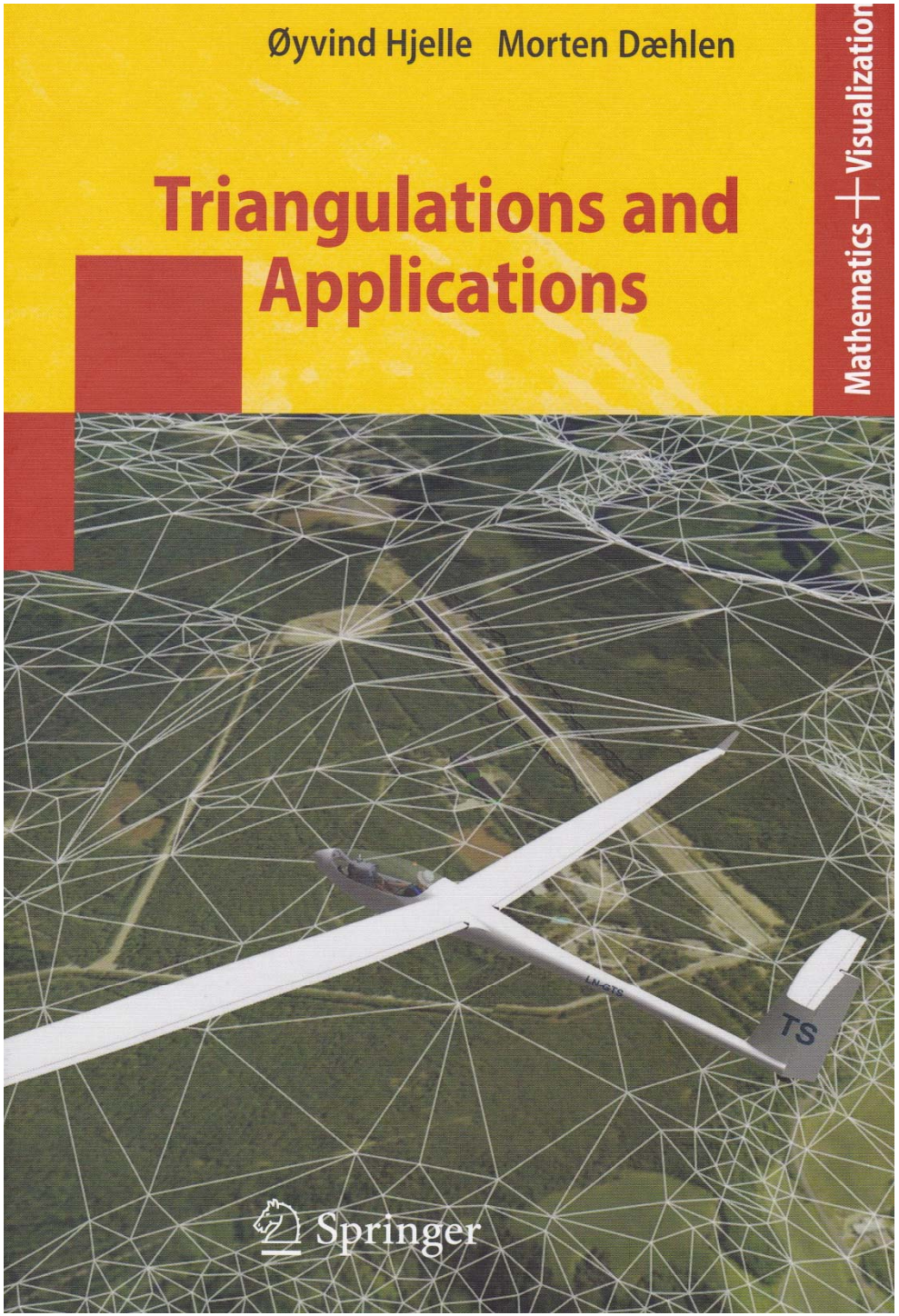
INF 4130
17. november 2011
Triangulering

Stein Krogdahl
Med sterk støtte fra
Petter Kristiansen

Skal først se på et eksempel fra Google Earth

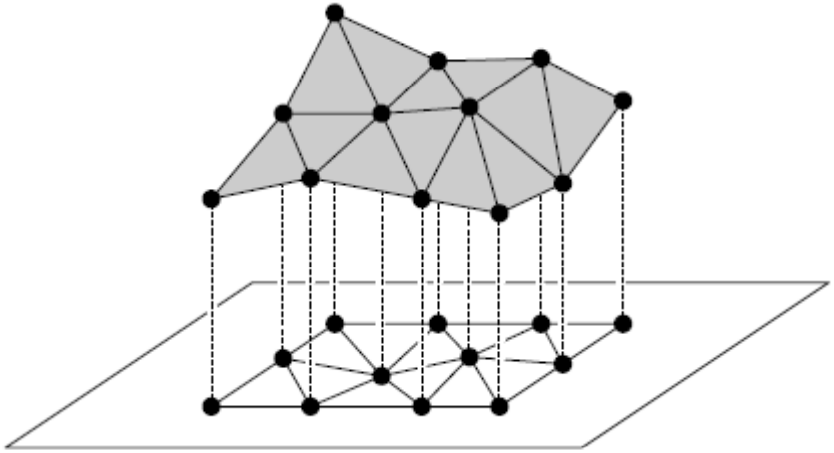
De bruker en underliggende triangulering av landskapet, men den ser ut til å synes bare på fjelltopper.

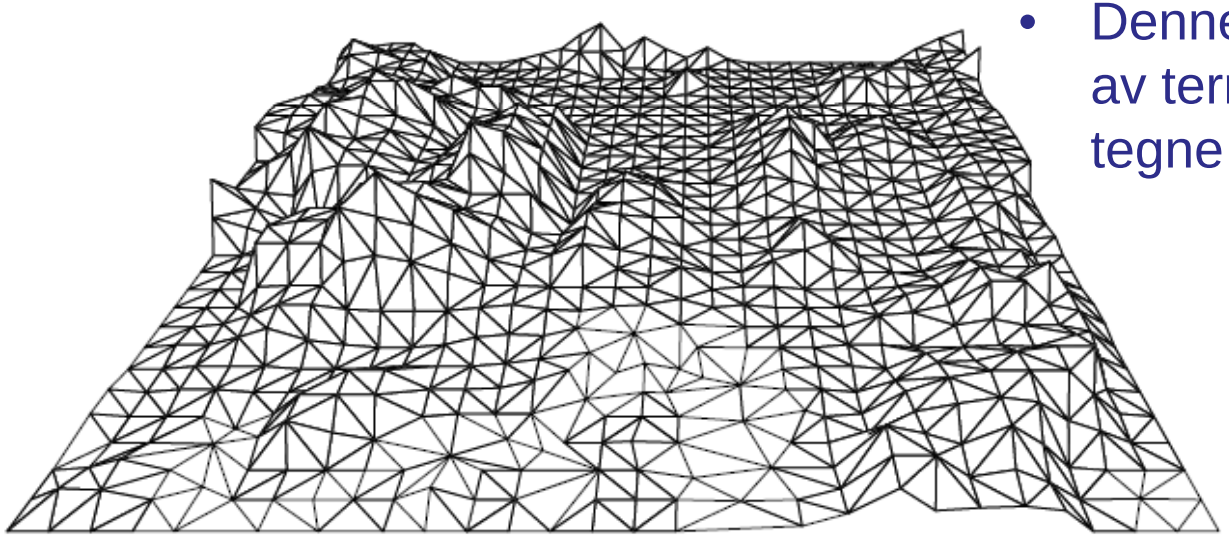
Vi ser på Mount Everest
(Det må du gjøre selv, og du må gå ganske nær, og se toppen fra siden. Da ser du tringler med ca. 60 meters sidekanter)



Fra en annen bok:

http://www.cs.uu.nl/geobook/interpolation.pdf

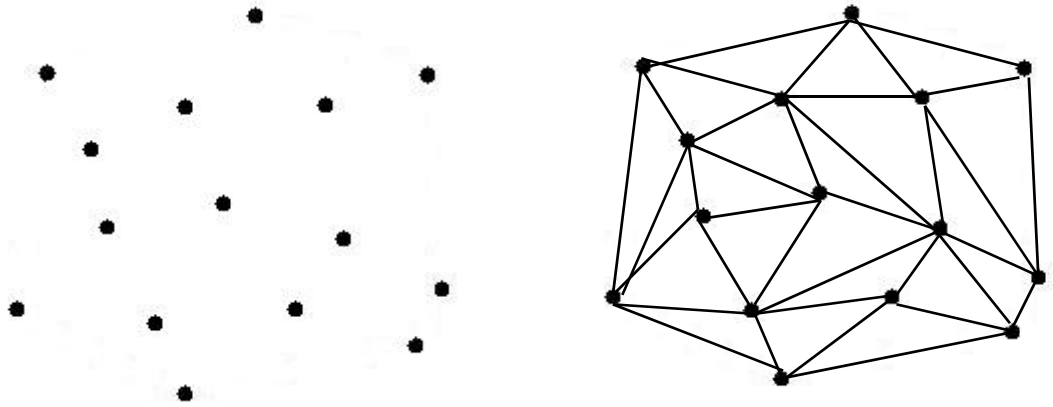




- Her har vi i en del terreng-punkter målt høyden over havet.
- Målepunktene er projisert ned på et tenkt havnivå, og det er der foretatt en ”fornuftig” triangulering.
- Denne gir så en triangulering av terrenget som er lett å tegne ut, skyggelegge, etc.

Utgangspunkt: En punktmengde i planet
Vi skal finne en ”triangulering”

Nedenfor er gitt en punktmengde, og en tilfeldig triangulering av den.

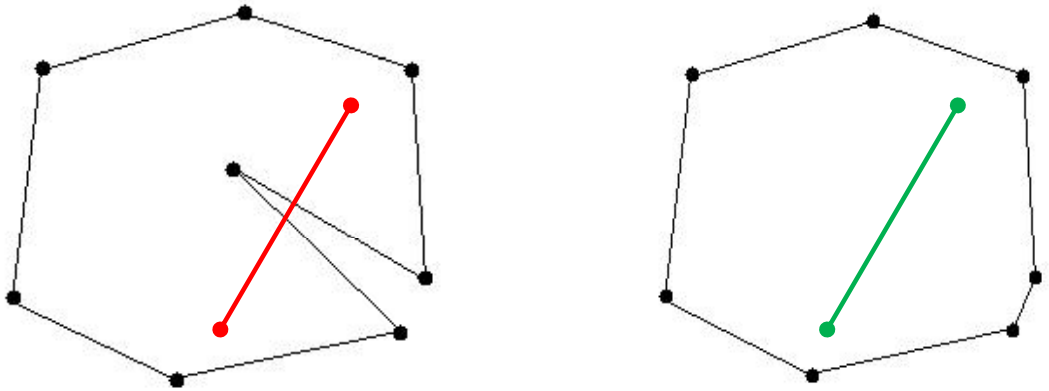


- Når man skal gjøre en triangulering skal man også ha angitt et polygon der hjørnene er punkter i punktmengden, og der alle de andre punktene er inne i polygonet
- Men vi skal her anta (slik vi har gjort over) at dette polygonet er den *konvekse innhyllingen* av punkt-mengden (se neste foil)

Om konveks innhylling

La P være en mengde punkter i et k-dimensjonalt rom, $P \in \mathbf{R}^k$.
(Vi skal her bare se på $k = 2$.)

Definisjon En mengde $Q \in \mathbf{R}^2$ er *konveks* dersom:
for alle punkter $q_1, q_2 \in Q$ ligger hele linjesegmentet q_1q_2 i Q.



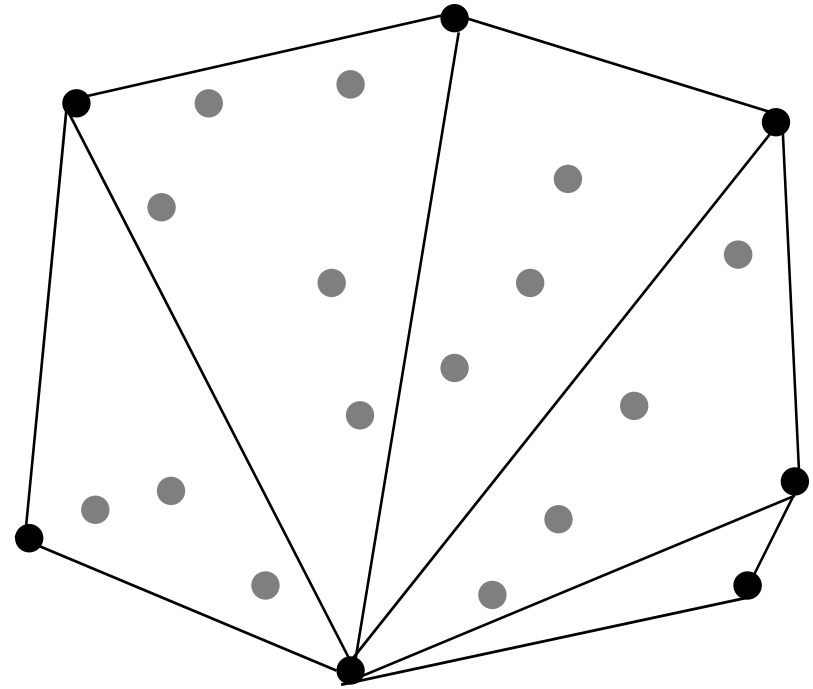
Definisjon Den *konvekse innhyllingen* til en mengde punkter $P \in \mathbf{R}^2$ er den minste konvekse mengde Q som inneholder punktene i P. Det er altså det omsluttende polyeder.

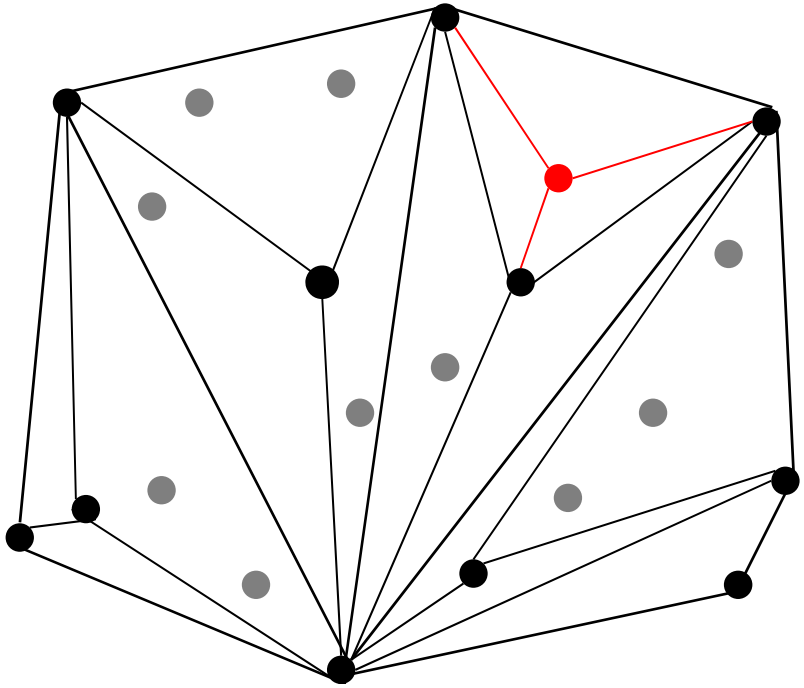
Hvor mange triangeler og kanter er det i en triangulering?

- Denne betrakningen gir også en algoritme for å finne en eller annen triangulering (og vi skal bruke en forfining av denne senere)
- Antall punkter langs det ytre polygon er n, og antall inni er m
  - Vi kan lage en triangulering for de ytre n punktene ved å velge ett av dem, og trekke kanter fra de n-3 motstående punktene.
  - Sammen med de ytre kantene gir dette n + (n-3) = 2*n-3 kanter
  - Og det gir n-2 triangler
  - Vi tar så med ett og ett av de øvrige punktene, finner det triangelet det ligger i og trekker linjer til dets tre hjørner. Det gir en økning på tre kanter og to triangler.
  - Dette skjer m ganger, og totalt får vi da:

    Antall kanter: $2*n - 3 + 3*m = 2*n + 3*m - 3$

    Antall triangler: $n - 2 + 2*m = n + 2*m - 2$
- Dette angir også antall kanter og triangler for enhver triangulering av så mange punkter.
- Vi skal ikke bevise dette, men satser på at det virker intuitivt rimelig





Oppbygging av en eller annen triangulering

(og illustrasjon til forrige side)

- Har i alt *n* noder på det ytre polygon, og *m* indre noder (inkluderer alle sorte, røde og grå)
- Tegningen viser steget når vi legger ett nytt punkt (rødt) til trianguleringen
- En ny node gir
  - Tre ekstra kanter
  - To ekstra triangler
- Derfor får vi:

  Antall kanter:

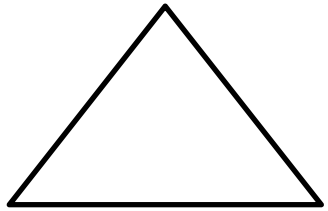
  2*n – 3 + 3*m = 2*n + 3*m – 3

  Antall triangler:

  n - 2 + 2*m = n + 2*m - 2

Det er mange forskjellige trianguleringer over en gitt punktmengde

- Vi antar at følgende sær-tilfeller ikke forekommer:
  - (1) Fire noder på én sirkel.
  - (2) Alle noder ligger på én rett linje



- Og hva er en *god* triangulering?
  - Vanligvis en der hvert triangel er så nær en likesidet trekant (med alle vinkler lik 60^{O}) som mulig.

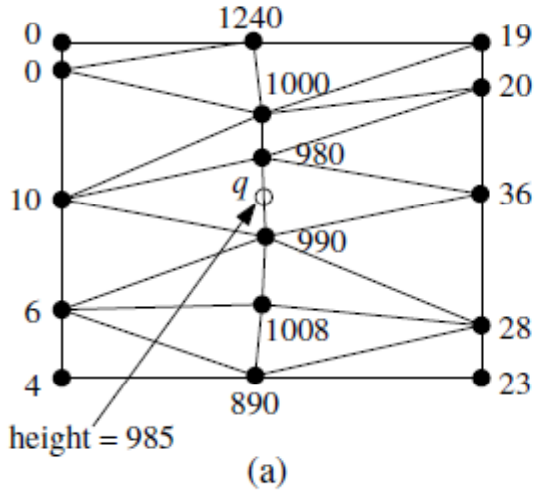
  To greie mål kunne være:
  - Minimalisering av maksimal vinkel: Den største vinkelen er så liten som mulig (merk: Den største vinklen vil alltid være minst 60^{O})
  - Maksimalisering av minimal vinkel: Den minste vinklen (som aldri er større enn 60^{O}) er så stor som mulig.

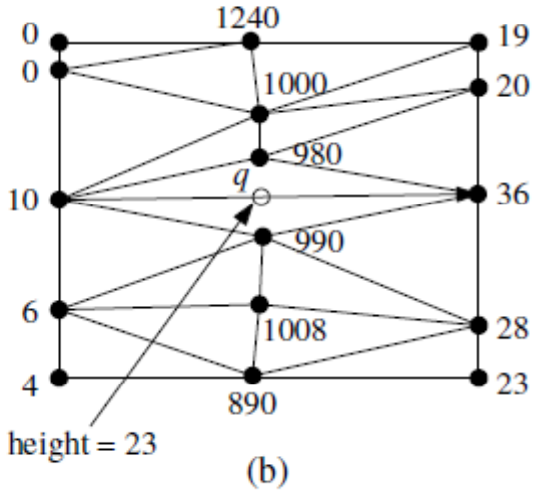
- Det viser seg at *maks-av-min* er vesentlig lettere å behandle enn min-av-max, så den er absolutt mest brukt.

Eksempel: Bedre med så store vinkler som mulig

Figuren viser høydemålinger fra terrenget

- Man vil ønske å "interpolere" over punktene
- D.v.s. at man finner høyden dette punktet har i det aktuelle triangelet (eller den aktuelle kanten, som her)
- Vi ser at den venstre gir et intuitivt høyderesultat for *q* enn det den høyre gjør.

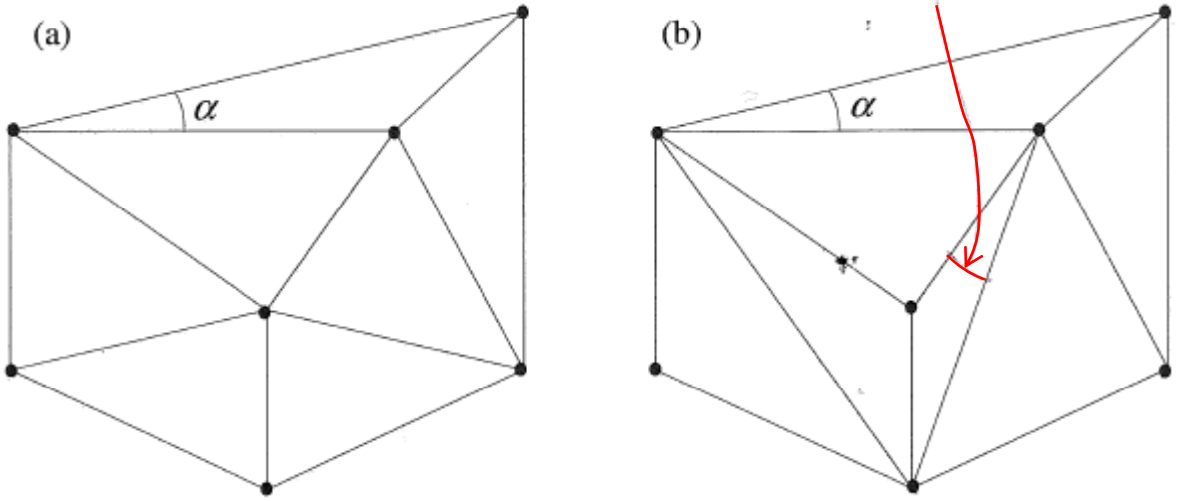




Definisjon av Delaunay-triangulering (maks-av-min)

- *DEF:* En triangulering over n gitte punkter som har *størst mulig minste vinkel av alle mulige trianguleringer* av disse punktene kalles en Delaunay-triangulering.
- Presisering: Når vinklene sorteres fra største til minste er Delaunay-trianguleringen den som er *leksikiografisk* størst

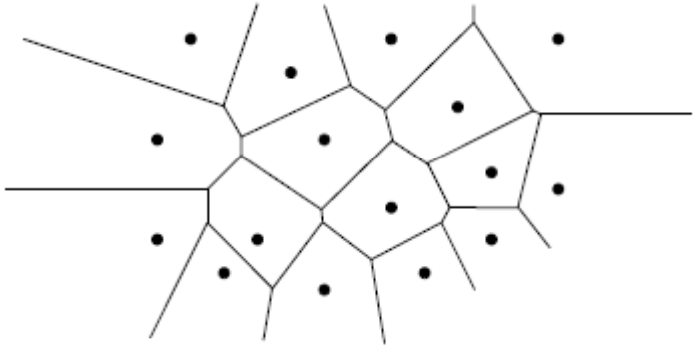
Denne vinklen er større enn alfa, men mindre enn alle andre vinkler i (a)

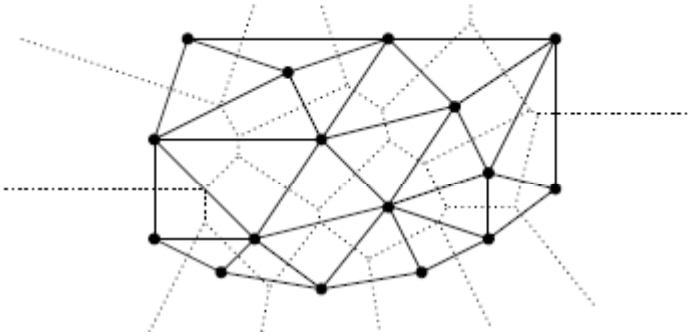


- Figur (a) er en Delaunay-triangulering, men ikke (b)
- De har like store minste-vinkler (alfa), men nest minste vinkel er størst i (a)

Veroni-diagram (midtlinje-prinsippet)

- Diagrammet til venstre er Veroni-diagrammet for de gitte punktene
  - Det er laget ut fra at punktene er øyer og at havet skal tildeles hver øy etter midtlinjeprinsippet

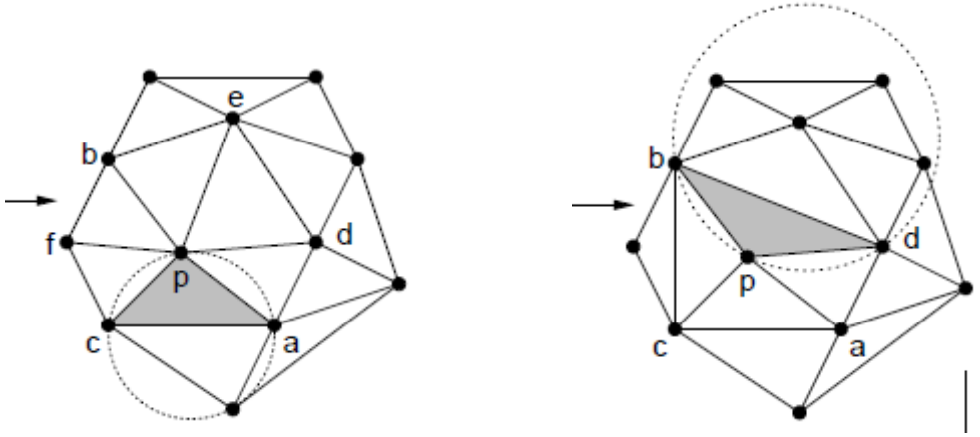




- Delauney-trianguleringen får vi da
  - ved å trekke linjer mellom de punkter/øyer som har et stykke felles grense mellom sine havområder.
  - Merk at disse linjene ikke alltid går gjennom det aktuelle grense-stykket
- Delaunay-grafen er den *duale grafen* til grafen som dannes av havgrensene

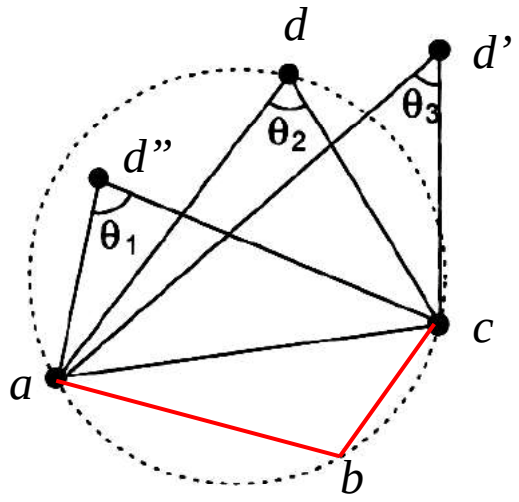
En egenskap, som kan vises å være ekvivalent med de andre definisjonene

- En triangulering er en Delaunay-triangulering hvis og bare hvis:
  - For alle triangler gjelder at det indre av sirklen gjennom de tre hjørnene ikke inneholder noen punkter.
  - Det er jo ikke i utgangspunktet klart at det alltid finnes en slik triangulering, men det gjør det altså, og det er bare én slik (om det ikke er sær-tilfeller), og det er nettopp Delaunay-trianguleringen.



- Den til venstre er en Delaunay-triangulering, men ikke den til høyre
  - Merk at vi her har et særtilfelle (fire punkter på en sirkel), og da er ikke Delaunay-trianguleringen entydig.
  - Vi kunne byttet ut kanten c-a med en kant fra p og nedover
- Det er denne definisjonen vi skal bruke i algoritmen for å finne en Delaunay-triangulering

Noen sannheter om vinkler og sirkler

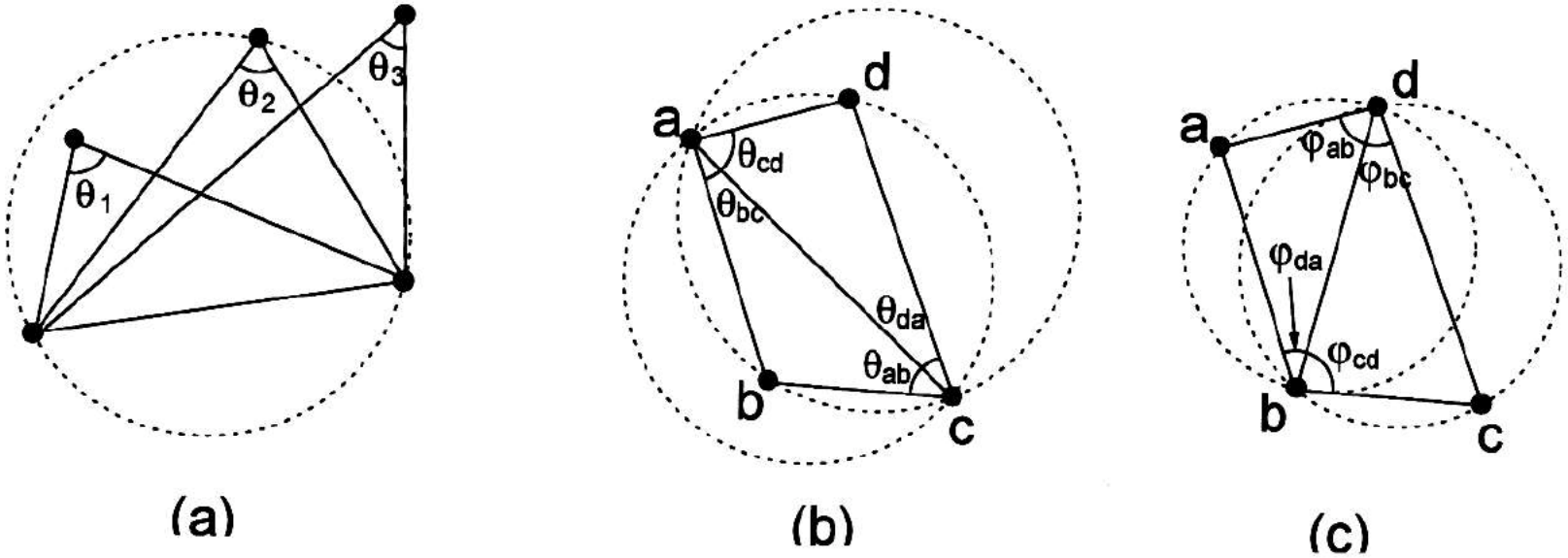


- Til venstre er teta1 > teta2 > teta3
- For å avgjøre om et punkt d er inni (eller på eller utenfor) sirklen gjennom a, b og c må vi sørge for at a, b, c og d kommer i rekkefølge mot klokka rundt firkanten, og så beregne determinanten:

$$\text{inCircle}(a,b,c,d) = \det \begin{pmatrix} a_x & a_y & a_x^2+a_y^2 & 1 \\ b_x & b_y & b_x^2+b_y^2 & 1 \\ c_x & c_y & c_x^2+c_y^2 & 1 \\ d_x & d_y & d_x^2+d_y^2 & 1 \end{pmatrix} > 0$$

Denne beregningen kan optimaliseres, og derved gjøres nokså effektivt

Delaunay-trikset



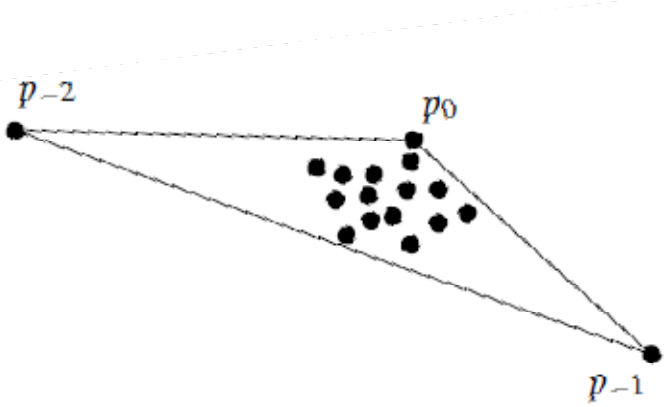
Vi antar som på figurene (b) og (c) at firkanten a-b-c-d er konveks.

- Vi ser her at dersom d ligger inni sirklen som inneholder a, b og c, så vil også sirklen gjennom a, c og d inneholde b.
  - Dermed er altså Delaunay-kravet ikke oppfylt
- Om vi da bytter ut kanten a-c med kanten b-d så vil a ligge utenfor sirkelen gjennom b, c og d, og c vil ligge utenfor sirklen
  - Dermed er Delaunay-kravet oppfylt, hvertfall lokalt

Algoritme som finner Denaulay-triangulerингen

- Vi har gitt et antall noder i planet.
- Start: For å ha en triangulering å starte med legger vi til tre noder, slik at det triangelet de utgjør inneholder alle de gitte nodene
  - Og vi lar dette ene triangelet være vår initielle triangulering, se under
- Vi gjør som vi gjorde da vi fant en helt tilfeldig triangulering på de første foilene (da vi tellet kanter og triangler)
  - Nemlig å legge til én og én node, og for hver vi legger til trekke de tre kantene til hjørnene av den trekanten de falt inni.
  - Dette kan gi en triangulering som *ikke* er Delaunay, og før vi legger til en ny node til vil vi gjenopprette Delaunay-egenskapen.
  - Hvordan dette gjøres følger på neste side.

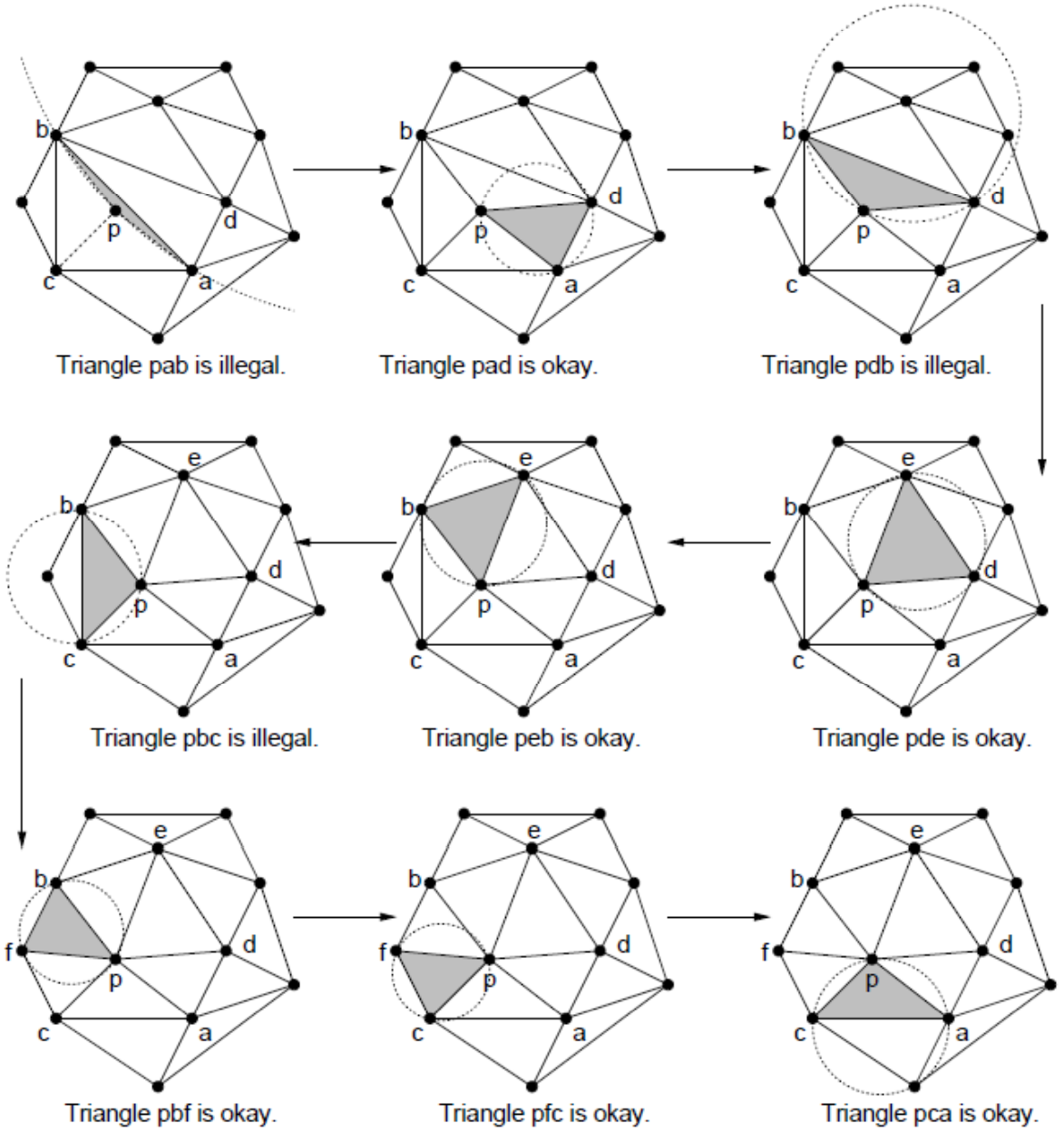
- Det er her viktig å la hvertfall to av punktene være *laaaangt* unna. Da blir det lettere å fjerne dem igjen til slutt. Det tredje punktet kan godt være fra mengden



Gjenoppretting av Delaunay-egenskapen

- Steget i algoritmen er altså slik (figur neste side):
  - Vi har en Delaunay-triangulering av den delemengde av nodene vi *har* tatt inn
  - Vi velger én ny node *p*, finner hvilket triangel den ligger i, og lager kanter fra noden til hjørnene i dette triangelet.
- Dette kan føre til at trianguleringen ikke lenger er Delaunay, og vi ønsker da å gjenopprette denne egenskapen før vi fortsetter
  - For å gjøre det holder det å se på *alle* trianglene som har ett hjørne i p, og se om Delaunay-egenskapen holder "lokalt "for dette i forhold til det motstående triangelet og dens fjerneste hjørne (men vi *beviser ikke* at dette er nok).
  - Om et triangel med p som hjørne og dets motstående triangel *ikke tilfredstiller dette* vil disse to trianglene alltid utgjøre en *konveks* firkant, og vi gjør da Delaunay-trikset på denne firkanten.
  - Da blir denne firkanten hvertfall "lokalt Delaunay", men noen av de andre trekantene med ett hjørne i *p* kan da ha fått ødelagt sin Delaunay-egenskap.
  - Mengden av triangeler med p i et hjørne vil dermed stadig forandre seg (og øke i antall), og man må fortsette prosessen til det hele roer seg.
  - Men, det *vil* roe seg, siden det ved hver bruk av Delaunay-trikset vil komme én ny kant mot p, og det finnes bare et endelig antall andre noder som *p* kan få kanter til.

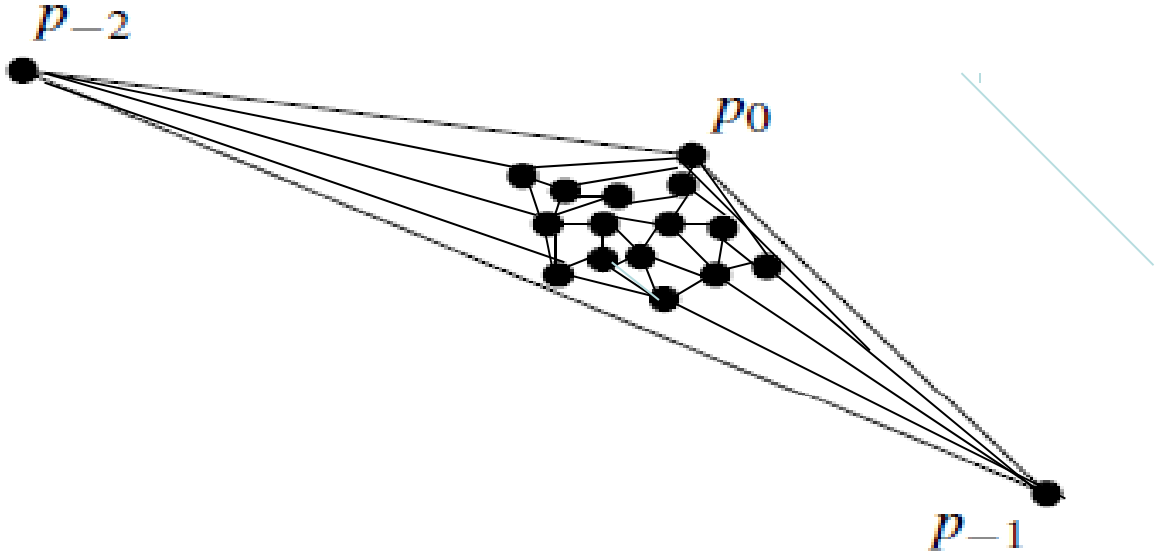
Tillegg av én ny node, med Delaunay-reperasjoner



Start og avslutning av algoritmen

Slik det er framstilt her starter vi altså algoritmen ved å legge til tre ekstra noder slik at triangelet de danner inneholder alle de opprinnelige nodene.

- Dette triangelet utgjør da start-trianguleringen, og:
- Det gjør at nye noder som tas inn alltid vil ligge inne i et triangel i den trianguleringen vi allerede har laget.
- Problemet er da å bli kvitt de tre ekstra punktene igjen til slutt. Dette er ikke vanskelig, men vi ser ikke på detaljene.
- Fjerningen blir enklest om ekstra-punktene ligger L-A-N-G-T unna nodemengden. Da kan det bare være å fjerne dem, og kantene til dem.

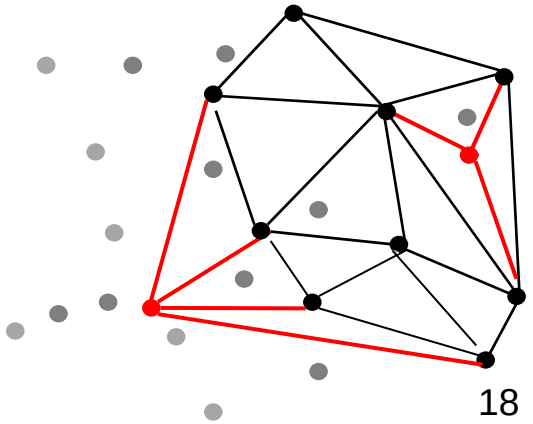


Alternativ start og utvidet hovedsteg i algoritmen

I stedet for å starte med å legge til tre ekstra noder kan man starte med å velge tre tilfeldige noder fra mengden.

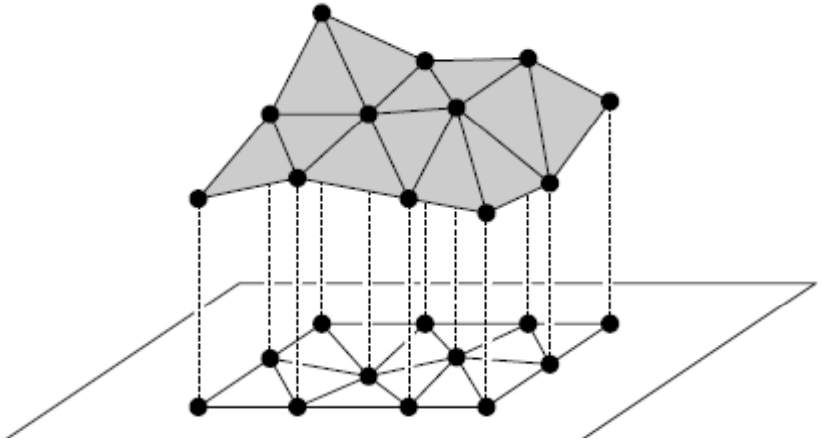
- Når man da legger til én og én node vil noen falle inne i et eksisterende triangel slik som beskrevet til nå.
- MEN, en ny node kan også falle helt utenfor de triangler som er laget til nå, og da må man bruke en annen metode for å få passet den nye noden inn blant de gamle.
- Dette gjøres ved å trekke kanter fra den nye til alle de ferdigbehandlede noder som er direkte synlige fra den nye.
- Dette kan igjen ødelegge Delaunay-egenskapen, og det må repareres. Dette kan gjøres etter samme prinsipp som i det tilfelle vi har sett på, men det må forandres noen steder. Vi går ikke inn på dette i detalj.

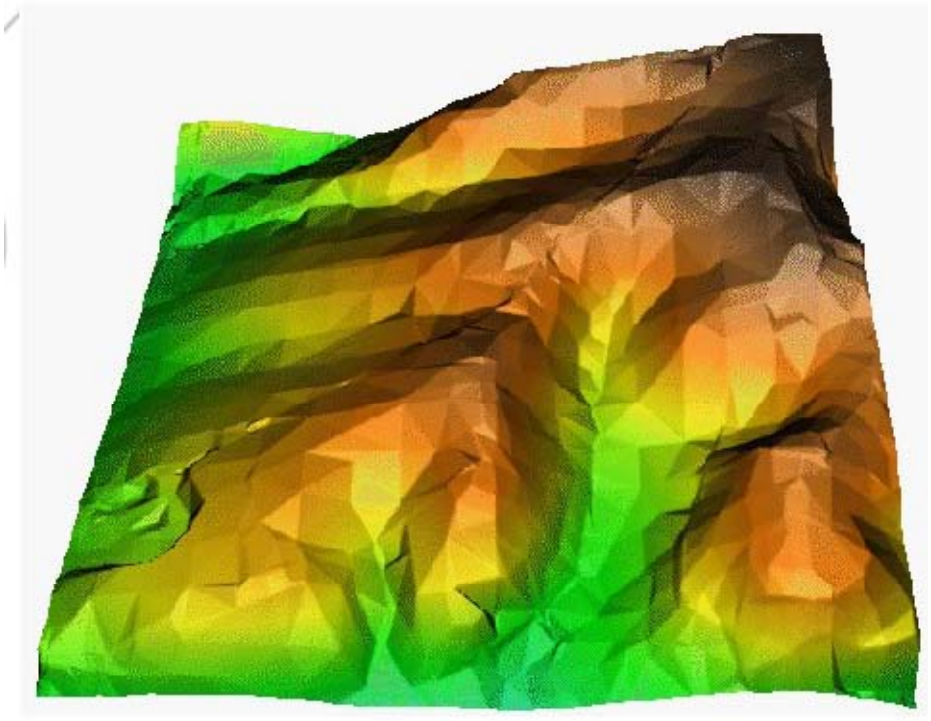
Effektivitet: Med god data-representasjon og litt optimalisering går begge metoder i tid $O(n \log n)$



Litt om terreng-representasjon

- Som antydet i starten kan man representere terreng ved å ha en ”flat” triangulering, og så angi høyder i hvert punkt
- Da er det lett å regne ut hvordan hvert triangel ligger i terrenget og hvordan de er vinklet i forhold til andre terreng-triangler.
- Det er også forholdsvis greit å tegne det ut i for eksempel i perspektiv, og sørge for å hoppe over skjulte flater
- Det er lett å farge hvert triangel f.eks. slik:
  - Skyggelegge den ut fra vinklen til en gitt lyskilde, og ting som skygger
  - Gi den en grunnfarge ut fra hvor høyt den ligger

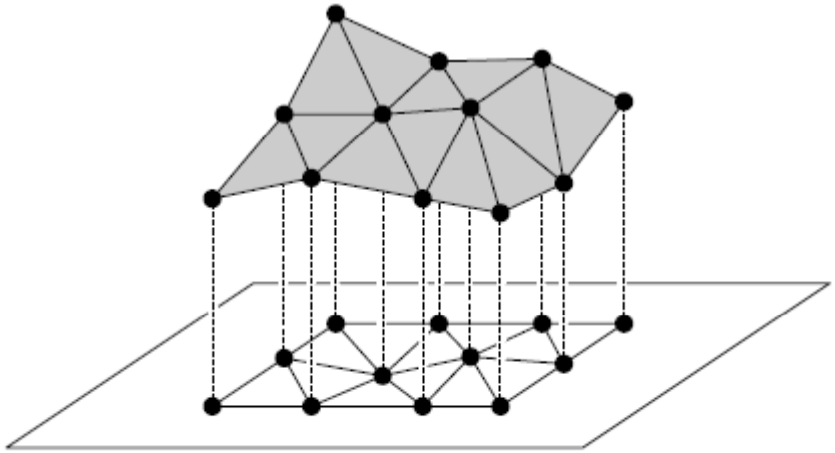


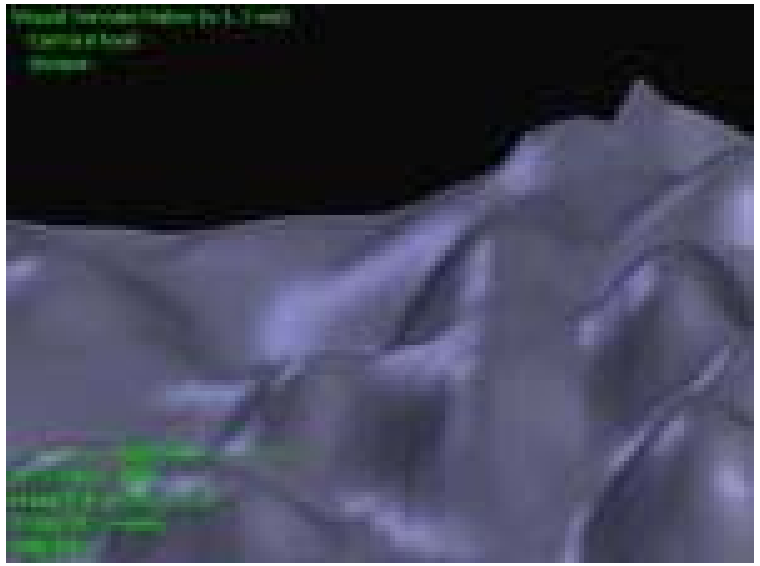


Mer om terreng-representasjon

(Mer spørsmål enn svar)

- Det finnes metoder for å avrunde en triangulering med høyder, og dermed få flater over trianglene som hvertfall stemmer over ens i første deriverte med nabo-flatene.
- Da vil man kanskje ikke insistere på at flaten har nøyaktig riktig høyde i hvert hjørne
- Det ser ut til at det her er vanlig å bruke ”minste kvadraters metode” for å få det riktigst mulig
- Det ser ut til at Google Earth gjør slik avrunding, bortsett fra på fjelltopper.
- Og det er meget aktuelt i forbindelse med spill.





Mer om terreng-representasjon

- Om vi har masse målepunkter er det ikke sikkert vi behøver å bruke alle målepunktene for å få god nok terreng-representasjon.
- Det er ofte både unødvendig og for ressurskrevende
- Hvordan kan man fornuftig gjøre et utvalg av punkter slik at man får en rasjonell og ”riktig nok” representasjon av terrenget?
- Altså, færre tringler der terrenget er flatt, men tettere der det humpte