Nähtamatute pindade eemaldamine

Dan Bogdanov
db@ut.ee

Kus me oleme?

- ~~Tippude transformeerimine & projitseerimine~~
- **Nähtamatu tahkude & tippude eemaldamine**
- Polügonide rasteriseerimine
- Pikslite värvimine
- Pikslite ekraanile väljastamine

Pindade eemaldamine: Miks?

- Tööaja kokkuhoid
- Reaalajas töötava rakenduse loomine
- Geomeetria lihtsustamine

- Kui me seda nagunii ei näeks (või vahet ei teeks), siis <u>miks peaks arvuti seda joonistama?</u>

Pügamine

- Peame lõikama ära oma maailmast need osad, mis ei mahu projitseeritavasse ruumi.
- Tihti tuleb objekte tükkideks teha.
- Üle ääre ulatuvad osad “pügatakse” ära.

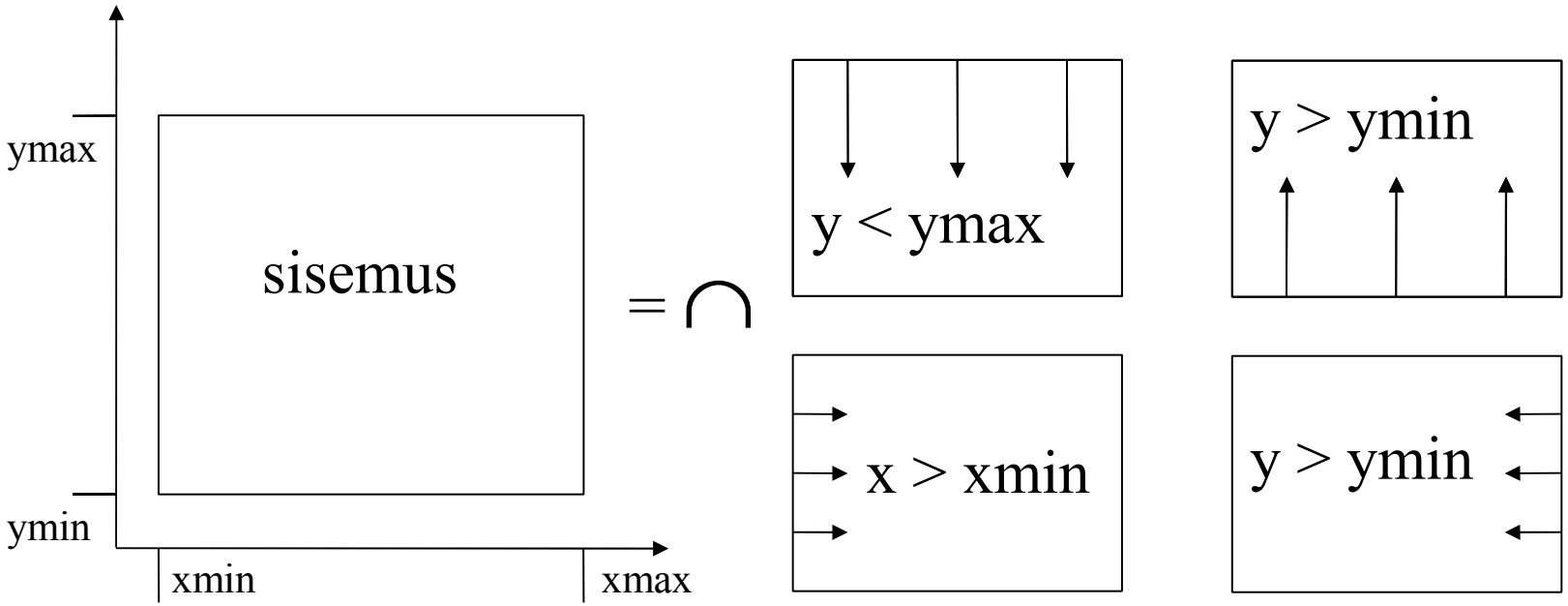
Normaliseerimine

- Sõltuvalt meie püramiidi paigutusest võib pügamine olla päris keeruline.
- **Lahendus: kujutame ta enne pügamist kuubiks**.
- Ülesanne lihtsustub oluliselt ning leiduvad efektiivsed algoritmid selle lahendamiseks.

Sirgjoone pügamine

- Vaja leida sirgjoone ja kuubi ühisosa.
- Kuup on määratud tema tahkude tasandite kaudu.
- Kui punkt on kõikide tasandite vahel, siis on ta ka kuubi sees.
- Vaatleme ülesande lihtsustust kahemõõtmelises maailmas.

Ristküliku sisemus



Üldine lähenemine

- Kas joone otspunkt on ristküliku sees? (kontrollime punkti suhet sirgega)

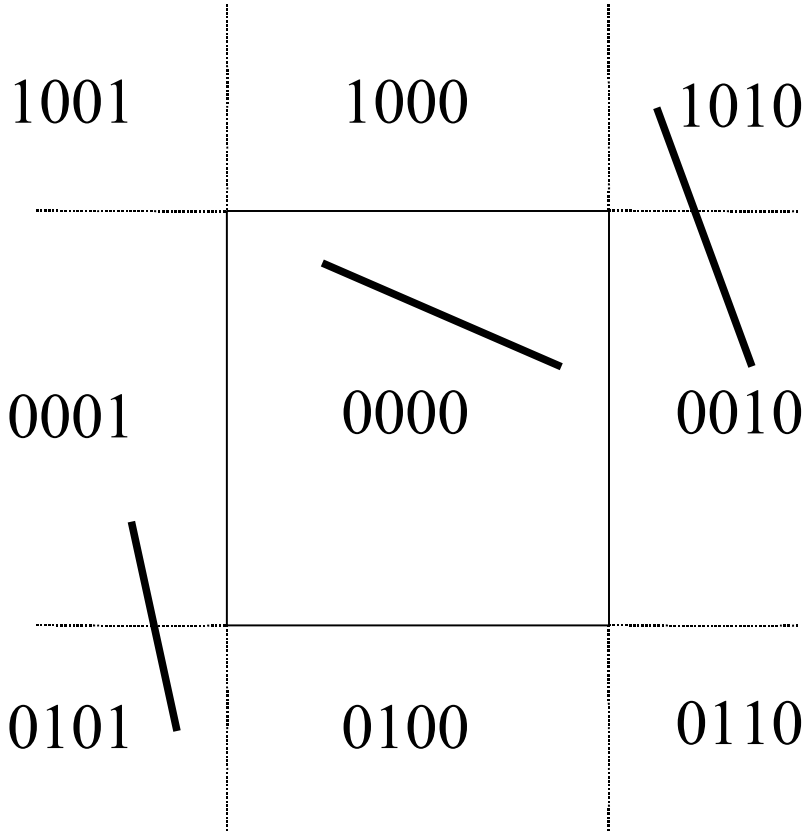
- Kui joone otspunkt on väljaspool, leiame joone ja ristküliku serva lõikepunkti ning teeme sellest uue otspunkti.

Cohen-Sutherlandi algoritm

- Sirged kodeeritakse nelja bitiga vastavalt nende otspunktide koordinaatidele.

- Kodeerimisreeglid:
  - Bitt 1: $y > ymax$
  - Bitt 2: $y < ymin$
  - Bitt 3: $x > xmax$
  - Bitt 4: $x < xmin$

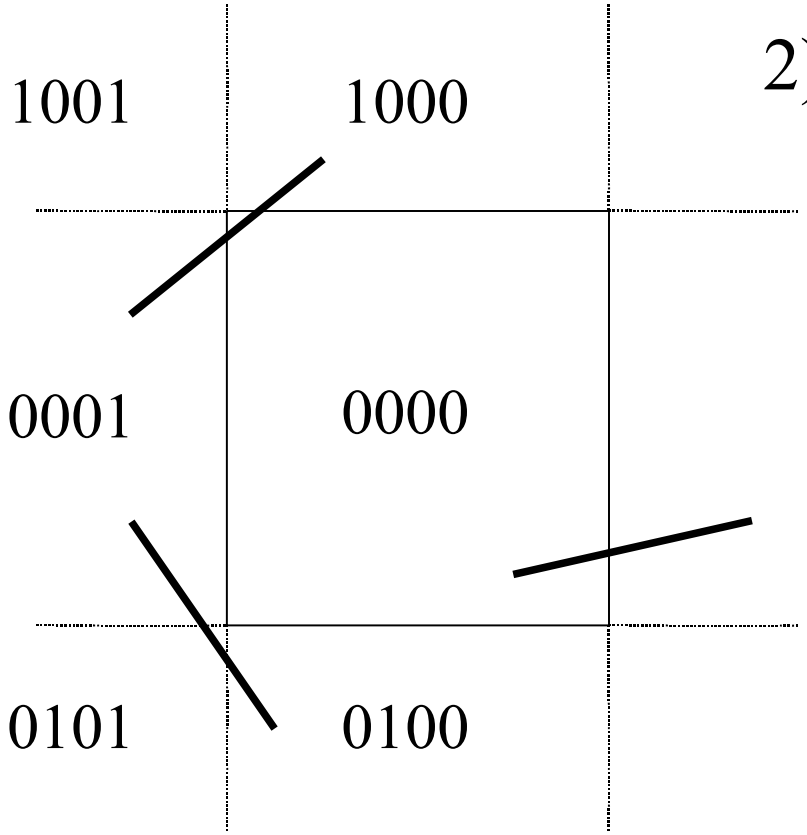
Cohen-Sutherland: Näide



1) Triviaalsed juhtumid:

- Mõlemad tipud 0000
- Tippude bitihaaval summa ei võrdu nulliga

Cohen-Sutherland: Näide (jätk)



2) Mittetriviaalsed juhud:
- Vali tipp väljaspoolt
- Vali ristuv serv ruudust
- Leia lõiketipp
- Asenda esialgne tipp lõiketipuga
- Korrata kuni kõik juhud on triviaalsed

Hulknurga pügamine

- Vaja leida hulknurga osa(d), mis jäävad etteantud pügamispiirkonda
- Variante:
  - Hulknurk täidab pügamispiirkonna täielikult
  - Hulknurgast lõigatakse pügamisel üks tükk
  - Ühest hulknurgast võib pügamisel tekkida mitu
  - Hulknurk ei kuulu pügamispiirkonda

Sutherland-Hodgmani algoritm

- Alamülesanne:
  - Pügada hulknurk (selle tippude hulk) ühe pügamisruumi tahu (tasandi) järgi.
  - Väljastada pügatud hulknurga tipud.
- Rakendada alamülesannet kõigi kuue kuubi tahu peal.
- Võime vaadelda ka kahemõõtmelist varianti.

Sutherland-Hodgmani algoritm

- Vaatleme antud alamülesande lahendamist:
  - Läbime järjestatud tippude loendi. Kui esimene tipp on sees, siis väljastame selle tipu. Edasi vaatleme üleminekuid tipult järgmisele tipule:
  - sees→sees: väljastame teise tipu
  - väljas→välja: ei väljasta midagi
  - sees→väljas: väljastame serva lõiketipu ruumi äärega
  - väljas→sees: väljastame lõiketipu ruumi äärega ning teise tipu

Täiendavad probleemid

- Mõned tipud võisid ära kaduda.
- Mõned tipud võisid juurde tekkida.
- Peame ümber arvutama esialgses hulknurgas kehtinud tekstuurimis/värvimiskoordinaadid.

Peidetud pindade avastamine

- Meetodid jaotuvad kaheks:
- 1) Objektipõhised meetodid
  - Uurivad objektide asetust ja suhteid ruumis
  - Lahendavad ülesandeid tervete objektide kaupa
- 2) Pikslipõhised meetodid
  - Analüüsivad objektide projitseerimist konkreetsete pikslite juures.

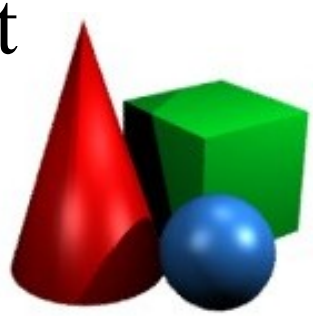
Tagakülgede tuvastamine

- Inglise keeles *back-face detection/culling*
- Eesmärgiks on leida objekti tahud, mis on meie suunas „seljaga“. Näide: kuubi tagumine tahk.
- Triviaalne meetod: Kui asume tipus *(x, y, z)*, on tahk suunatud meist eemale, kui tema tasandi võrrandi jaoks kehtib:

$$Ax + By + Cz + D < 0$$

Tagakülgede tuvastamine

- Lihtsustatud meetod.
- Vaatleme tasandi normaalvektorit N ning kaamerast väljuvat vektorit V. Tahk on tagakülg, kui $\mathbf{V \cdot N > 0}$.
- Parema käe koordinaatsüsteemis, (kui vaate suund on *z*-teljega paralleelne ning negatiivse suunaga), on hulknurk tagakülg, kui selle normaalvektori *z*-koordinaat on negatiivne.



Sügavuspuhver

- Iga piksli puhul võrreldakse sellele punktile projitseeritavate objekti kaugust *z*-teljel. Sellest ka nimi – *z-buffer*.
- Sügavuspuhvrisse salvestatakse antud pikslil nähtava objekti vastava punkti intensiivsus.
- Koosneb tegelikult kahest puhvrist:
  - sügavuspuhver – kaugus projektsioonitasandist
  - pikslipuhver – vastava piksli intensiivsus

Sügavuspuhvri täitmine

- Algväärtused:
  - sügavus (x, y) = 0
  - intensiivsus (x, y) = taustaintensiivsus
- Iga objekti iga tahu iga piksli (x', y') jaoks:
  - Leia piksli x', y' jaoks z' ning projektsiooni intensiivsus i.
  - Kui z > sügavus (x, y), siis
    - sügavus (x, y) = z ning intensiivsus (x, y) = i

Sügavuspuhvri täitmine

- Z-koordinaate arvutatakse tasandi võrrandist:

$$z' = (-Ax - By - D) / C$$

- Järgmine piksel reas:

$$z' = (-A\,(x + 1) - By - D) / C$$

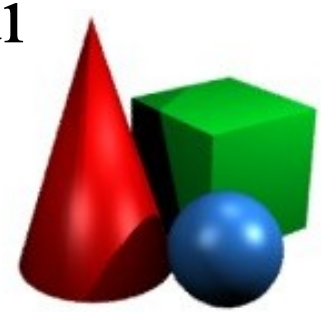
- Järjestikuseid arvutusi saab optimeerida, jättes meelde konstantsed väärtused. (nagu näiteks -A / C).

Ridahaaval töötlemine

- Sarnane sügavuspuhvrile, aga:
  - vajab rohkem informatsiooni objektide struktuuri kohta – servade ja tahkude tabeleid
  - töötleb piksleid horisontaalridade kaupa
  - võimaldab algoritmi optimeerida, sest kasutab rohkem informatsiooni

Ridahaaval töötlemine

- Servade tabel sisaldab vastavate sirgete koordinaate.
- Tahkude tabel sisaldab vastavate tasandite valemite kordajaid ning informatsiooni pinna värvi kohta.
- Iga rea jaoks hoitakse nimekirja aktiivsetest (antud reaga lõikuvatest) servadest. See nimekiri on sorteeritud x-koordinaadi järgi kasvavalt.



Ridahaaval töötlemine

- Lisaks hoitakse iga objekti juures lipukest, mis on püsti parajasti siis, kui oleme vaadeldavas reas jõudnud antud objekti „kohale“.
- Kui mingil hetkel on aktiivne ainult üks objekt, st midagi ei ole selle taga, siis ei ole sügavusarvutusi vaja teha.
- Kui objekte on mitu, tuleb võrrelda kauguseid ning lähim salvestada.

Ridahaaval töötlemine

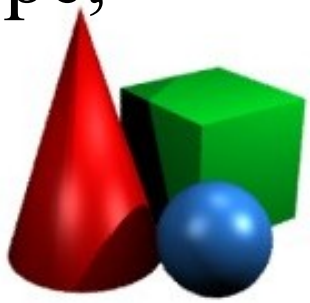
- Kui kahel järjestikusel real on aktiivsete servade nimekirjad samad, saame teise rea töötlemisel kasutada esimese rea jaoks leitud tulemusi.
- Kui servade lõikumistes muudatusi pole, saame eeldada sama kauguste suhet, mis oli eelmisel real
- NB! Üldiselt on siiski võimalik, et objektide kaugus vahepeal muutub.

BSP-puud

- *BSP – Binary Space Partitioning*
- Ruumi pooleks jaotamiste (jaga-ja-valitse) teel saadakse lihtsalt sorteeritav objektide struktuur.
- Pildi renderdamisel joonistatakse tagumised objektid enne, nii jäävad need esimeste taha.

BSP-puud (näide)

- Ruumi jagatakse pooleks poolitustasandiga.
- Tasandist vaataja poole jäävad objektid on „ees“ ning teisel pool tasandit on „tagumised“ objektid. Kõik poolitustasandid võivad jaotada objektid „eesmisteks“ ja “tagumisteks“.
- Lõpuks tuleb lihtsalt puust lugeda tippe, alustades tagumistest.



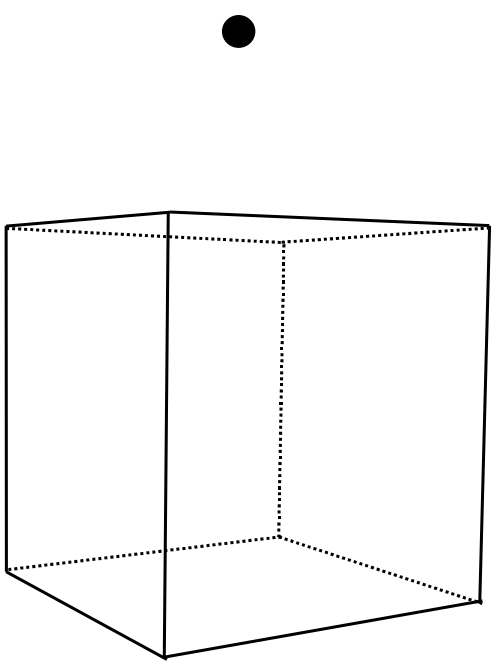
BSP-puud

- BSP-laadseid lahendusi kasutatakse päris palju.
- Nimelt on võimalik vastavalt kaamera asukohale objektid ümber grupeerida ning saada puu, kus jällegi kõik nähtavad objektid on teiste „ees“.
- Mitmed mängumootorid põhinevad BSP-lahendustel (Quake, Half-Life).

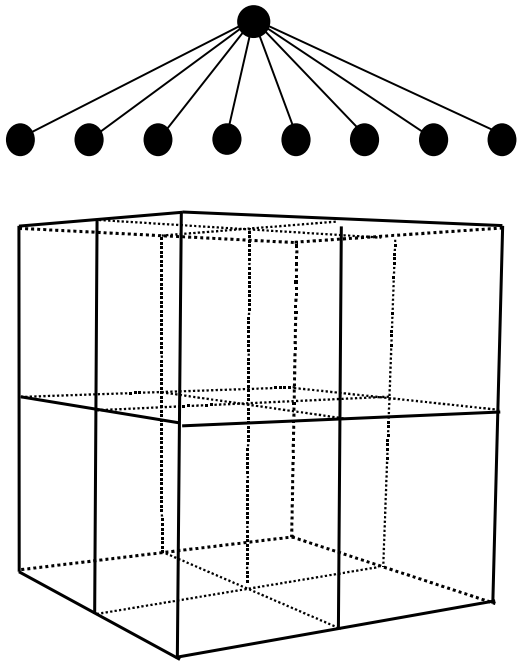
Kuupide kaheksandpuu

- Inglise keeles *octree*.
- Samuti ruumi jaotamise vahend, kuid jaotust viiakse läbi teisiti.
- Vaadeldakse kuubikujulist ruumi, mis jaotatakse kaheksaks võrdse suurusega kuubiks. Iga saadud kuubi võib omakorda tükkideks jagada.

Kuupide kaheksandpuu



tase 0



tasemed 0 ja 1

Kaheksandpuu koostamine

- Kuupe jaotatakse tükkideks, kuni etteseatud piiri saabumiseni või hetkeni, mil kõigil objektidel on puus oma tipp.
- Tulemusena saadakse struktuur, mille peal on hea teha nähtavuskontrolle.
- Kui kasutaja ei näe suuremat kuupi, ei näe ta ka midagi, mis on seal sees.
- Seda saab ära kasutada mitmete kontrollide lihtsustamisel.

Detailsusaste

- Inglise keeles *level of detail* (*LOD*)
- Ruumiliste objektide renderdamisel kasutatakse erineva detailsusastmega mudeleid. See võimaldab vastavalt kaamera kaugusele valida keerulise või lihtsa mudeli.
- Kui me vaatame objekti kaugelt, ei näeks me detaile nagunii – seega pole neid mõtet joonistada. Kui liigume lähemale, oleks hea näha kõiki detaile.

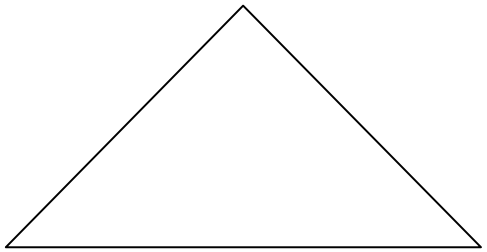
Detailsusaste

- Kaks sorti LODi: staatiline ja dünaamiline.
- Staatilise LOD puhul kasutatakse mitut erineva detailsusastmega mudelit, mille vahel siis joonistamisel vastavalt kaamera kaugusele valitakse.
- Dünaamilise LOD puhul kasutatakse spetsiaalseid algoritme, mis võimaldavad lennult objekte lihtsustada ja hulknurkade arvu vähendada.

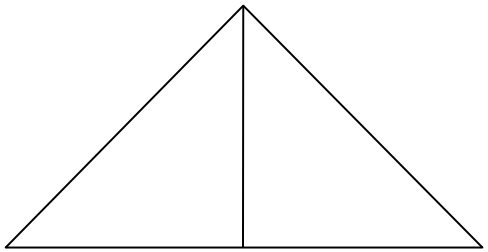
Kahend- ja neljandpuud

- Osutub, et peale kaheksandpuude saab kasutada ka kahend- ja neljandpuid.
- Mõlemat on võimalik kasutada maastiku erineva detailsusastmega versiooni kuvamisel.
- Olgu ette antud on maastiku kõrguskaart, mis seab koordinaatrastrile vastavusse antud punkti kõrguse merepinnast.

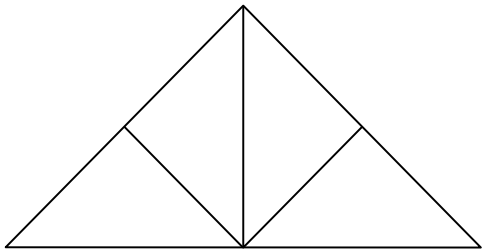
Kolmnurkade kahendpuu



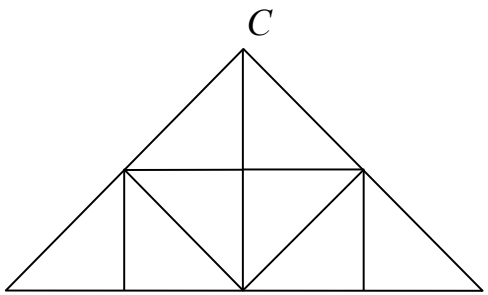
tase 0



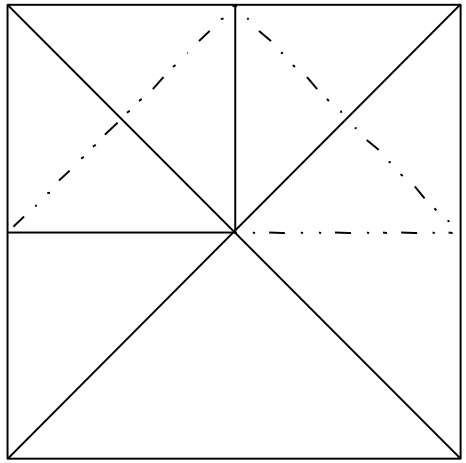
tase 1



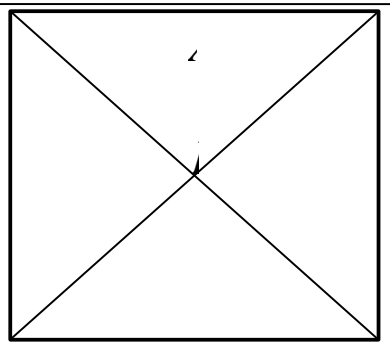
tase 2



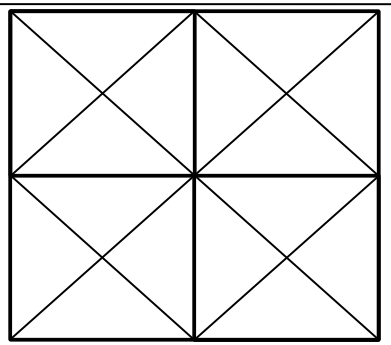
tase 3



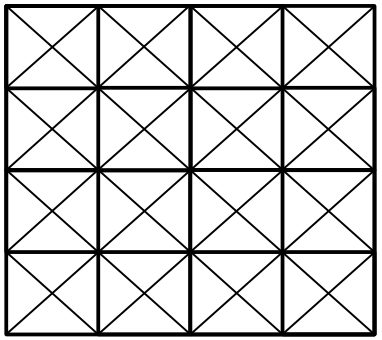
Ruutude neljandpuu



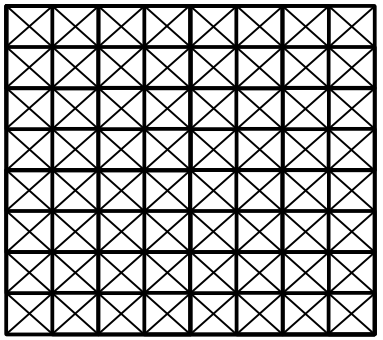
tase 0



tase 1



tase 2



tase 3

Kus me nüüd oleme?

- ~~Tippude transformeerimine & projitseerimine~~
- ~~Nähtamatu tahkude & tippude eemaldamine~~
- Polügonide rasteriseerimine
- Pikslite värvimine
- Pikslite ekraanile väljastamine