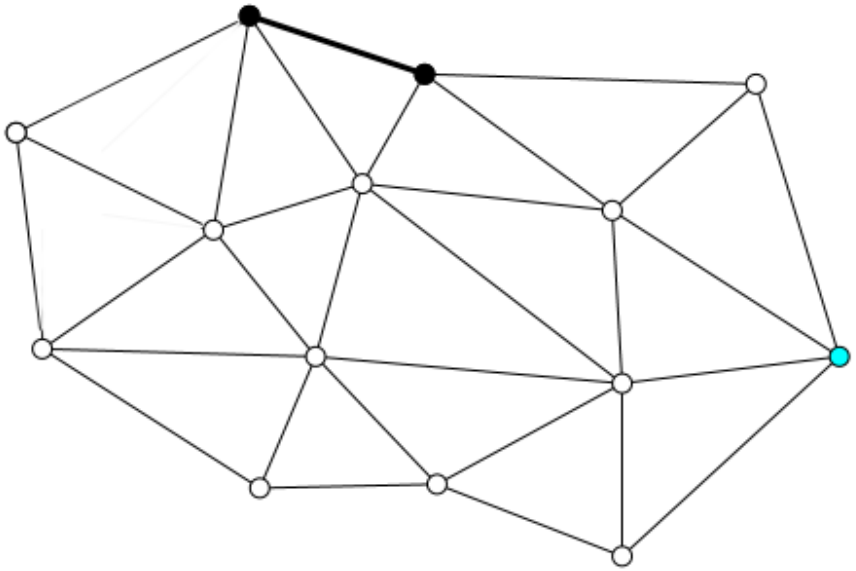


Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c

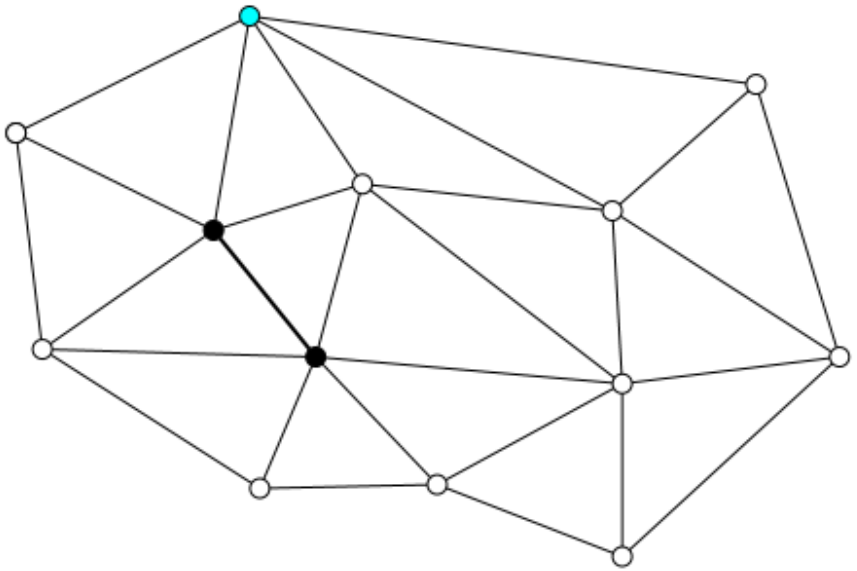


Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c

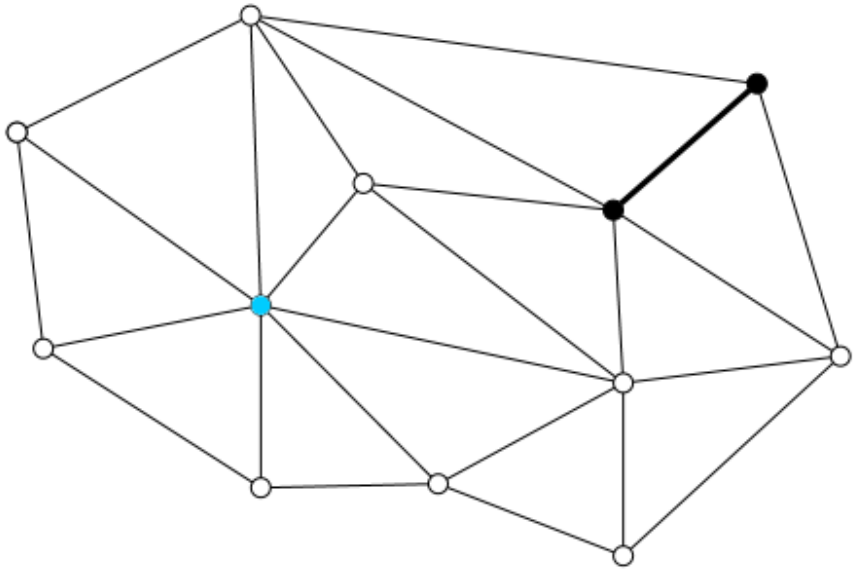


Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c



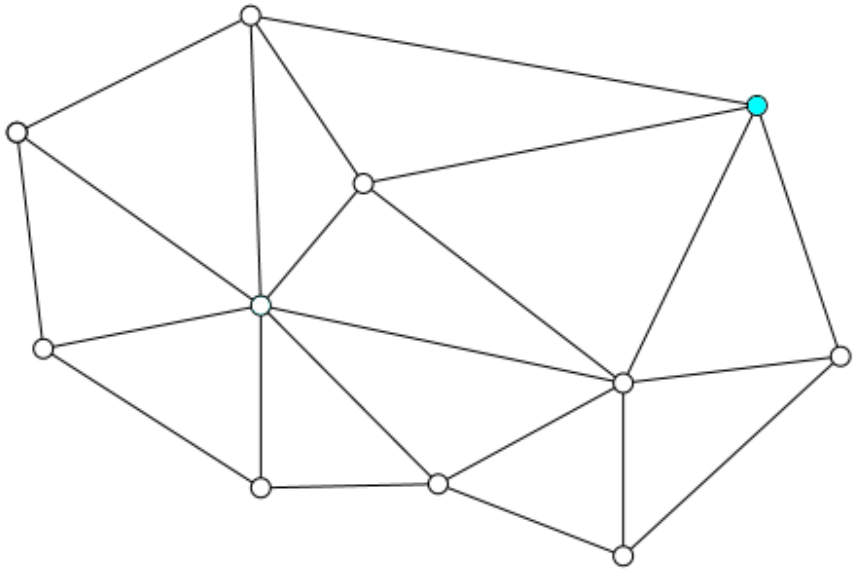
Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

Freie Universität Berlin

mehrere Auflösungen?

- der Algorithmus erzeugt eine Folge von Kontraktionsoperationen
- eine einzelne Kontraktion ist wohldefiniert durch eine Abbildung der Vertices ab auf den neuen Vertex c



Multiresolutional Modelling

- man speichert die Folge aller Operationen
- ...und können damit das modell in jeder beliebiger Auflösung rekunstruieren
- und das in linearer Zeit!

Problem

Fehlerkontrolle

Problem

- nicht jeder Kontraktion erhält die Form gleichermaßen

Fehlerkontrolle

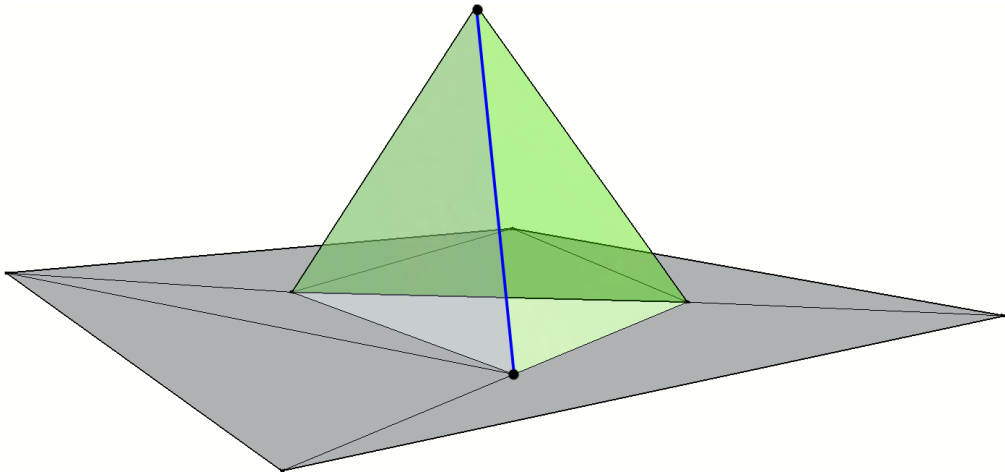
Problem

- nicht jeder Kontraktion erhält die Form gleichermaßen
- wo soll der neuer Vertex eingefügt werden?

Fehlerkontrolle

Problem

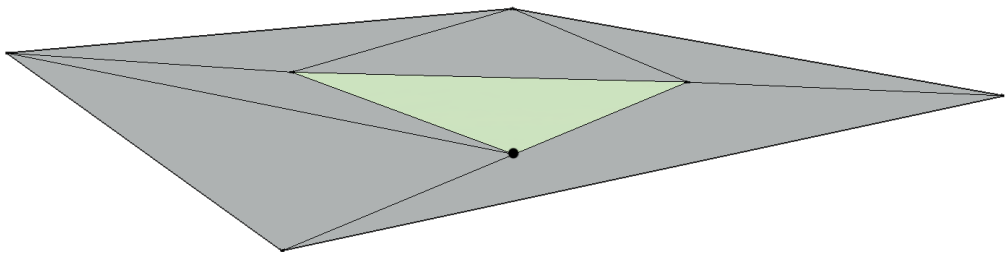
- nicht jeder Kontraktion erhält die Form gleichermaßen
- wo soll der neuer Vertex eingefügt werden?



Fehlerkontrolle

Problem

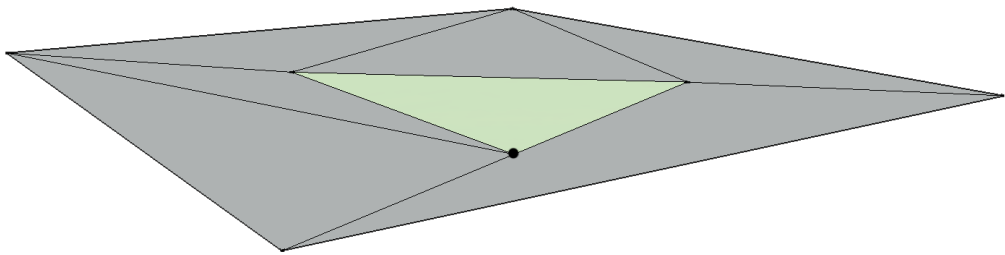
- nicht jeder Kontraktion erhält die Form gleichermaßen
- wo soll der neuer Vertex eingefügt werden?



Fehlerkontrolle

Problem

- nicht jeder Kontraktion erhält die Form gleichermaßen
- wo soll der neuer Vertex eingefügt werden?



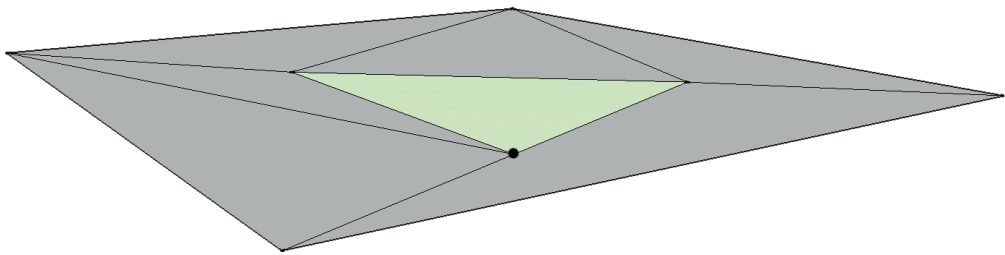
Fazit

- wir brauchen eine Fehlerkontrolle

Fehlerkontrolle

Problem

- nicht jeder Kontraktion erhält die Form gleichermaßen
- wo soll der neuer Vertex eingefügt werden?



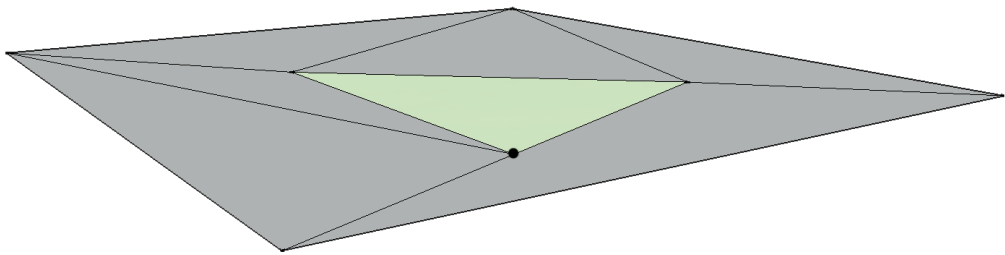
Fazit

- wir brauchen eine Fehlerkontrolle
- die uns die Abweichung zum Original angibt

Fehlerkontrolle

Problem

- nicht jeder Kontraktion erhält die Form gleichermaßen
- wo soll der neuer Vertex eingefügt werden?



Fazit

- wir brauchen eine Fehlerkontrolle
- die uns die Abweichung zum Original angibt
- der Fehler soll dabei möglichst klein sein

Fehlerkontrolle

- die Differenz zum Originalmodell auszurechnen ist sehr teuer

Fehlerkontrolle

- die Differenz zum Originalmodell auszurechnen ist sehr teuer
- deshalb: berechne Differenz zur letzten Iteration

Fehlerkontrolle

- die Differenz zum Originalmodell auszurechnen ist sehr teuer
- deshalb: berechne Differenz zur letzten Iteration
- es reicht sich dabei die lokale Umgebung anzusehen

Fehlerkontrolle

- die Differenz zum Originalmodell auszurechnen ist sehr teuer
- deshalb: berechne Differenz zur letzten Iteration
- es reicht sich dabei die lokale Umgebung anzusehen

Zusammenfassung

Wir brauchen...

Fehlerkontrolle

- die Differenz zum Originalmodell auszurechnen ist sehr teuer
- deshalb: berechne Differenz zur letzten Iteration
- es reicht sich dabei die lokale Umgebung anzusehen

Zusammenfassung

Wir brauchen...

- eine Kostenfunktion, welche den Fehler für eine Kontraktion berechnet

Fehlerkontrolle

- die Differenz zum Originalmodell auszurechnen ist sehr teuer
- deshalb: berechne Differenz zur letzten Iteration
- es reicht sich dabei die lokale Umgebung anzusehen

Zusammenfassung

Wir brauchen...

- eine Kostenfunktion, welche den Fehler für eine Kontraktion berechnet
- der neue Vertex wird wird so gewählt, dass die Kosten minimal sind

Quadratische Distanzen

Idee:

- für jeden Vertex definieren wir eine Abstandsfunktion zu seinen Dreiecksebenen

Wozu das Ganze?

Quadratische Distanzen

Idee:

- für jeden Vertex definieren wir eine Abstandsfunktion zu seinen Dreiecksebenen
- genauer: die Summe der quadratischen Distanzen

$$E(x) = \sum_{H \in planes(v)} d(x, H)^2$$

Wozu das Ganze?

Quadratische Distanzen

Idee:

- für jeden Vertex definieren wir eine Abstandsfunktion zu seinen Dreiecksebenen
- genauer: die Summe der quadratischen Distanzen

$$E(x) = \sum_{H \in planes(v)} d(x, H)^2$$

Wozu das Ganze?

- Diese Abstandsfunktionen lassen sich durch eine 4x4 Matrix beschreiben

Quadratische Distanzen

Idee:

- für jeden Vertex definieren wir eine Abstandsfunktion zu seinen Dreiecksebenen
- genauer: die Summe der quadratischen Distanzen

$$E(x) = \sum_{H \in planes(v)} d(x, H)^2$$

Wozu das Ganze?

- Diese Abstandsfunktionen lassen sich durch eine 4x4 Matrix beschreiben
- Die Kostenfunktion für eine Kontraktion entsteht durch Addition der Matrizen für beide Vertices

Quadratische Distanzen

Idee:

- für jeden Vertex definieren wir eine Abstandsfunktion zu seinen Dreiecksebenen
- genauer: die Summe der quadratischen Distanzen

$$E(x) = \sum_{H \in planes(v)} d(x, H)^2$$

Wozu das Ganze?

- Diese Abstandsfunktionen lassen sich durch eine 4x4 Matrix beschreiben
- Die Kostenfunktion für eine Kontraktion entsteht durch Addition der Matrizen für beide Vertices

Zusammenfassung

Quadratische Distanzen

Idee:

- für jeden Vertex definieren wir eine Abstandsfunktion zu seinen Dreiecksebenen
- genauer: die Summe der quadratischen Distanzen

$$E(x) = \sum_{H \in planes(v)} d(x, H)^2$$

Wozu das Ganze?

- Diese Abstandsfunktionen lassen sich durch eine 4x4 Matrix beschreiben
- Die Kostenfunktion für eine Kontraktion entsteht durch Addition der Matrizen für beide Vertices

Zusammenfassung

- Wir erhalten eine Funktion, welche die Summe der Distanzen von einem Vertex zu einer Menge von Ebenen berechnet

Quadratische Distanzen

Idee:

- für jeden Vertex definieren wir eine Abstandsfunktion zu seinen Dreiecksebenen
- genauer: die Summe der quadratischen Distanzen

$$E(x) = \sum_{H \in planes(v)} d(x, H)^2$$

Wozu das Ganze?

- Diese Abstandsfunktionen lassen sich durch eine 4x4 Matrix beschreiben
- Die Kostenfunktion für eine Kontraktion entsteht durch Addition der Matrizen für beide Vertices

Zusammenfassung

- Wir erhalten eine Funktion, welche die Summe der Distanzen von einem Vertex zu einer Menge von Ebenen berechnet
- die Kosten für eine Kontraktion ab ist die Summe der Distanzen zu den Ebenen von a und b

Quadratische Distanzen

Idee:

- für jeden Vertex definieren wir eine Abstandsfunktion zu seinen Dreiecksebenen
- genauer: die Summe der quadratischen Distanzen

$$E(x) = \sum_{H \in planes(v)} d(x, H)^2$$

Wozu das Ganze?

- Diese Abstandsfunktionen lassen sich durch eine 4x4 Matrix beschreiben
- Die Kostenfunktion für eine Kontraktion entsteht durch Addition der Matrizen für beide Vertices

Zusammenfassung

- Wir erhalten eine Funktion, welche die Summe der Distanzen von einem Vertex zu einer Menge von Ebenen berechnet
- die Kosten für eine Kontraktion ab ist die Summe der Distanzen zu den Ebenen von a und b
- dies ergibt eine (quadratische) Funktion!

Quadratische Distanzen

Idee:

- für jeden Vertex definieren wir eine Abstandsfunktion zu seinen Dreiecksebenen
- genauer: die Summe der quadratischen Distanzen

$$E(x) = \sum_{H \in planes(v)} d(x, H)^2$$

Wozu das Ganze?

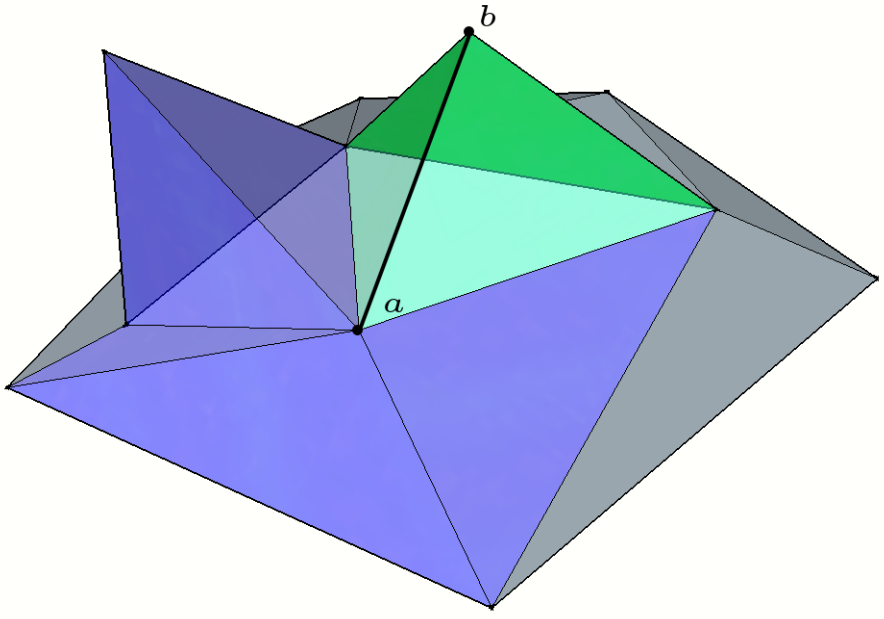
- Diese Abstandsfunktionen lassen sich durch eine 4x4 Matrix beschreiben
- Die Kostenfunktion für eine Kontraktion entsteht durch Addition der Matrizen für beide Vertices

Zusammenfassung

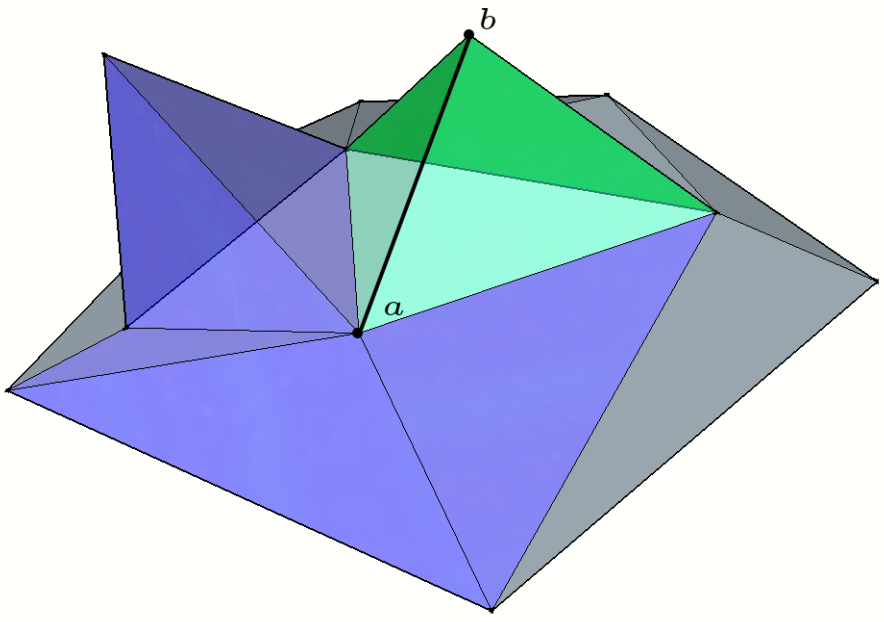
- Wir erhalten eine Funktion, welche die Summe der Distanzen von einem Vertex zu einer Menge von Ebenen berechnet
- die Kosten für eine Kontraktion ab ist die Summe der Distanzen zu den Ebenen von a und b
- dies ergibt eine (quadratische) Funktion!
- wir können diese minimieren und erhalten den optimalen Punkt c

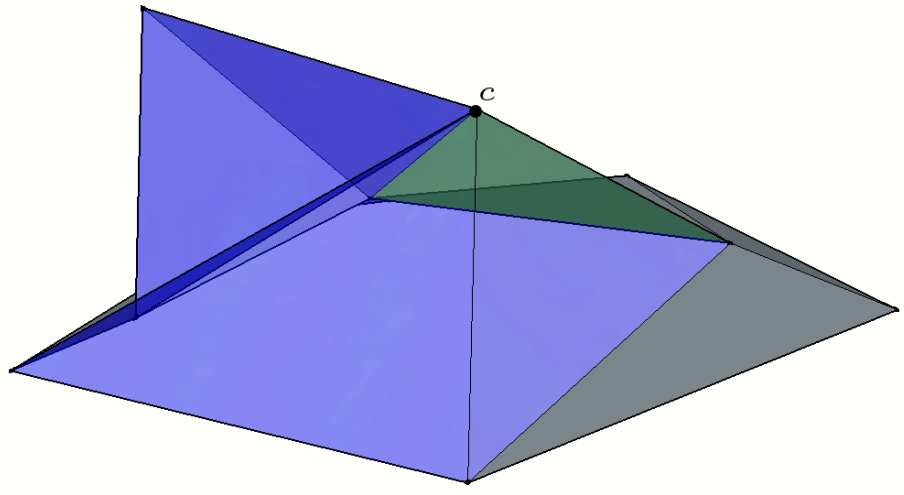
Ein kleines Beispiel

Ein kleines Beispiel



Ein kleines Beispiel





Ablauf

1. Berechne Abstandsfunktion für alle Vertices

Algorithmus

Ablauf

1. Berechne Abstandsfunktion für alle Vertices
2. Bestimme die Kostenfunktion für alle Kanten ab

Algorithmus

Ablauf

1. Berechne Abstandsfunktion für alle Vertices
2. Bestimme die Kostenfunktion für alle Kanten ab
3. Finde für jede Kante einen optimalen neuen Punkt c durch Minimierung der Kostenfunktion

Algorithmus

Ablauf

1. Berechne Abstandsfunktion für alle Vertices
2. Bestimme die Kostenfunktion für alle Kanten ab
3. Finde für jede Kante einen optimalen neuen Punkt c durch Minimierung der Kostenfunktion
4. Füge alle Kanten in einen Heap ein, geordnet nach den Kosten

Algorithmus

Ablauf

1. Berechne Abstandsfunktion für alle Vertices
2. Bestimme die Kostenfunktion für alle Kanten ab
3. Finde für jede Kante einen optimalen neuen Punkt c durch Minimierung der Kostenfunktion
4. Füge alle Kanten in einen Heap ein, geordnet nach den Kosten
5. Entnehme iterativ eine Kante aus dem Heap, und führe eine Kontraktion aus

Algorithmus

Ablauf

1. Berechne Abstandsfunktion für alle Vertices
2. Bestimme die Kostenfunktion für alle Kanten ab
3. Finde für jede Kante einen optimalen neuen Punkt c durch Minimierung der Kostenfunktion
4. Füge alle Kanten in einen Heap ein, geordnet nach den Kosten
5. Entnehme iterativ eine Kante aus dem Heap, und führe eine Kontraktion aus
6. Aktualisiere die Abstandsfunktionen für alle gänderten Vertices

Algorithmus

Ablauf

1. Berechne Abstandsfunktion für alle Vertices
2. Bestimme die Kostenfunktion für alle Kanten ab
3. Finde für jede Kante einen optimalen neuen Punkt c durch Minimierung der Kostenfunktion
4. Füge alle Kanten in einen Heap ein, geordnet nach den Kosten
5. Entnehme iterativ eine Kante aus dem Heap, und führe eine Kontraktion aus
6. Aktualisiere die Abstandsfunktionen für alle gänderten Vertices

- Schritt 5. und 6. werden so lange wiederholt, bis die gewünschte Qualität erreicht wurde

Worauf sollte man noch achten?

Topologieerhaltung

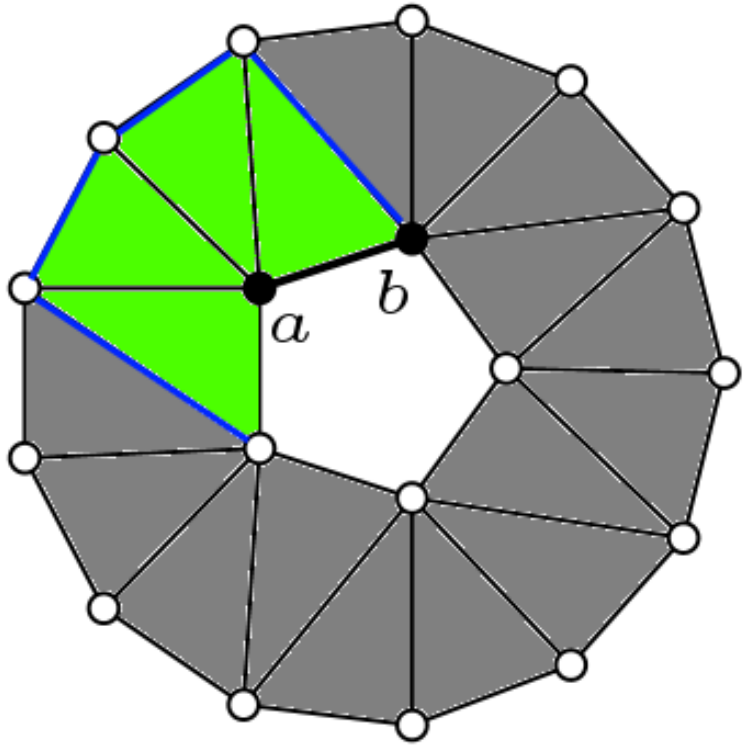
Worauf sollte man noch achten?

- Die Topologie könnte verändert werden
- Beispiel: Oberflächen mit Löchern
- das Loch könnte geschlossen werden!
- das gleiche gilt für einen Torus...

Topologieerhaltung

Worauf sollte man noch achten?

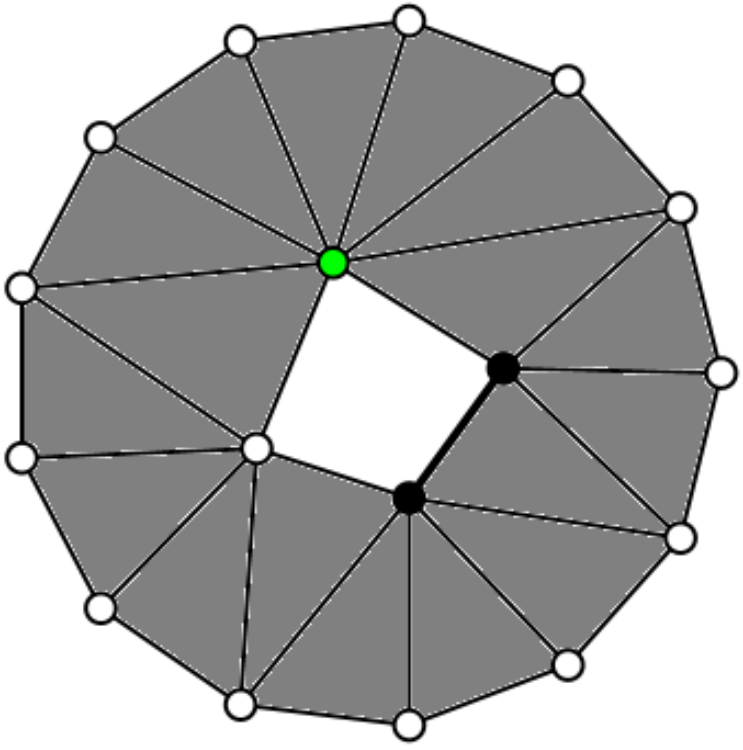
- Die Topologie könnte verändert werden
- Beispiel: Oberflächen mit Löchern
- das Loch könnte geschlossen werden!
- das gleiche gilt für einen Torus...



Topologieerhaltung

Worauf sollte man noch achten?

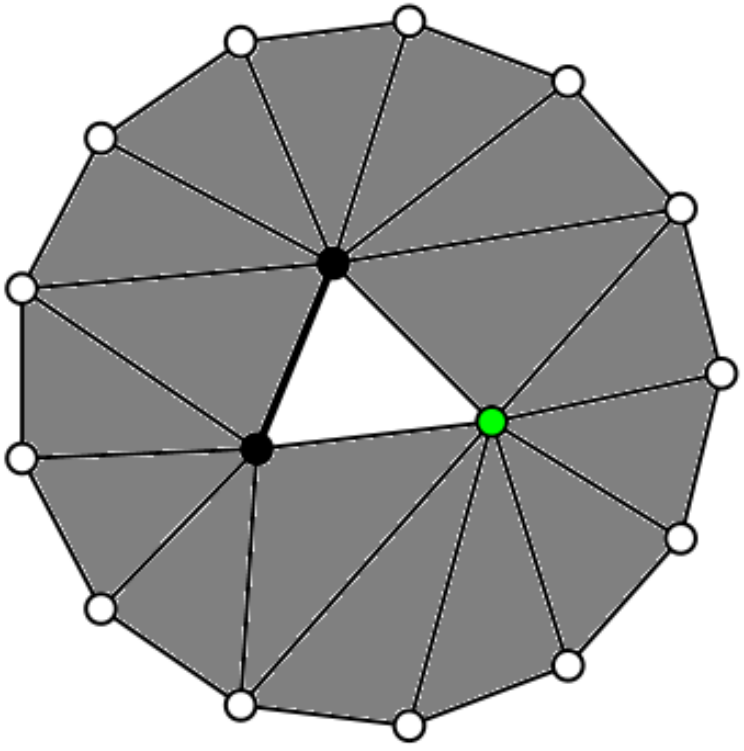
- Die Topologie könnte verändert werden
- Beispiel: Oberflächen mit Löchern
- das Loch könnte geschlossen werden!
- das gleiche gilt für einen Torus...



Topologieerhaltung

Worauf sollte man noch achten?

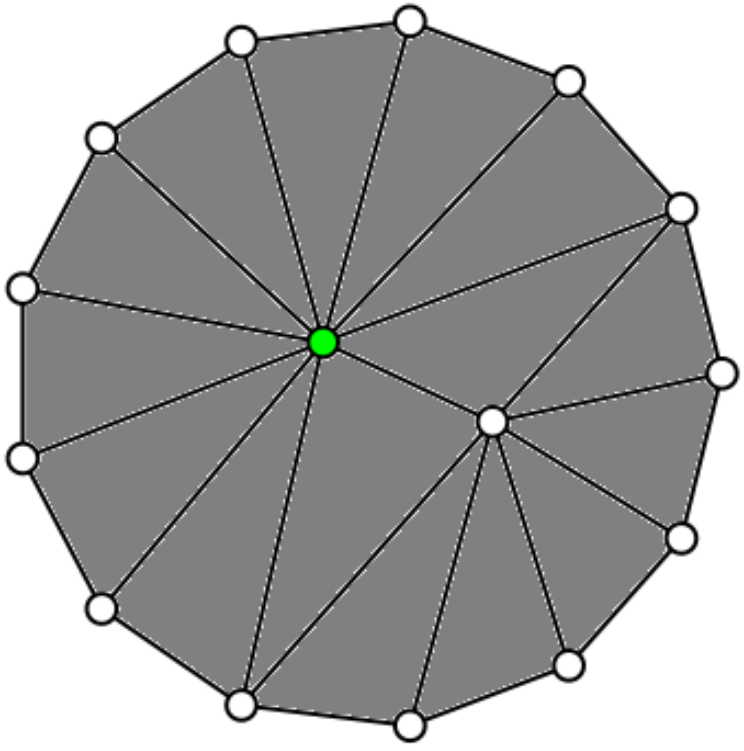
- Die Topologie könnte verändert werden
- Beispiel: Oberflächen mit Löchern
- das Loch könnte geschlossen werden!
- das gleiche gilt für einen Torus...



Topologieerhaltung

Worauf sollte man noch achten?

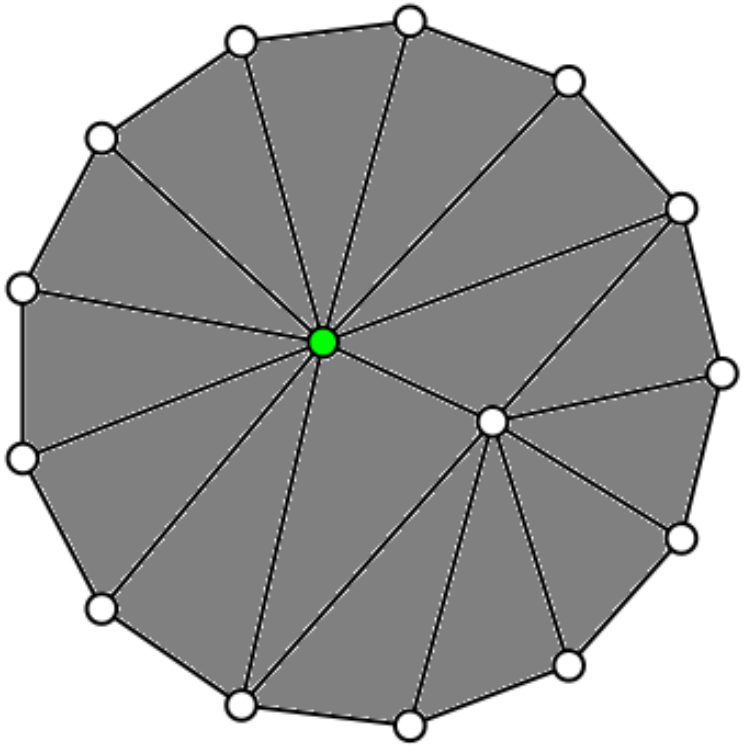
- Die Topologie könnte verändert werden
- Beispiel: Oberflächen mit Löchern
- das Loch könnte geschlossen werden!
- das gleiche gilt für einen Torus...



Topologieerhaltung

Worauf sollte man noch achten?

- Die Topologie könnte verändert werden
- Beispiel: Oberflächen mit Löchern
- das Loch könnte geschlossen werden!
- das gleiche gilt für einen Torus...



Fazit:

- Man muss vorher überprüfen ob eine Kontraktion die Topologie verändert!
- wie das geht, steht in meiner Ausarbeitung...

Literatur

M. Garland and P. Heckbert
Surface Simplification Using Quadric Error Metrics
SIGGRAPH 1997, 209–216
http://www.cs.cmu.edu/~garland/quadrics/

M. Garland
Multiresolution Modeling: Survey & Future Opportunities
EUROGRAPHICS 1999
http://mgarland.org/files/papers/STAR99.pdf

T.K. Dey et al.
Topology Preserving Edge Contraction
Publ. Inst. Math. (Beograd), 1998