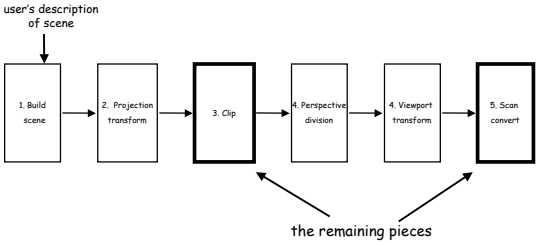
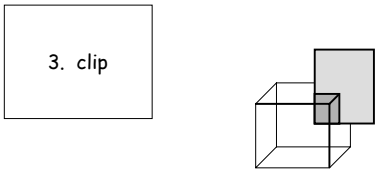
graphics pipeline



graphics pipeline 3

eliminate "outside" primitive



primitives

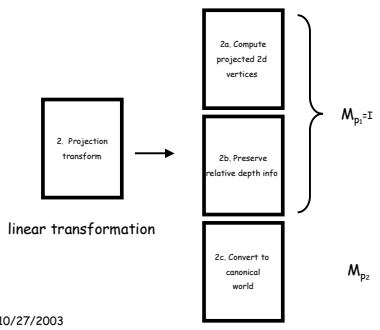
- vertices
- line segments
- polygons

when & how to clip

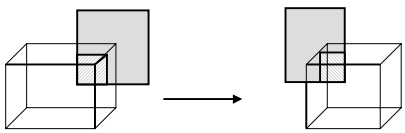
- pre-projection, mid-projection, post-projection
- 3d or 4d

let's do orthographic first

Orthographic: projection step simply converts to canonical world

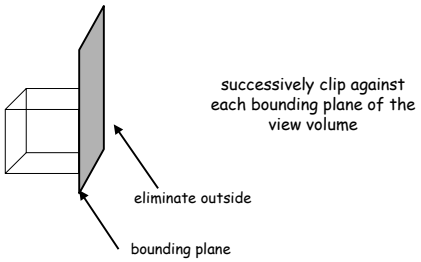


orthographic: convert world

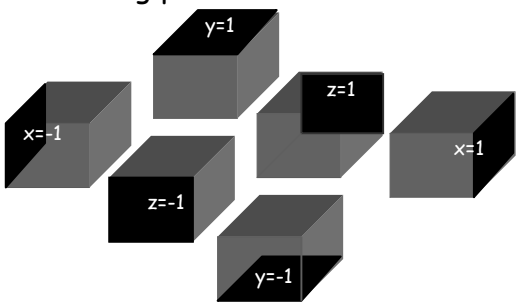


pre-projection clipping = post-projection clipping

3d clipping

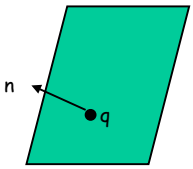


bounding planes of canonical world

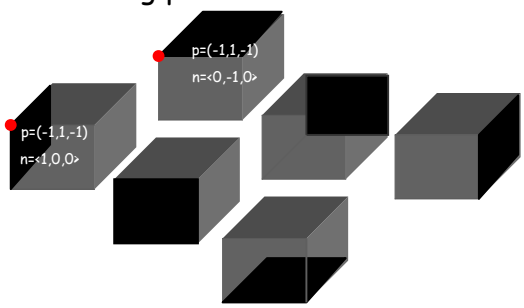


bounding plane description

1. point on plane
2. inward-pointing normal



bounding planes of canonical world



exercise: compute p & n for remaining planes

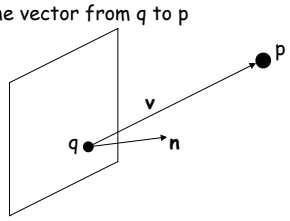
3d clipping

- vertex clipping
- line segment clipping
- polygon clipping

vertex clipping

p is in with respect to the clipping plane iff
n • **v** ≥ 0 where

- **n** is the inward facing normal
- **v** is the vector from q to p

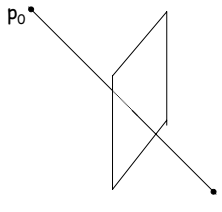


3d clipping

- vertex clipping
- **line segment clipping**
- polygon clipping

line segment clipping

use test for vertex clipping



Classify endpoint p_o & p_1

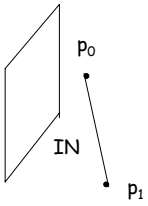
Case: p_o & p_1 in _____

Case: p_o & p_1 out_____

Case: p_o in & p_1 out _____

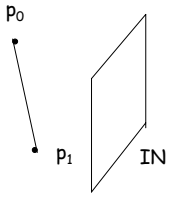
Case: p_o out & p_1 in _____

line segment clipping



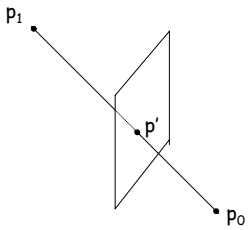
Case p_o & p_1 in:
return (p_0, p_1)

line segment clipping



Case p_o & p_1 out:
return null

line segment clipping



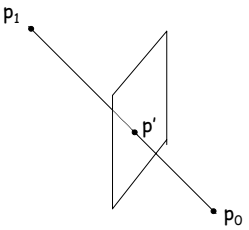
Case p_o in & p_1 out:
return (p_0,p')

Case: p_o out & p_1 in:
return (p',p_1)

do you know how to compute p'?

what should happen to w component?

line segment clipping



color at p':

we'll come back to this when we talk about color models

exercise 1

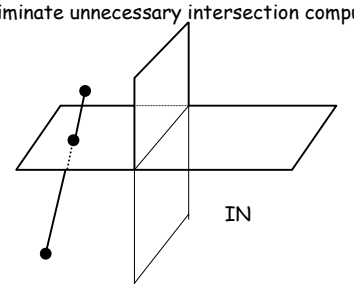
- clip the line with endpoints (-10,-10,-12) and (2,2,3) using the following order of bounding planes:
  - near
  - far
  - left
  - right
  - top
  - bottom
- show the endpoints at the beginning of each step
- how many intersection computations did you do? against which planes?

exercise 2

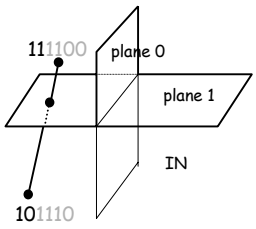
- clip the line with endpoints (-10,-2,0) and (10,-2,0) using the following order of bounding planes:
  - near
  - far
  - left
  - right
  - top
  - bottom
- show the endpoints at the beginning of each step
- how many intersection computations did you do? against which planes?

out-code optimization

eliminate unnecessary intersection computations



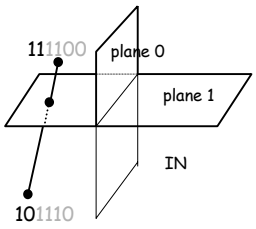
out-code optimization



endpoint p has out-code $b_0b_1...b_5$:

- b_i=0 if p is inside plane i
- b_i=1 else

out-code optimization

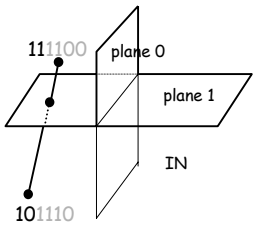


endpoint p has out-code $b_0b_1...b_5$:

- b_i=0 if p is inside plane i
- b_i=1 else

what does it mean if the i^{th} bit of both endpoints is 1?

out-code optimization



endpoint p has out-code $b_0b_1...b_5$:

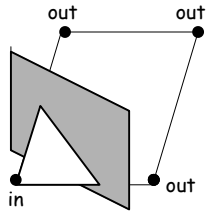
- b_i=0 if p is inside plane i
- b_i=1 else

what does it mean if the i^{th} bit of both endpoints is 0?

3d clipping

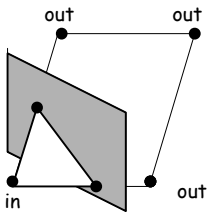
- vertex clipping
- line clipping
- **polygon clipping**

polygon clipping



1. classify vertices

polygon clipping



1. classify vertices
2. compute intersection points of intersecting edges

   &

   write out new polygon

polygon clipping

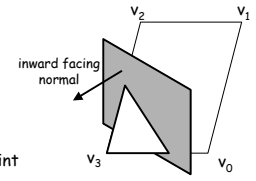
- if v_0 is in then write v_0
- for i=0…n-1
  - case v_i & v_{i+1} in: write v_{i+1}
  - case v_i & v_{i+1} out: do nothing
  - case v_i in and v_{i+1} out: write intersection point
  - case v_i out and v_{i+1} in: write intersection point and v_{i+1}

indices taken modulo n

example

if v_0 is in then write v_0
for i=0…3

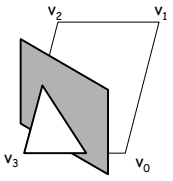
- case v_i & v_{i+1} in:
  write v_{i+1}
- case v_i & v_{i+1} out:
  do nothing
- case v_i in and v_{i+1} out:
  write intersection point
- case v_i out and v_{i+1} in:
  write intersection point and v_{i+1}



example

if v_0 is in then write v_0
for i=0…3

- case v_i & v_{i+1} in:
  write v_{i+1}
- case v_i & v_{i+1} out:
  do nothing
- case v_i in and v_{i+1} out:
  write intersection point
- case v_i out and v_{i+1} in:
  write intersection point and v_{i+1}

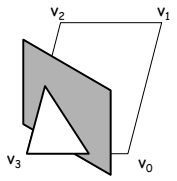


do nothing

example

i=0

- case v_i & v_{i+1} in:
  write v_{i+1}
- case v_i & v_{i+1} out:
  do nothing
- case v_i in and v_{i+1} out:
  write intersection point
- case v_i out and v_{i+1} in:
  write intersection point
  and v_{i+1}

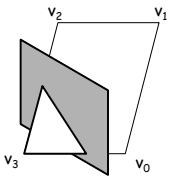


do nothing

example

i=1

- case v_i & v_{i+1} in:
  write v_{i+1}
- case v_i & v_{i+1} out:
  do nothing
- case v_i in and v_{i+1} out:
  write intersection point
- case v_i out and v_{i+1} in:
  write intersection point
  and v_{i+1}

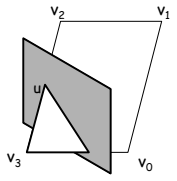


do nothing

example

i=2

- case v_i & v_{i+1} in:
  write v_{i+1}
- case v_i & v_{i+1} out:
  do nothing
- case v_i in and v_{i+1} out:
  write intersection point
- case v_i out and v_{i+1} in:
  write intersection point
  and v_{i+1}



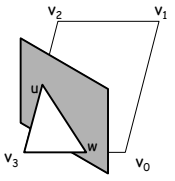
write u, v_3

example

output: u, v_3

i=3

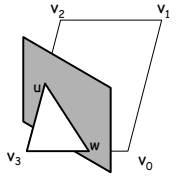
- case v_i & v_{i+1} in:
  write v_{i+1}
- case v_i & v_{i+1} out:
  do nothing
- case v_i in and v_{i+1} out:
  write intersection point
- case v_i out and v_{i+1} in:
  write intersection point
  and v_{i+1}



write w

example

output: u, v_3, w



exercise 3

- clip the triangle with vertices (-10,2,4), (5,2,-4), and (-10,-2,0) using the following order of bounding planes
  - near (team 1)
  - far (team 2)
  - left (team 3)
  - right (team 1)
  - top (team 2)
  - bottom (team 3)
- each team should write the vertices after each of its steps on the board

when & how to clip

- pre-projection, mid-projection, post-projection
- 3d or 4d

let's do orthographic first
now for perspective

we need to wave our hands a little

hyperbolic interpolation

similar to linear interpolation ...

graphics pipeline