Advanced Computer Graphics

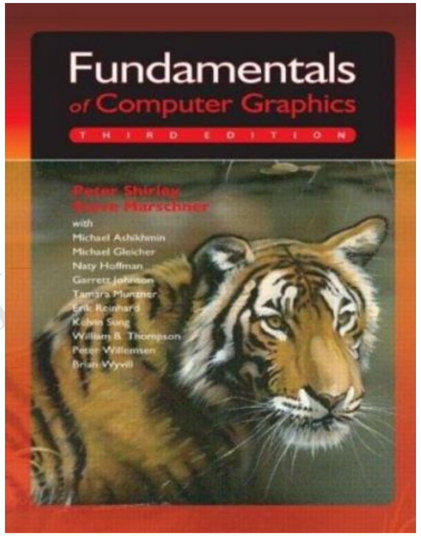
Lecture-06 Hidden face removal

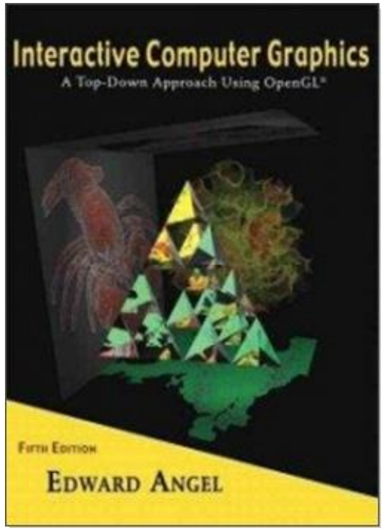
Tzung-Han Lin
National Taiwan University of Science and Technology
Graduate Institute of Color and Illumination Technology
e-mail: thl@mail.ntust.edu.tw

TAIWAN TECH National Taiwan University of Science and Technology
CIT 色彩與照明科技研究所 Graduate Institute of Color and Illumination Technology

Content in textbook

- Fundamentals of Computer Graphics, Chapter 8
- Interactive Computer Graphics, 5th edition, Chapter 7





NTUST. CIIC.

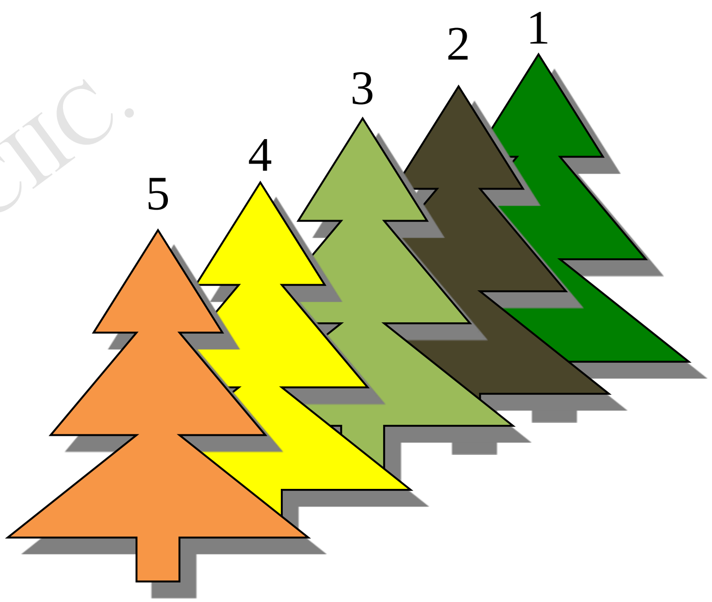
Outline

- Priority tree (brief introduction)
  - Back face culling
- Scan line algorithm
- Z-buffering (depth buffering)*

NTUST. CIIC.

Priority tree

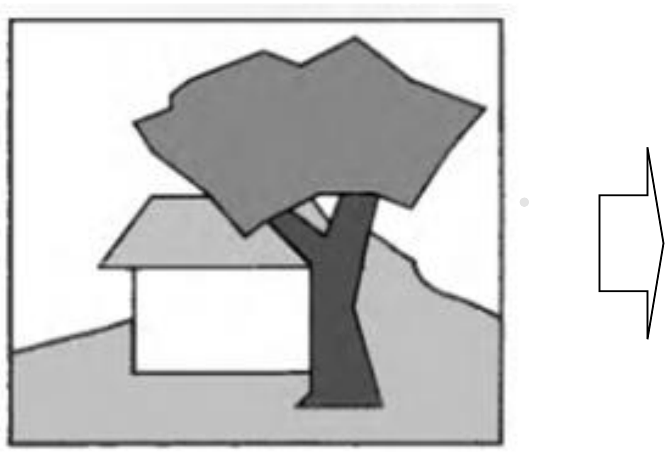
- Priority tree (BSP-binary space partition)
- In 2D, the drawing priority is higher if the object is further. For example, the priority in following figure will be 1→2→3→4→5.

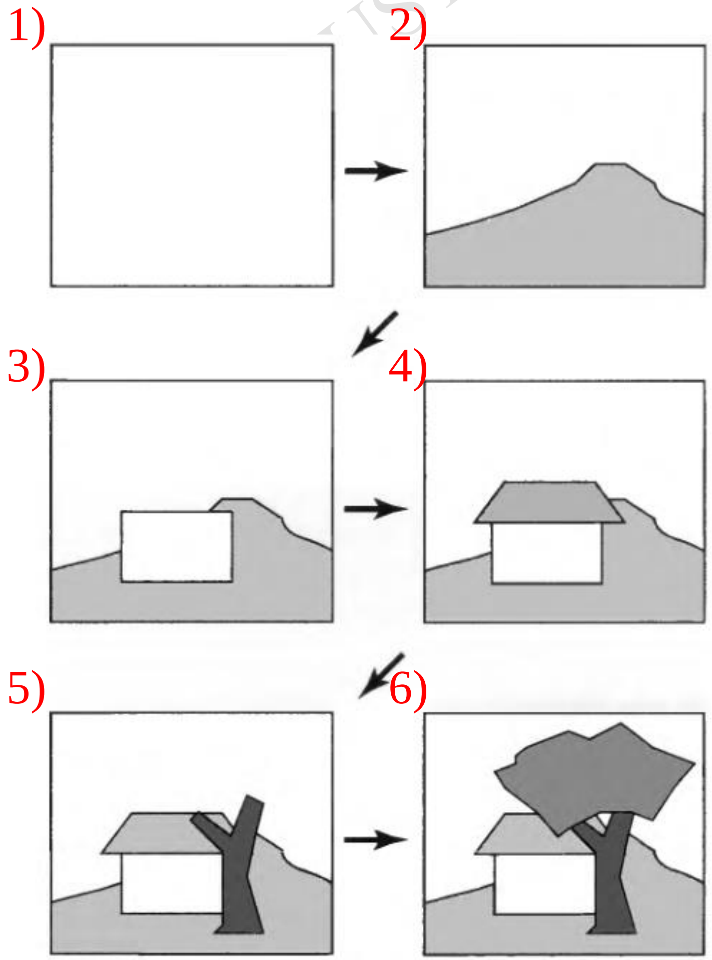


NTUST. CIIC.

Priority tree

- For example

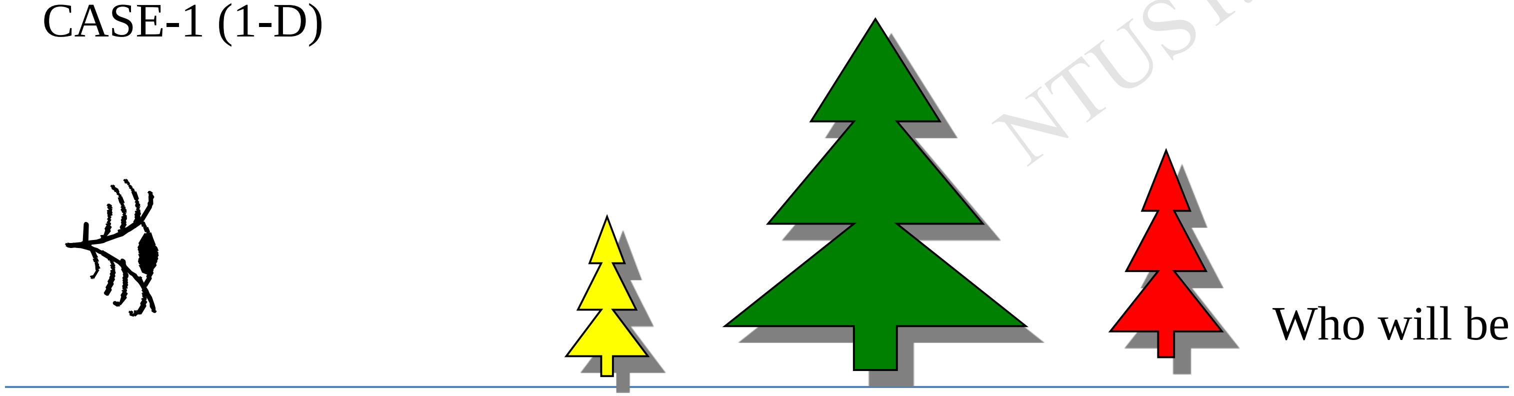




NTUST. CIIC.

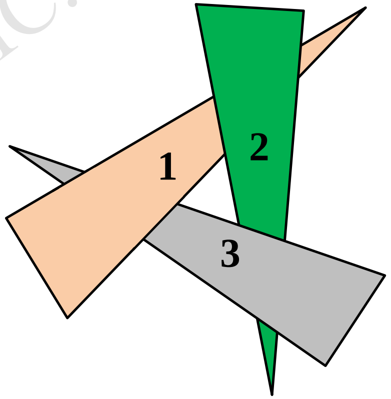
Priority tree

CASE-1 (1-D)



Who will be ?

CASE-2 (2-D)

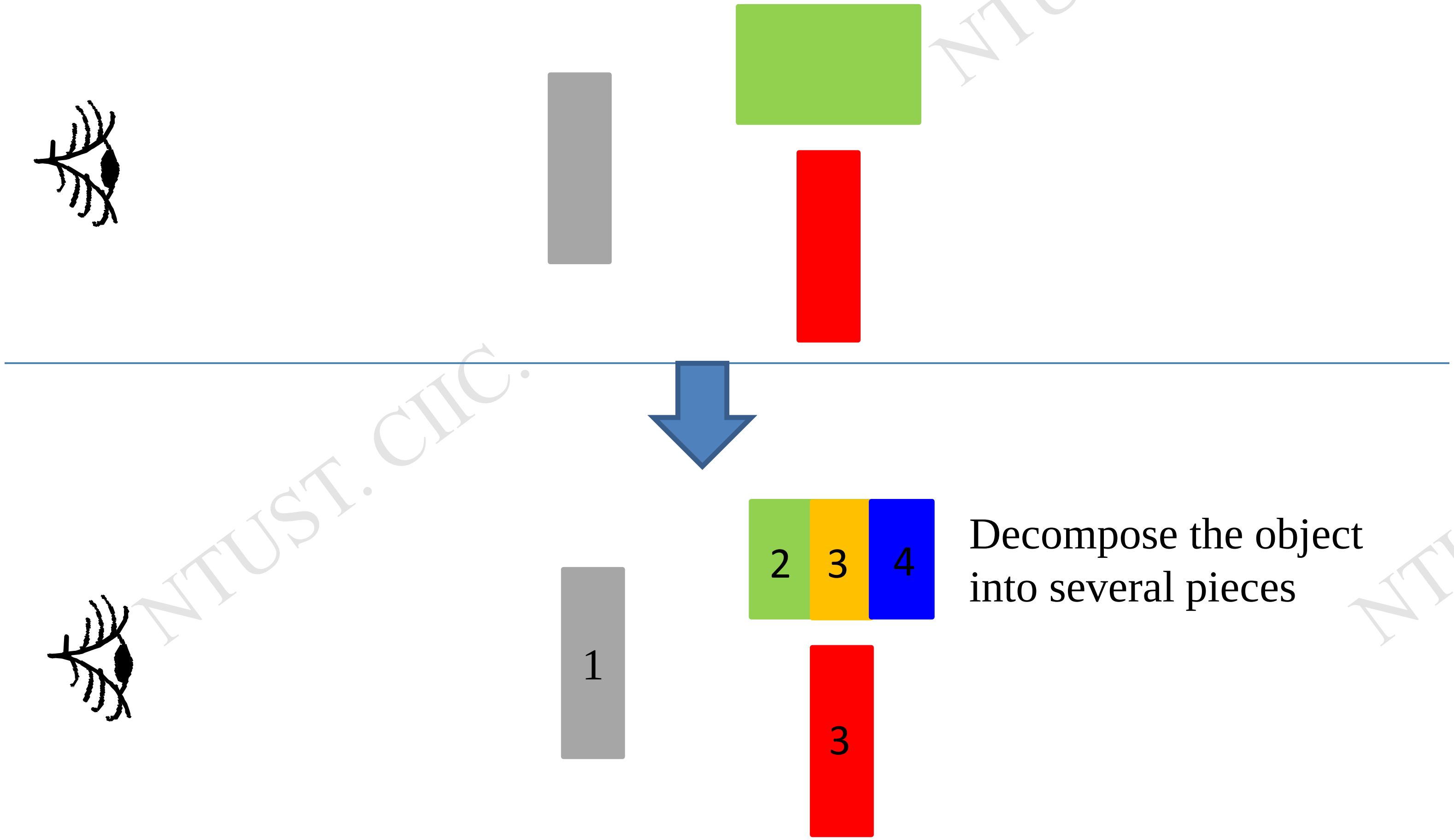


Who will be further than other else?
What priority it should be?

NTUST. CIIC.

Priority tree

- Priority tree (BSP-binary space partition)

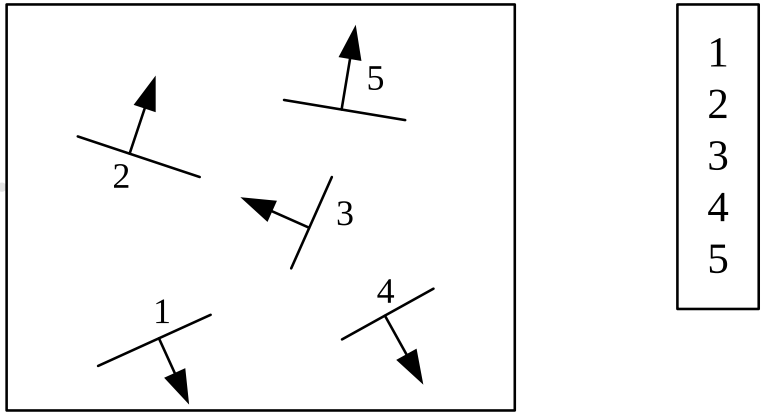


Decompose the object into several pieces

NTUST. CIIC.

Priority tree (build a binary tree)

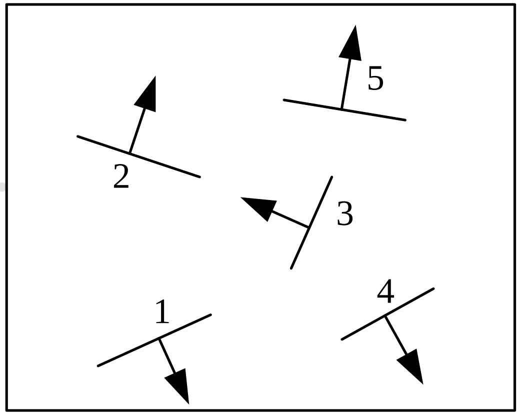
- BSP (binary space partition)
  - Build a tree (randomly)
  - Given a viewing direction
  - Traveling the tree according to the viewing direction (to draw all polygons)



NTUST. CIIC.

Priority tree (build a binary tree)

- BSP (binary space partition): Build a tree
  - First of all, you need to know all the normal directions of polygons.
  - Randomly select the "root", says a polygon, of the tree, then partition all polygon recursively.

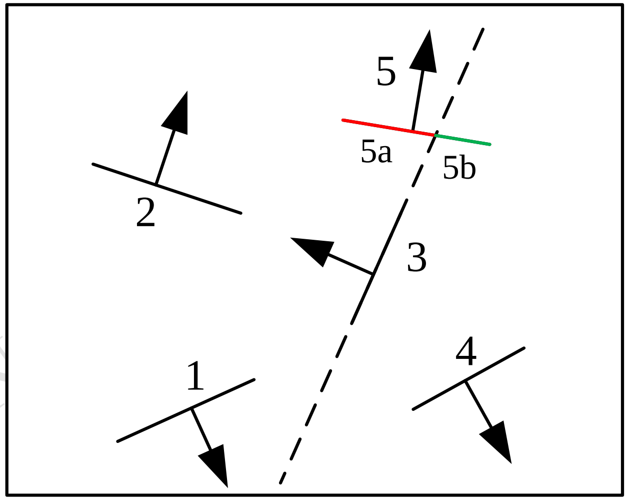


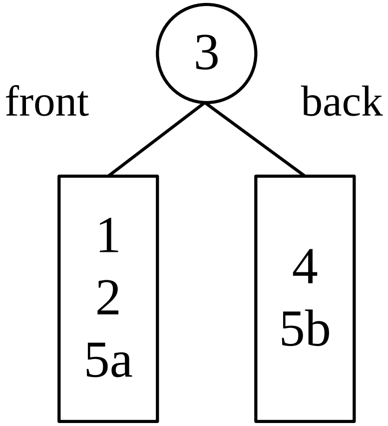
1
2
3
4
5

NTUST. CIIC.

Priority tree (build a binary tree)

- BSP (binary space partition): Build a tree—Example
  - Step-1: A polygon “3” is selected to be a root.
  - Step-2: Partition all the rest polygon into two categories, say front and back.

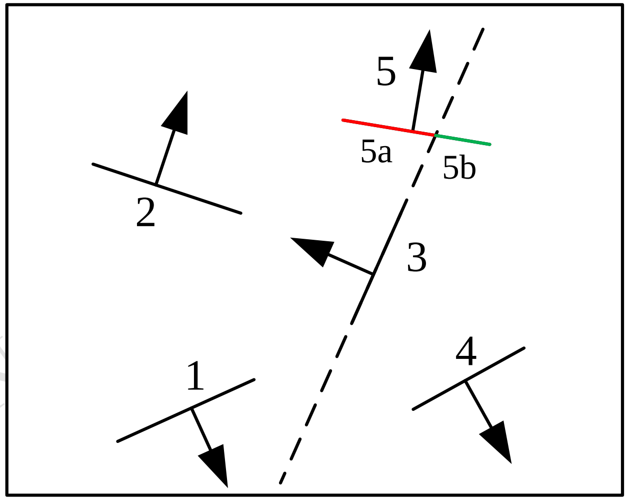


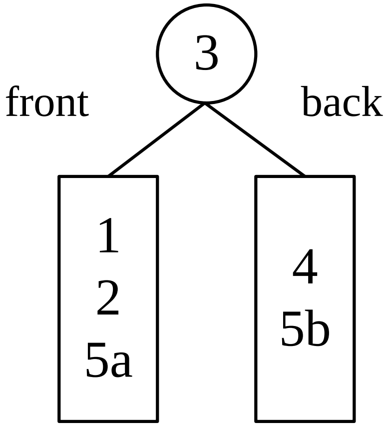


NTUST. CIIC.

Priority tree (build a binary tree)

- BSP (binary space partition): Build a tree—Example
  - You may find polygon "5" is in both front and back space. In this case, it is needed to be splitted into two pieces, says "5a" and "5 b".

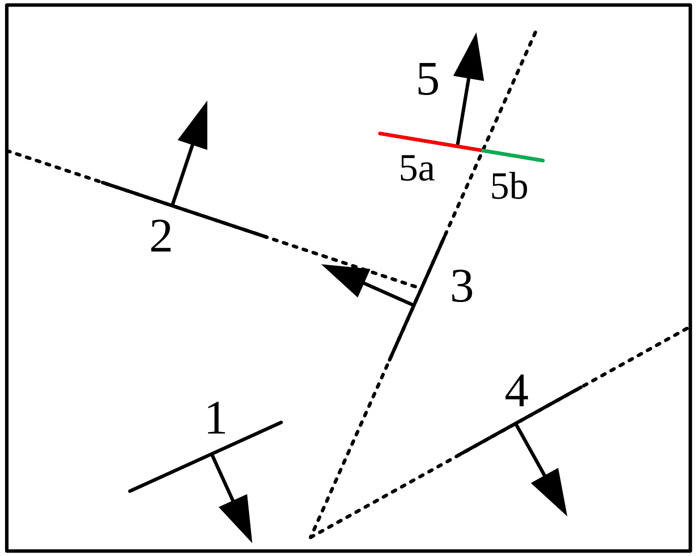


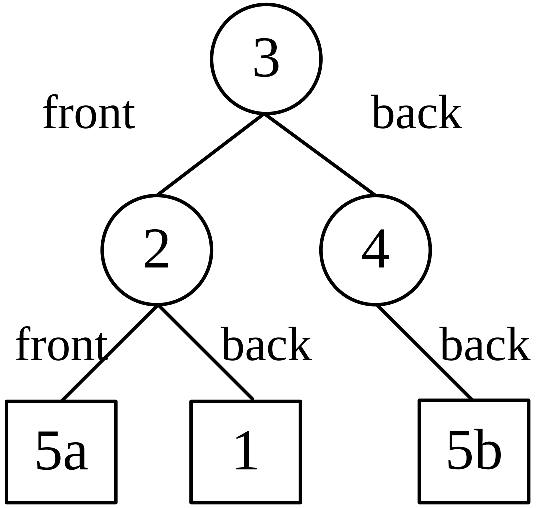


NTUST.CIIC.

Priority tree (build a binary tree)

- BSP (binary space partition): Build a tree—Example
  - Step-3: Once again, in each space, another sub-root “2” is selected to partition the sub-space (in front of “3”).

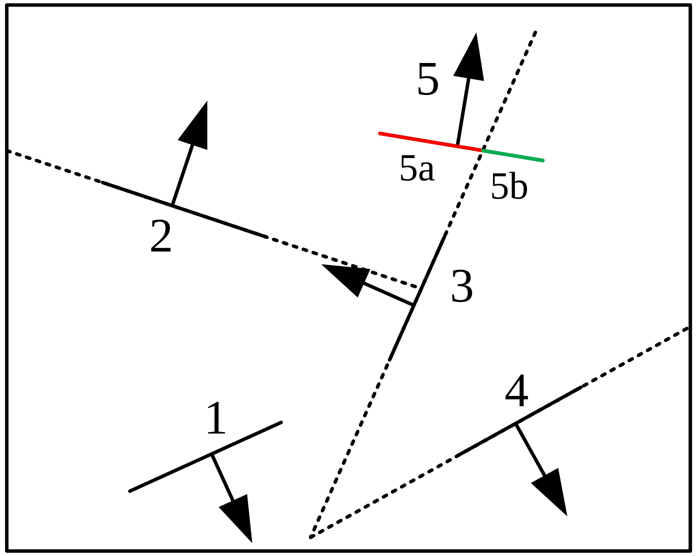


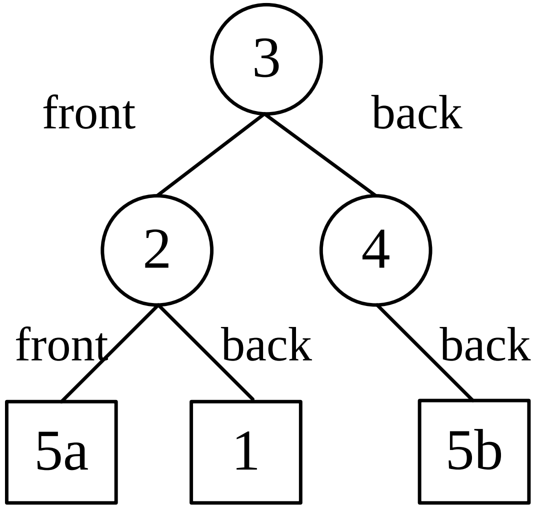


NTUST.CIIC.

Priority tree (build a binary tree)

- BSP (binary space partition): Build a tree—Example
  - Step-4: Recursively partition all sub-space. Finally, you may have a binary tree, whose nodes represent the polygon in space.

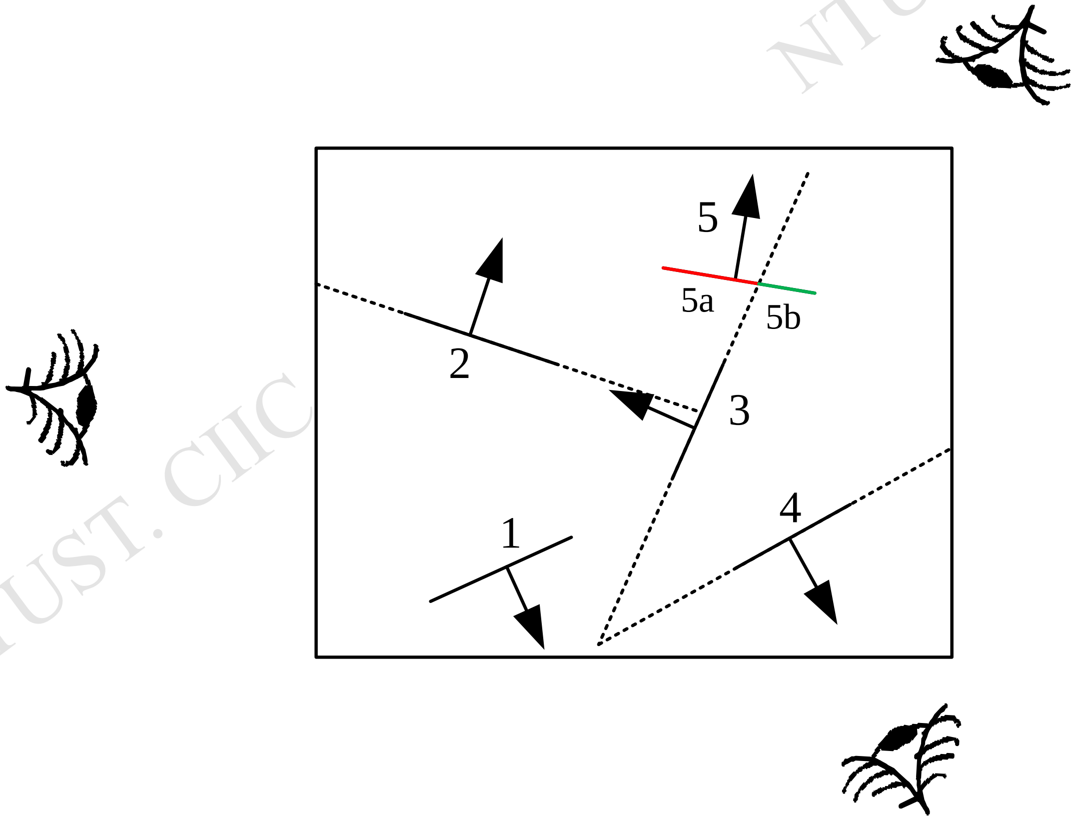




NTUST. CIIC.

Priority tree (go through tree by view dir.)

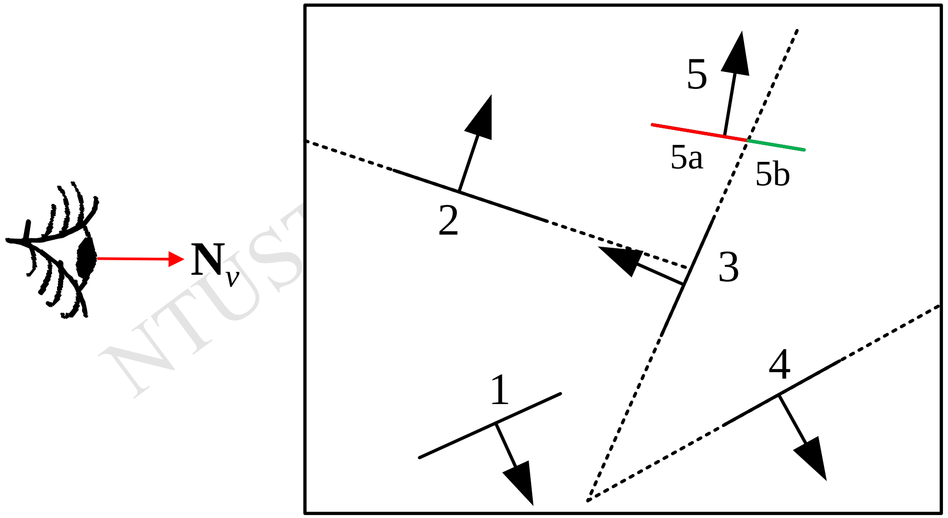
- BSP Tree: Given a viewing direction—Example

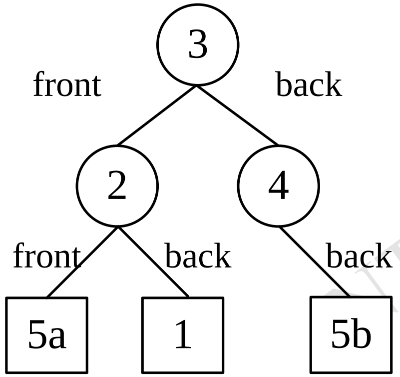


NTUST. CIIC.

Priority tree (go through tree by view dir.)

- BSP Tree: Traveling the tree according to the viewing direction (to draw all polygons)—Example.
  - According the draw policy, the further polygon has higher priority.

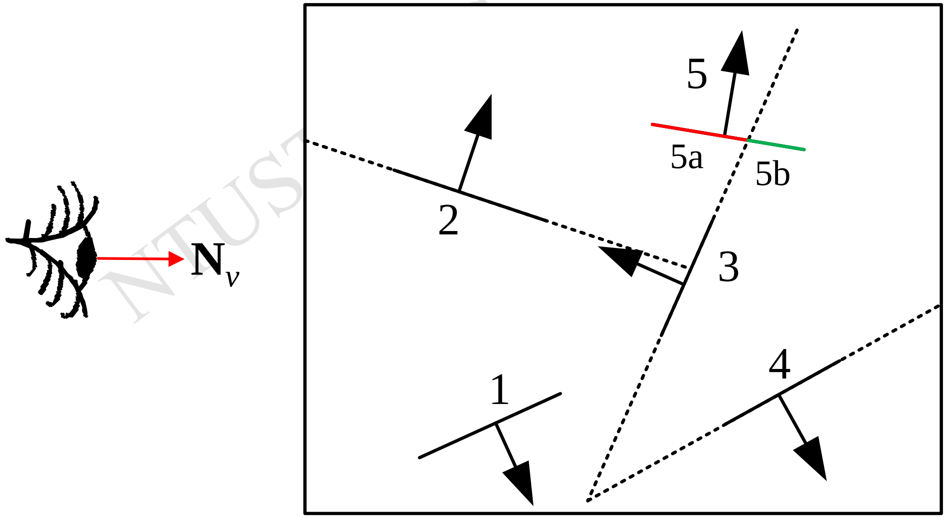


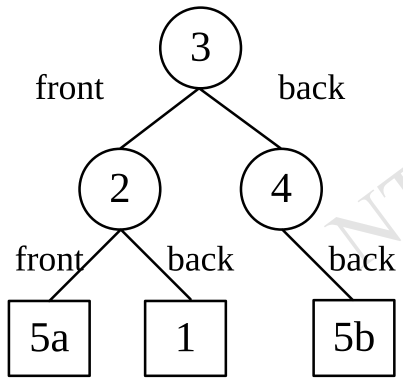


NTUST.CIIC.

Priority tree (go through tree by view dir.)

- BSP Tree: Traveling the tree according to the viewing direction (to draw all polygons)—Example-1.
  - Compare viewing vector with root “3”: since Nv·N3<0 → travel to back-space “4”, then compare with “4” Nv·N4>0 →travel to front-space “nothing”…
  - So we have 4→5b→3→5a→2→1 as the drawing sequence.

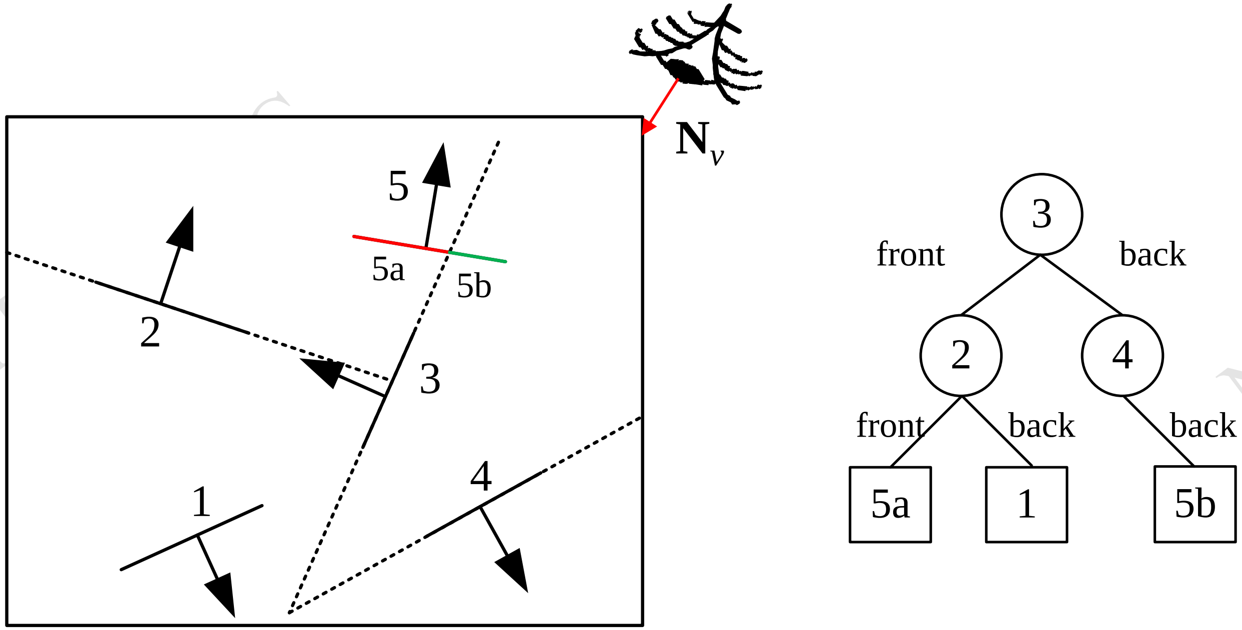




NTUST.CIIC.

Priority tree (go through tree by view dir.)

- BSP Tree: Traveling the tree according to the viewing direction (to draw all polygons)—Example-2.
  - Compare viewing direction with root “3”, since Nv·N3>0 →travel to front-space, then compare with “2”, since Nv·N2<0 →travel to its back..
  - Finally, we have 1→2→5a→3→4→5b.



NTUST.CIIC.

Priority tree

Another statement (wiki):

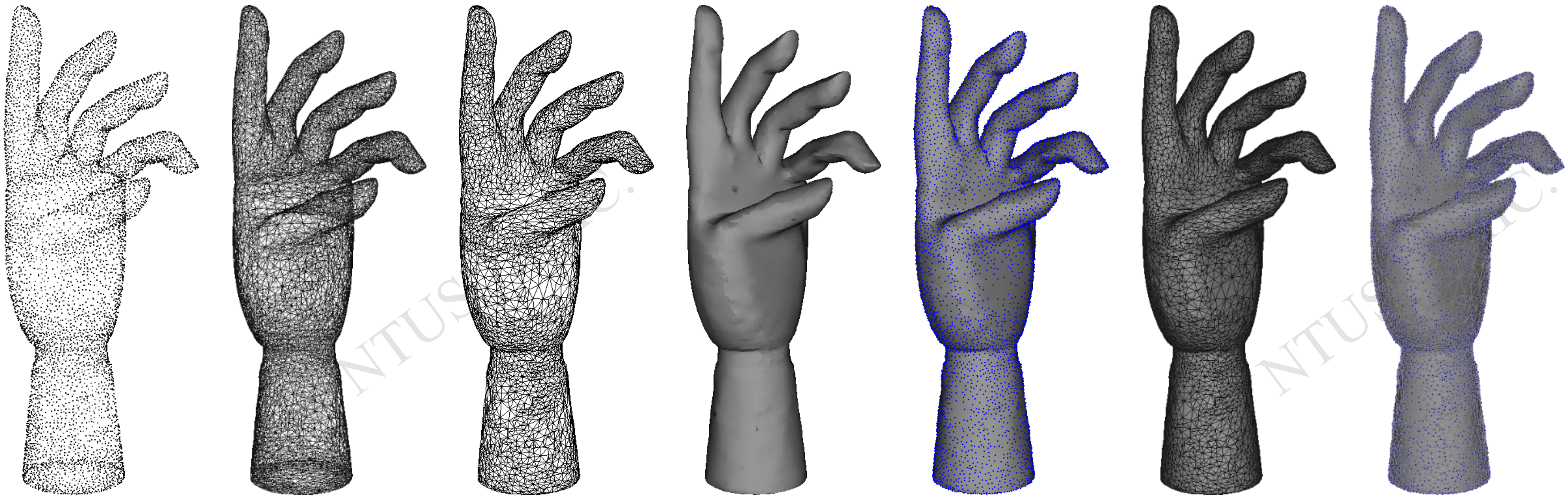
- In computer science, binary space partitioning (BSP) is a method for recursively subdividing a space into convex sets by hyperplanes. This subdivision gives rise to a representation of objects within the space by means of a tree data structure known as a BSP tree.
- Binary space partitioning was developed in the context of 3D computer graphics, where the structure of a BSP tree allows spatial information about the objects in a scene that is useful in rendering, such as their ordering from front-to-back with respect to a viewer at a given location, to be accessed rapidly. Other applications include performing geometrical operations with shapes (constructive solid geometry) in CAD, collision detection in robotics and 3-D video games, ray tracing and other computer applications that involve handling of complex spatial scenes.

https://en.wikipedia.org/wiki/Binary_space_partitioning

NTUST. CIIC.

Back face culling

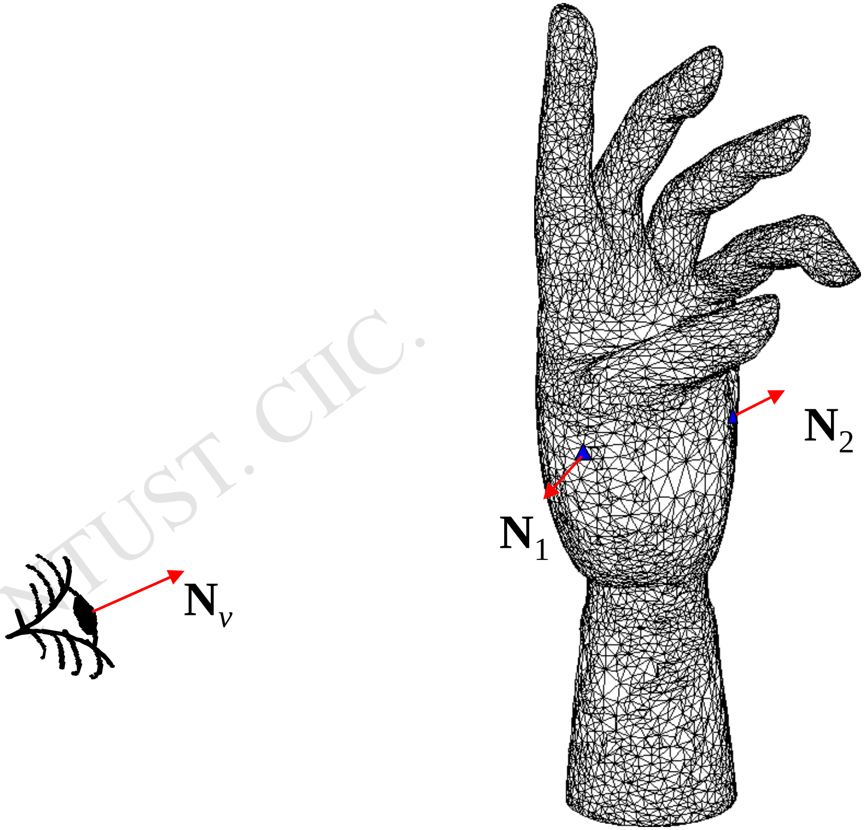
- "Back-face culling" is a rule to determine whether a polygon of a graphical object is visible.



NTUST.CIIC.

Back face culling

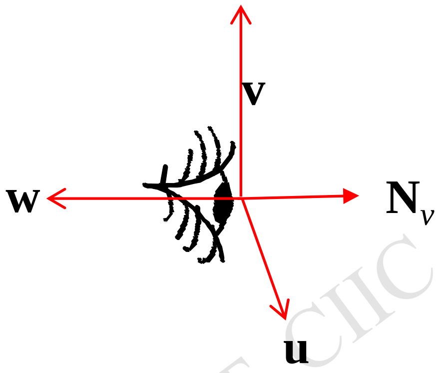
- Nv·N1<0 : meet the face's front (visible)
- Nv·N2>0 : not meet face's front (invisible or not?)



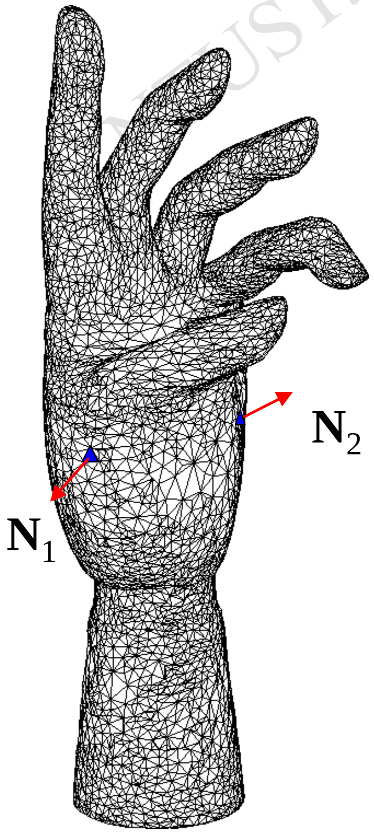
NTUST. CIIC.

Back face culling (cull face)

- Simply judge the “z” value of each face in “camera coordinate”



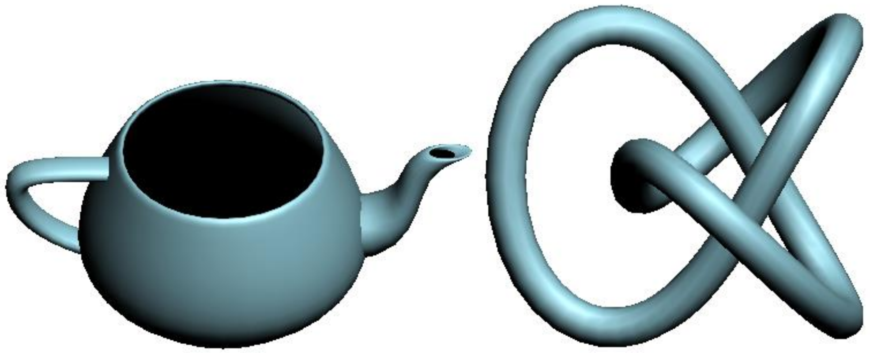
$N_1[2]>=0$ Draw face
$N_2[2]<0$ Do NOT draw



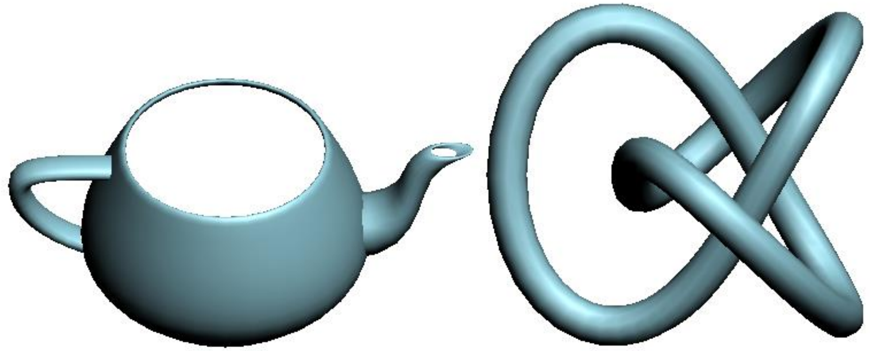
NTUST. CIIC.

Back face culling—Example

Without back face culling



Back face culling is enable

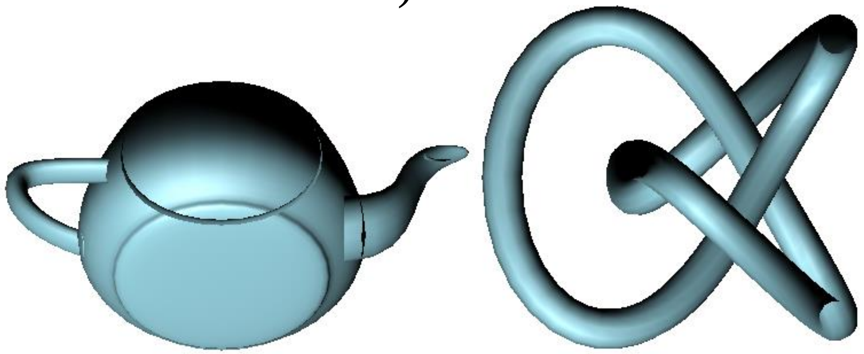


NTUST. CIIC.

Back face culling—Example

Without back face culling
(Normal vectors are inward)

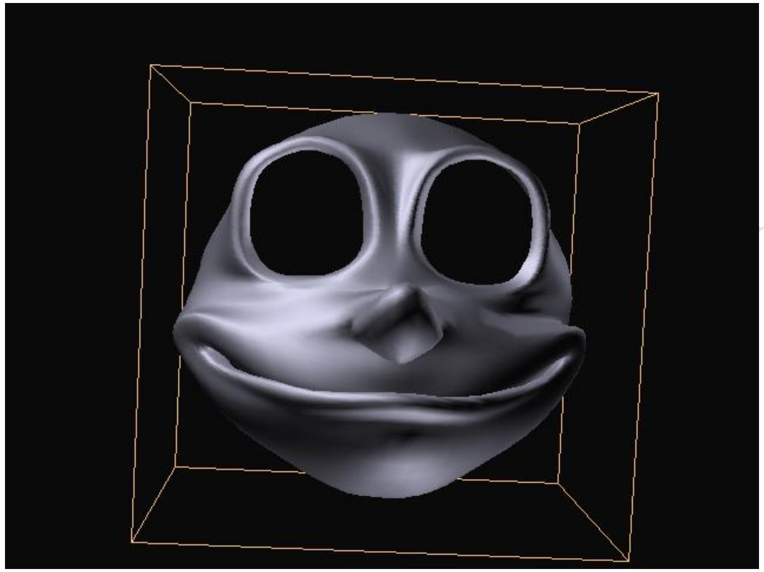
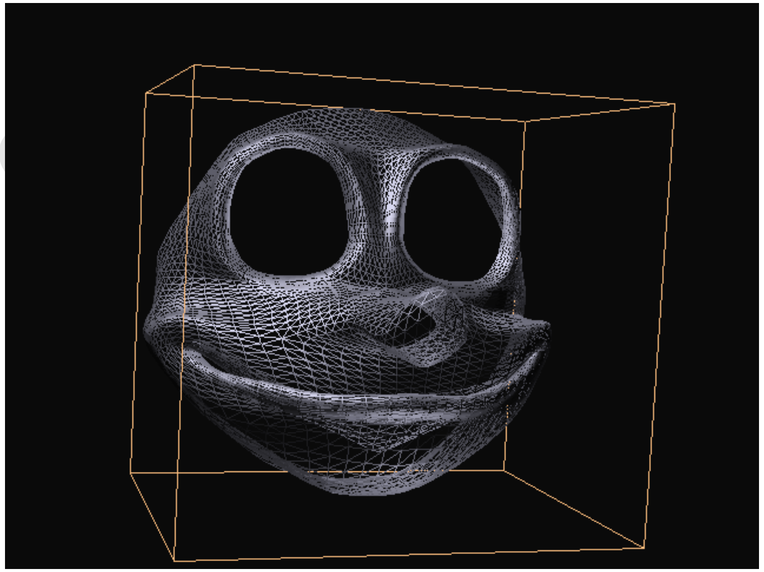
Back face culling is enable
(Normal vectors are inward)



NTUST. CIIC.

Back face culling—Example

- In openGL
  - Simply enable GL_CULL by glEnable.
  - Only affect the shape which is drawn under ”polygon (GL_POLYGON)” (Neither under point/GL_POINTS nor line/GL_LINE..)


NTUST. CIIC.
NTUST. CIIC.

Scan line and Z buffering algorithms?

- When do we need scan line and Z buffering algorithm?
  - Scan line can perform under very low "memory resource ", and it is able to sequentially output data.
  - Z buffering performs efficiently with specific array data, however it costs much "memory resource".
- In modern computer graphics, Z buffering is the popular solution to deal with 3D problem.

NTUST. CIIC.
NTUST. CIIC.
NTUST. CIIC.
NTUST. CIIC.

Scan line algorithm

- Two main purposes
  - Polygon filling
  - Combine with shading

- Further applications
  - 3D model slicing
  - Mesh to voxel

NTUST. CIIC.

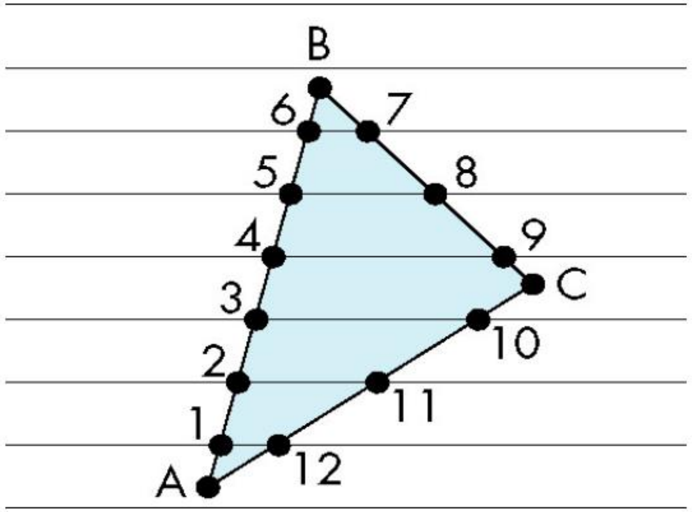
Scan line algorithm

- Advantages
  - Takes advantage of coherence resulting in fast algorithm
  - Does not require as much storage as depth buffer
- Disadvantages
  - More complex algorithm
  - Requires all polygons sent to CPU before drawing

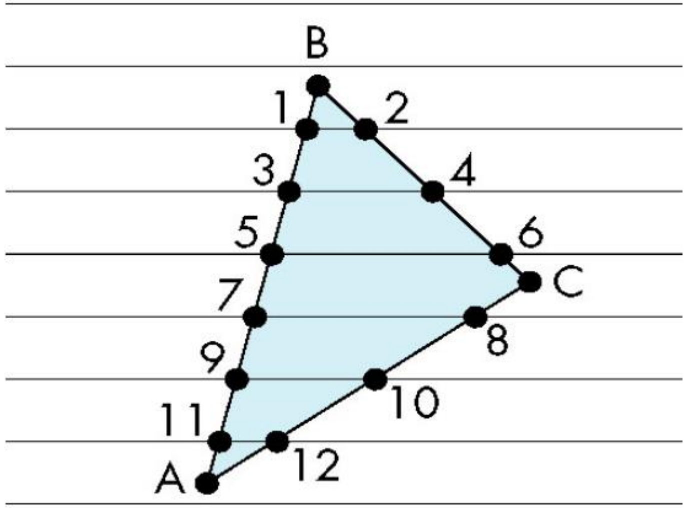
NTUST. CIIC.
NTUST. CIIC.
NTUST. CIIC.
NTUST. CIIC.

Scan line algorithm

- Main purpose
- Polygon filling (for simple polygon), can be integrated with shading algorithm



Vertex order
(process in vertex list)

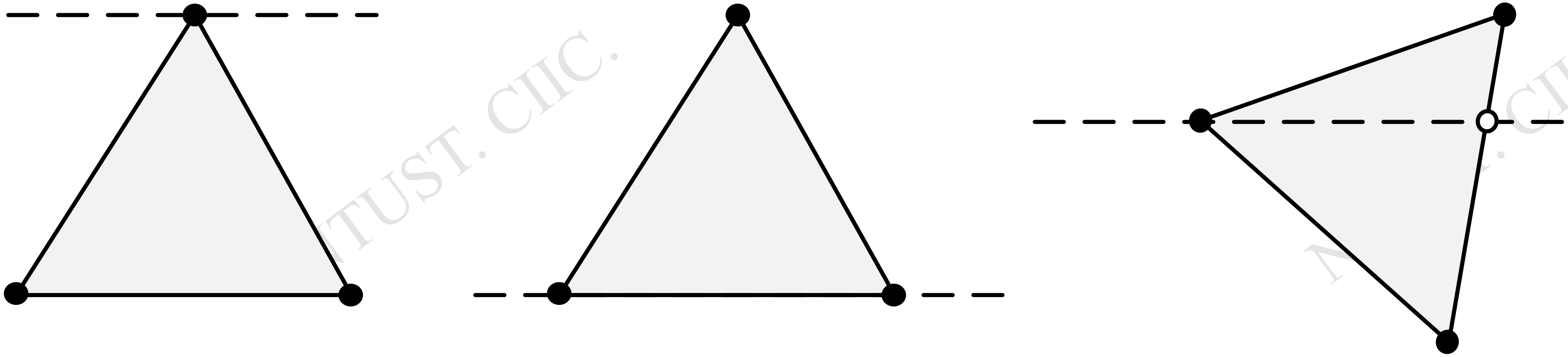


Desired order

NTUST. CIIC.

Scan line algorithm

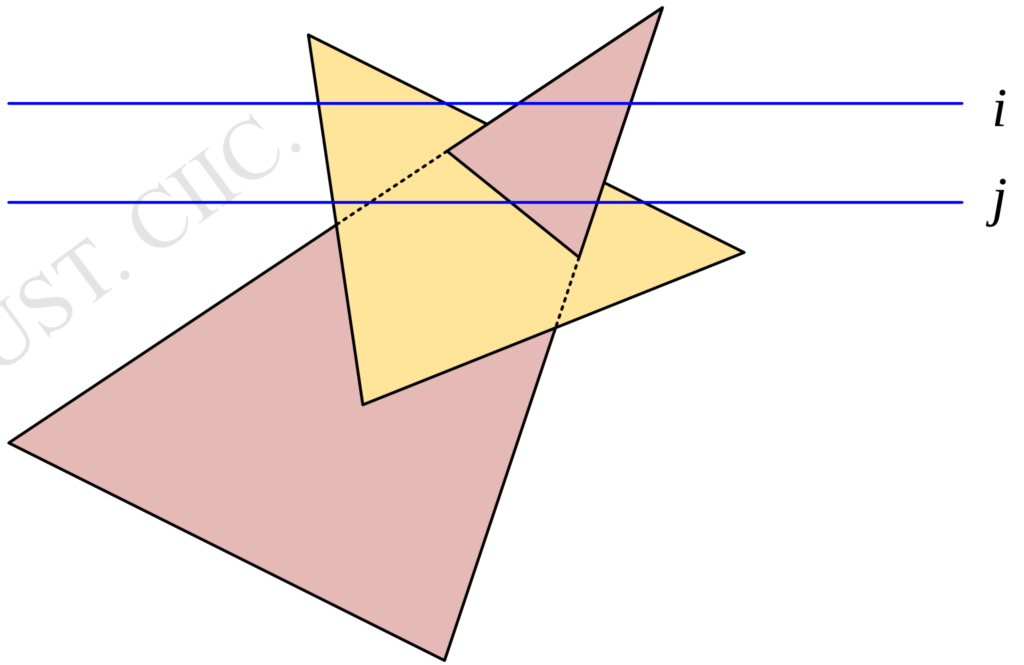
- Main purpose-1
- Polygon filling in constant color (for simple polygon), can be integrated with shading algorithm—cont.
  - “Special cases” v.s. “opportunistic”



NTUST. CIIC.

Scan line algorithm

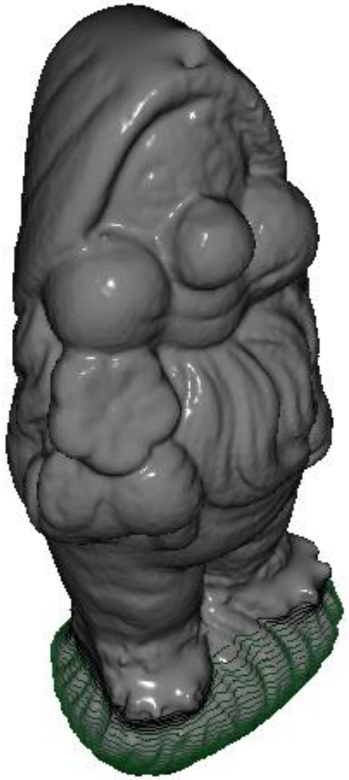
- Main purpose-2
- Scan line i: no need for depth comparison, can only be in one polygon
- Scan line j: need depth comparison only when in more than one polygon.



NTUST. CIIC.

Scan line algorithm

- Application
- 3D model slicing—Example

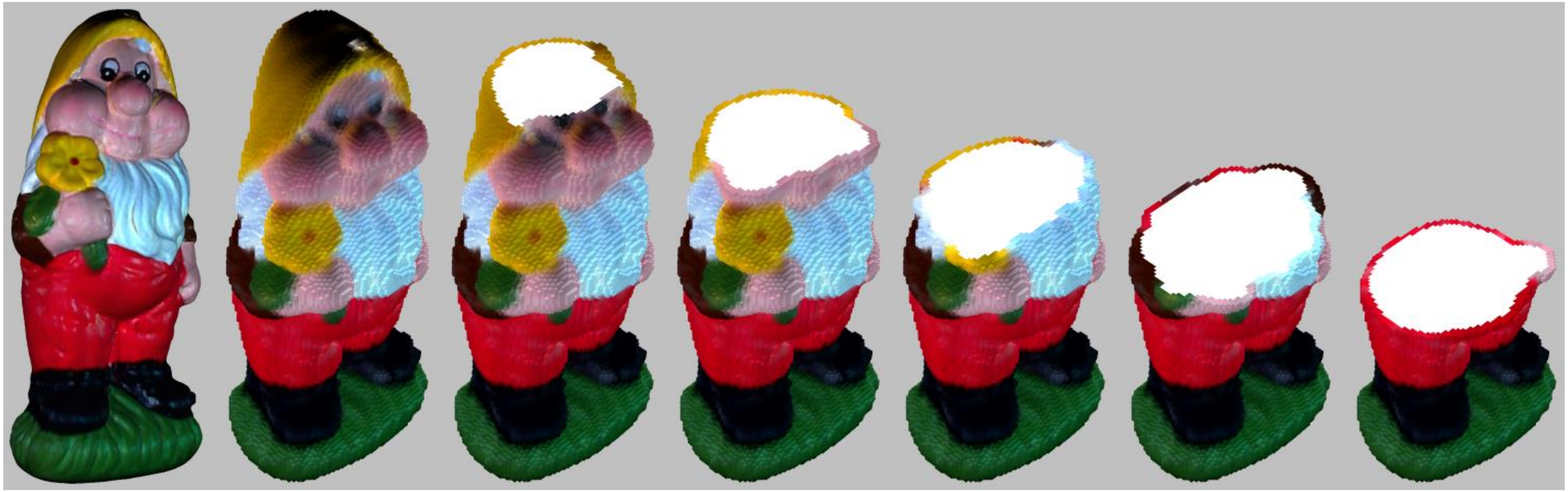


NTUST. CIIC.
NTUST. CIIC.

Scan line algorithm

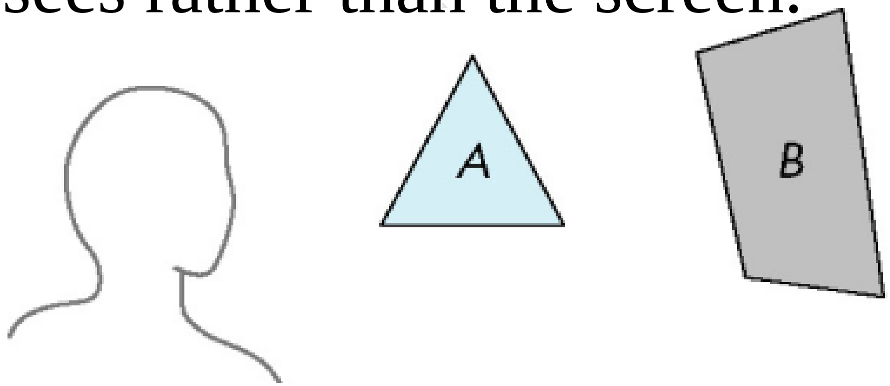
- Application
- Mesh to voxel—Example

NTUST. CIIC.
NTUST. CIIC.

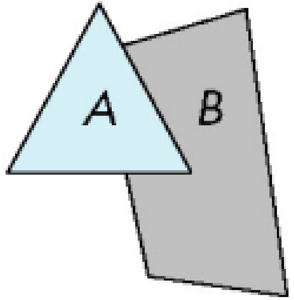


Z-buffering (depth buffering)

- Z-buffering works by testing pixel depth and comparing the current position, z value, with stored data in a buffer.
- The buffer is called a “z-buffer” that holds information about each pixel's last “z” value. The pixel in the closer position to the viewer is the one to be displayed, just as the person in front of the television is what the viewer sees rather than the screen.



B behind A as seen by viewer



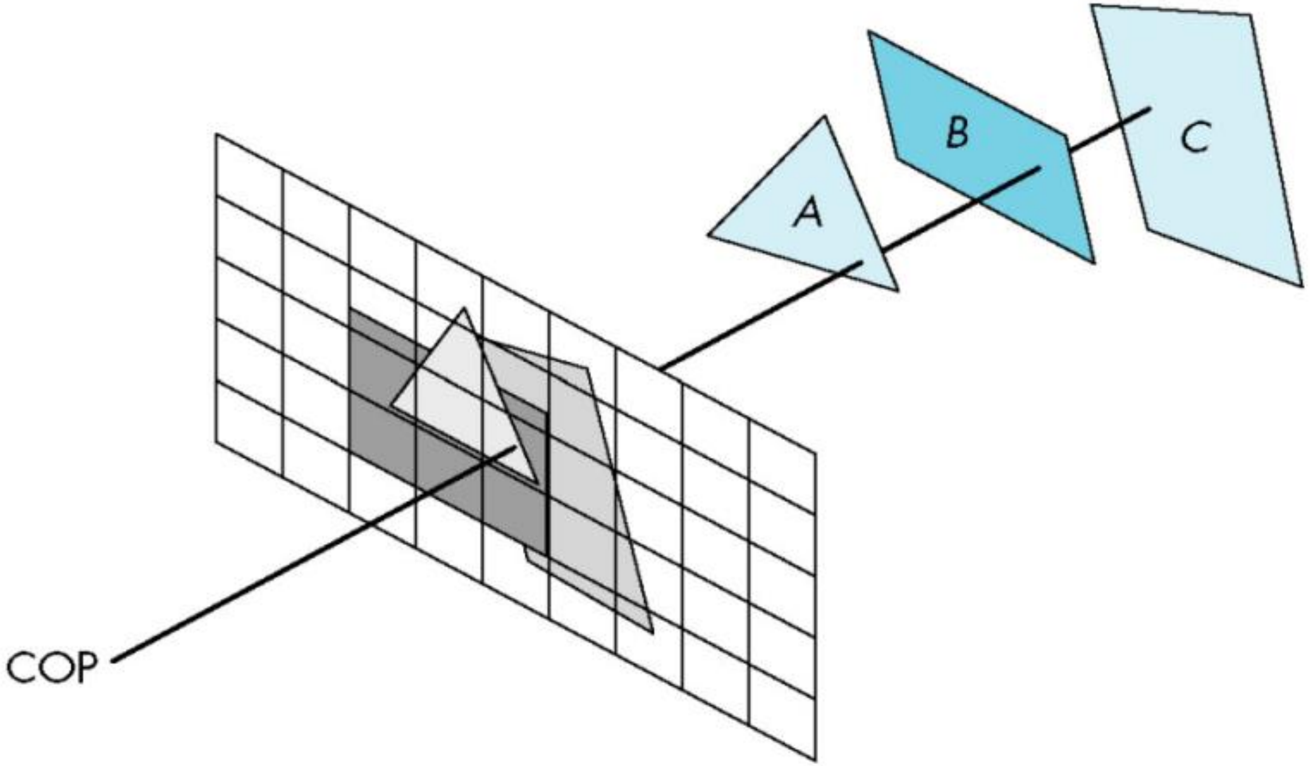
Fill B then A

Picture from book: interactive computer graphics

Z-buffering (depth buffering)

- In short, sort all “z values” for each pixel.

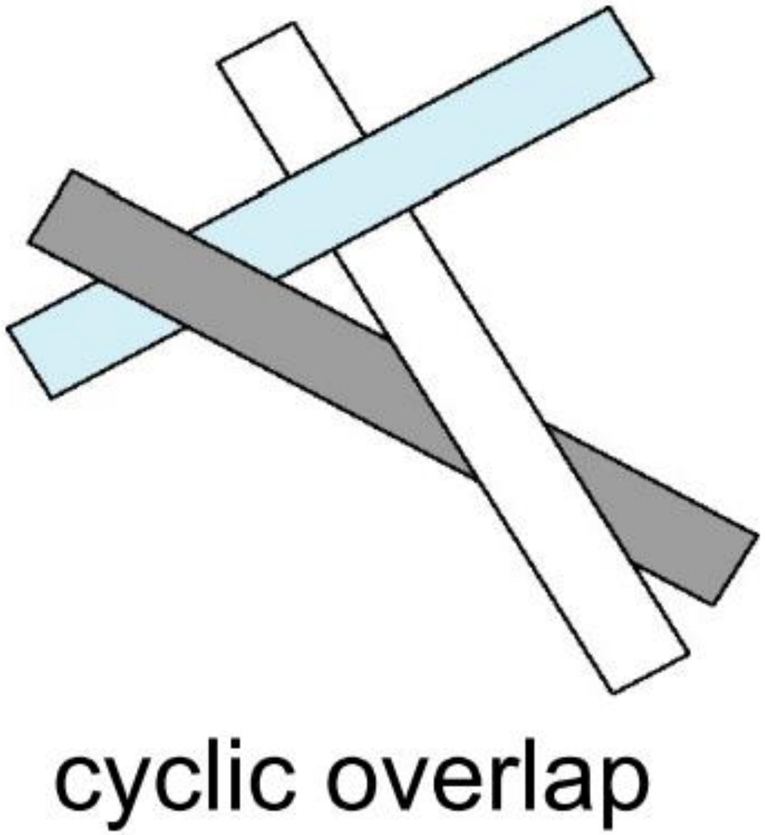
NTUST. CIIC.
NTUST. CIIC.
NTUST. CIIC.

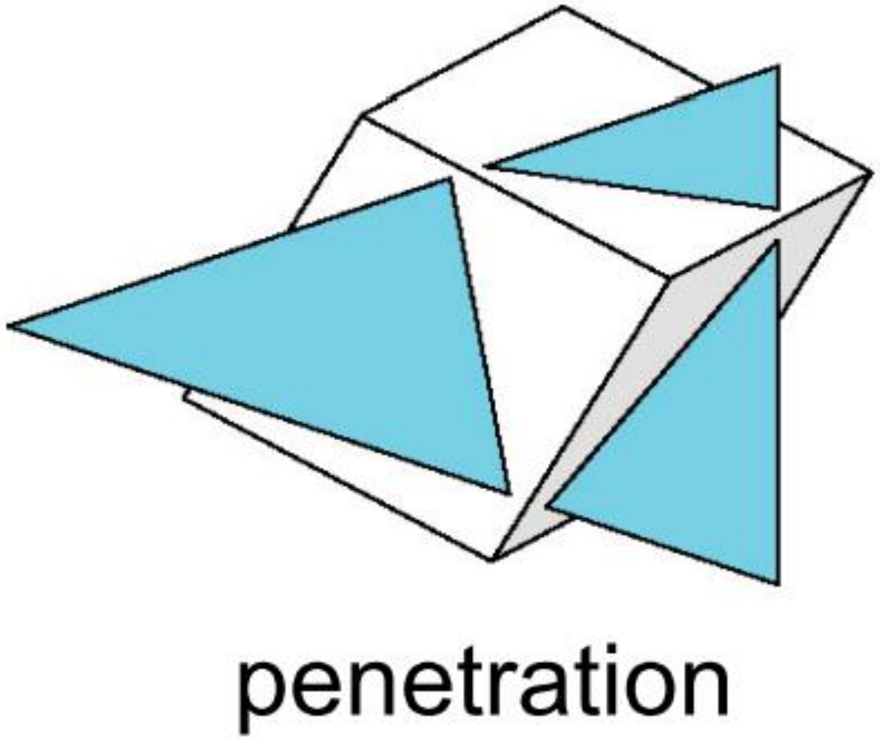


Picture from book: interactive computer graphics

Z-buffering (depth buffering)

- Rendering result—example





Picture from book: interactive computer graphics

Z-buffering (depth buffering)

- Data and image—example

Depth buffer (value)

∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞

1	∞	∞	∞
1	3	∞	∞
1	3	5	∞
1	3	5	7

1	∞	∞	1
1	3	3	1
1	3	3	1
1	3	3	1

∞	∞	∞	1
∞	∞	3	1
∞	5	3	1
7	5	3	1

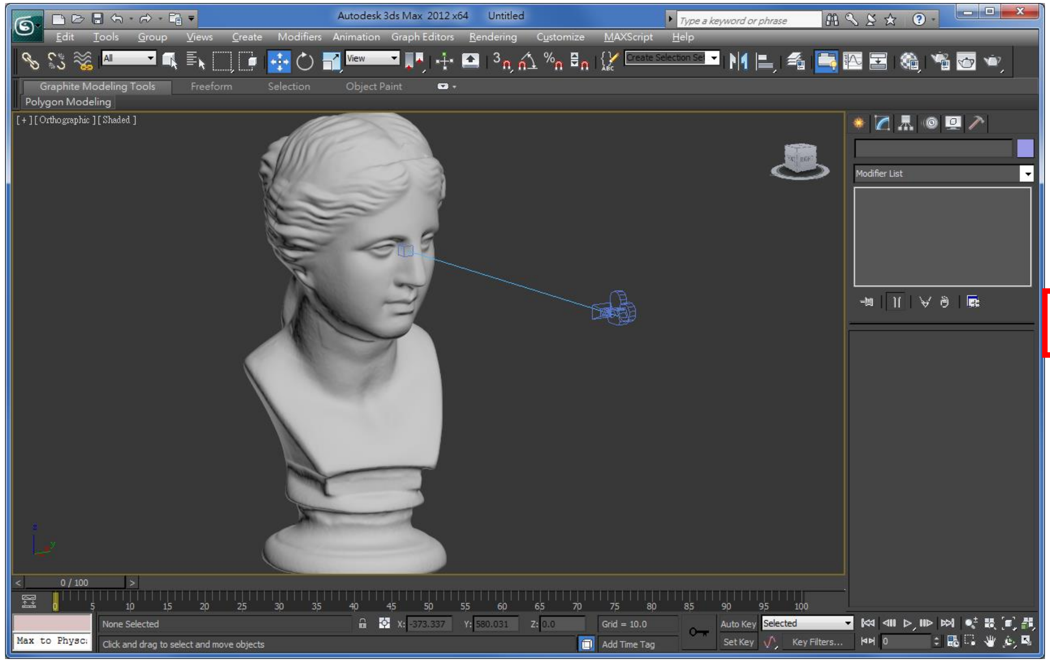
1	∞	∞	1
1	3	3	1
1	3	3	1
1	3	3	1

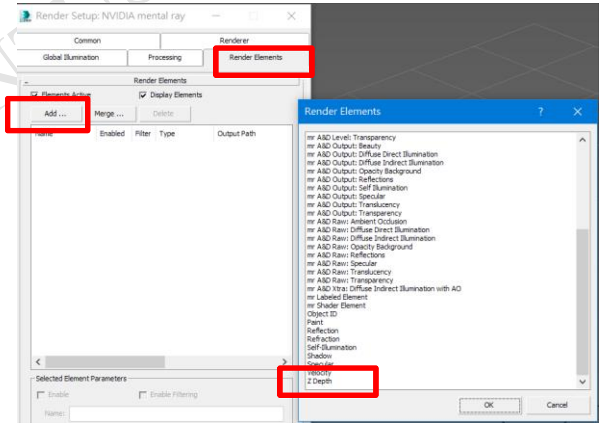
Depth buffer (image)

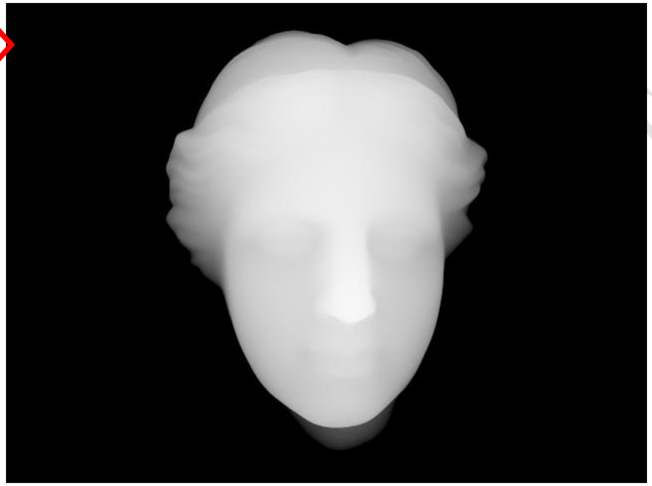
NTUST. CIIC.
NTUST. CIIC.
CIIC.

Z-buffering (depth buffering)

- Render the scene as a depth image—example.

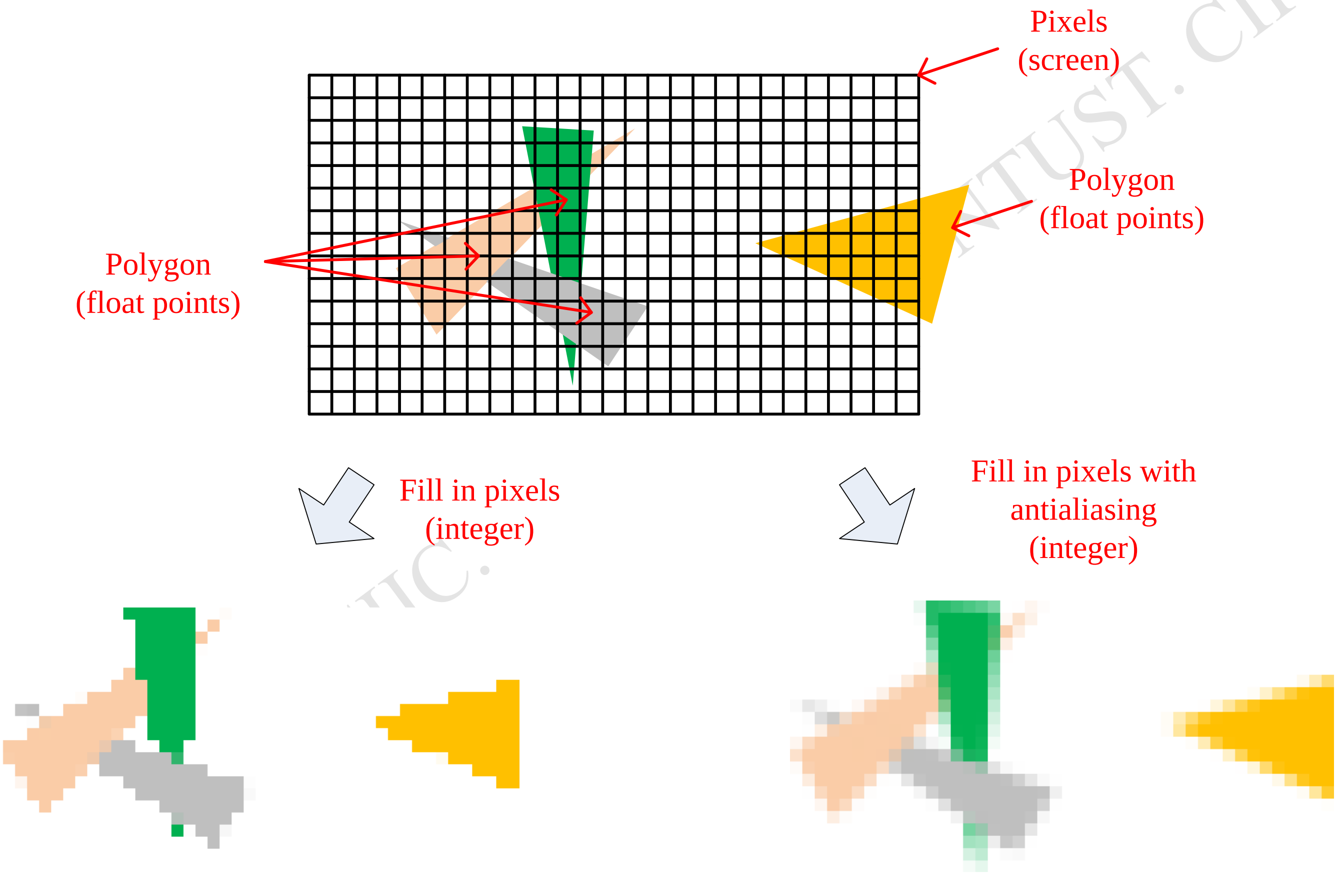






NTUST. CIIC.

Z-buffering (depth buffering)



NTUST. CIIC.

Z-buffering (depth buffering)

NTUST. CIIC.

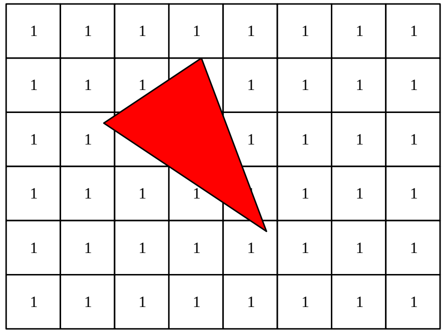
Depth buffer

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	-1	1	1	1	1	1
1	1	1	1	1	1	1	1

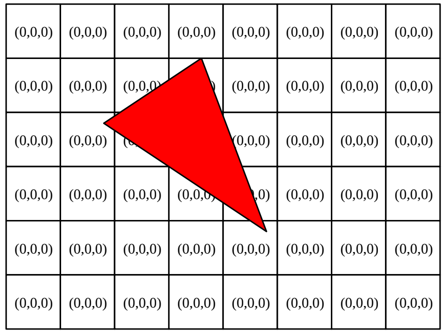
Initial status

Color buffer

(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)



Fill in a triangle



1	1	1	1	1	1	1	1
1	1	1	0.6	1	1	1	1
1	1	0.5	0.4	1	1	1	1
1	1	1	0.3	0.25	1	1	1
1	1	1	-1	1	1	1	1
1	1	1	1	1	1	1	1

Internal data

(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(1,0,0)	(1,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(1,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)

Z-buffering (depth buffering)

- Color buffer and depth buffer in openGL
  - Readily use “glReadPixels” to retrieve data.

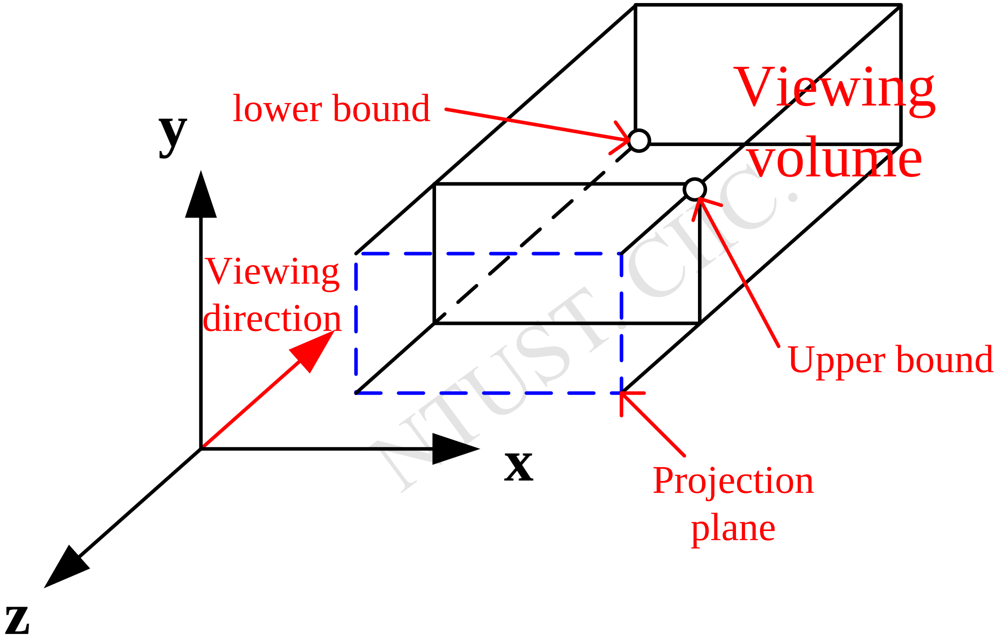
NTUST. CIIC.

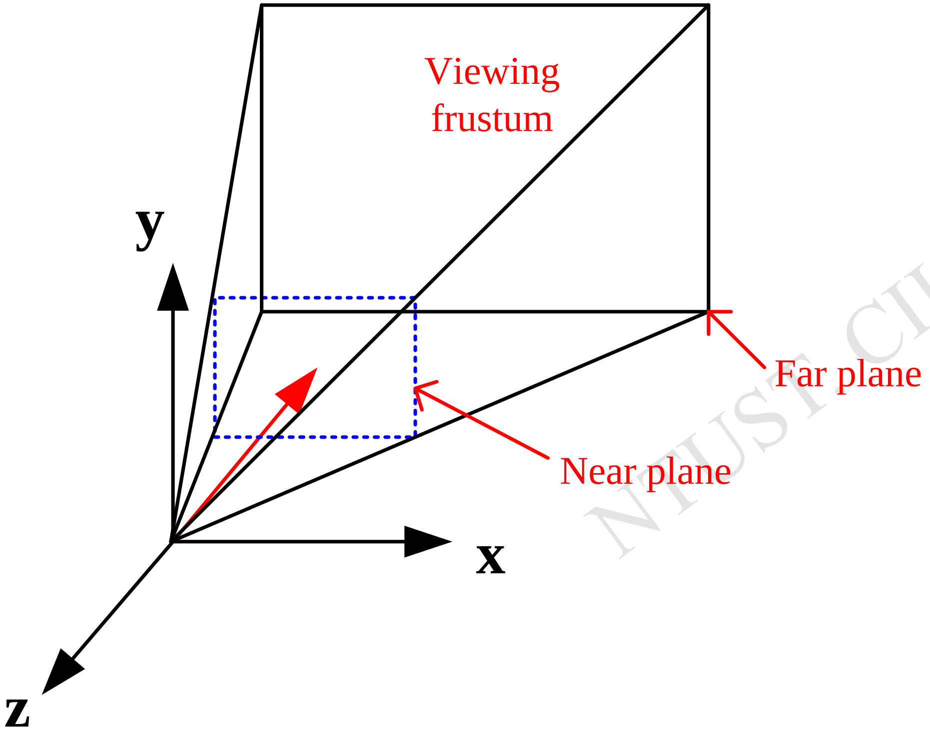
Color buffer
(mostly what you see)

Depth buffer (*z*-buffer)
(mostly internal data, this figure shows depth is converted from [0,1] to [0,255])

Z-buffering (depth buffering)

- Depth Resolution problem
- Deal with the “near” and “far” planes carefully





NTUST. CIIC.

Z-buffering (depth buffering)

- In openGL, switch with GL_DEPTH by glEnable / glDisable.
- Set the range of mapped depth. In default the value is [0,1].

glDepthRange

The **glDepthRange** function specifies the mapping of *z* values from normalized device coordinates to window coordinates.

```
void glDepthRange(
  GLclampd znear,
  GLclampd zfar
);
```

Parameters

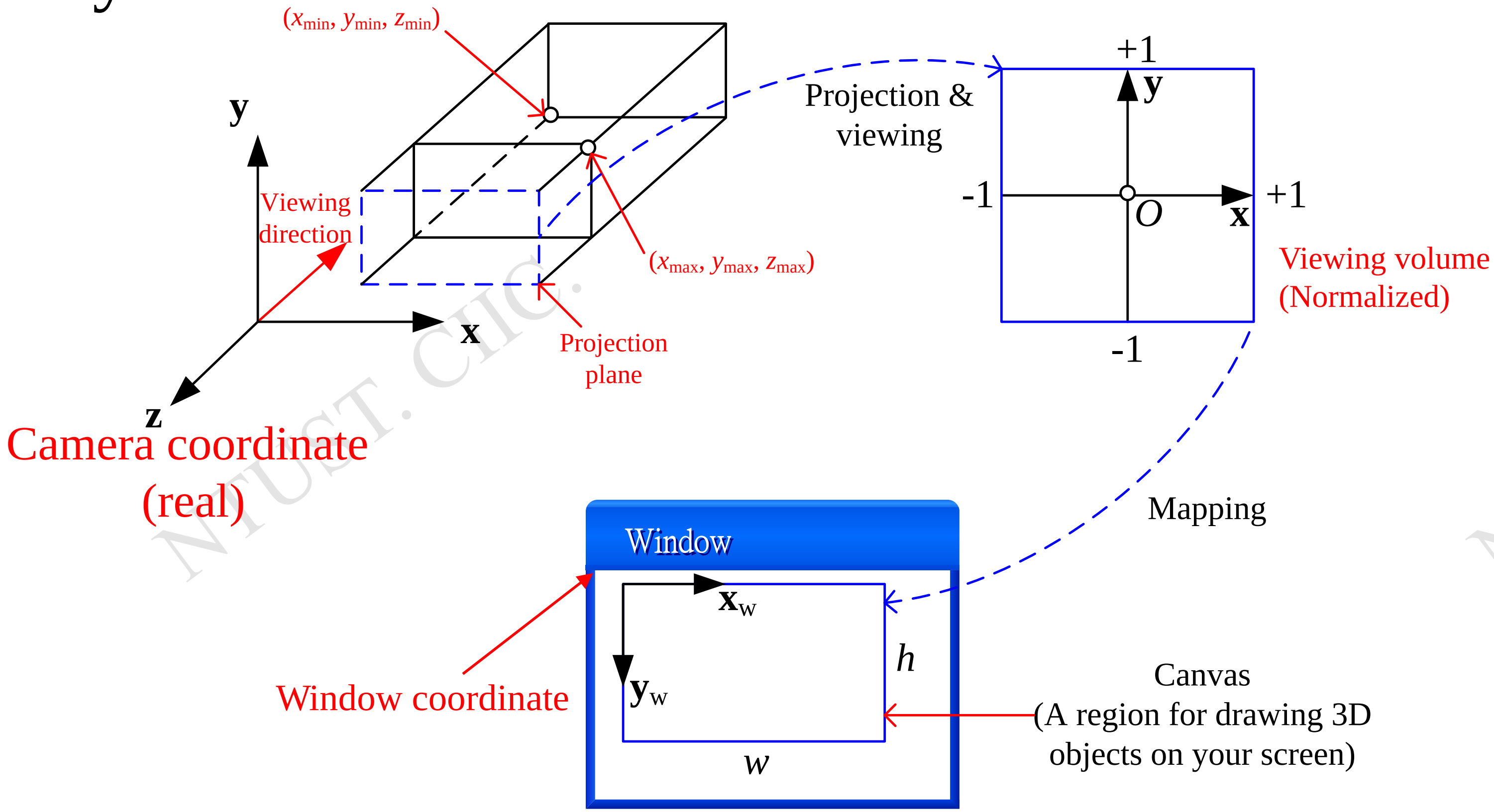
znear
The mapping of the near clipping plane to window coordinates. The default value is 0.

zfar
The mapping of the far clipping plane to window coordinates. The default value is 1.

NTUST.CIIC. NTUST.CIIC. CIIC.

Z-buffering (depth buffering)

- The Z-value is from ”viewing volume” to a clamped range [0,1]. An example of parallel projection.
- Note the ray directions.

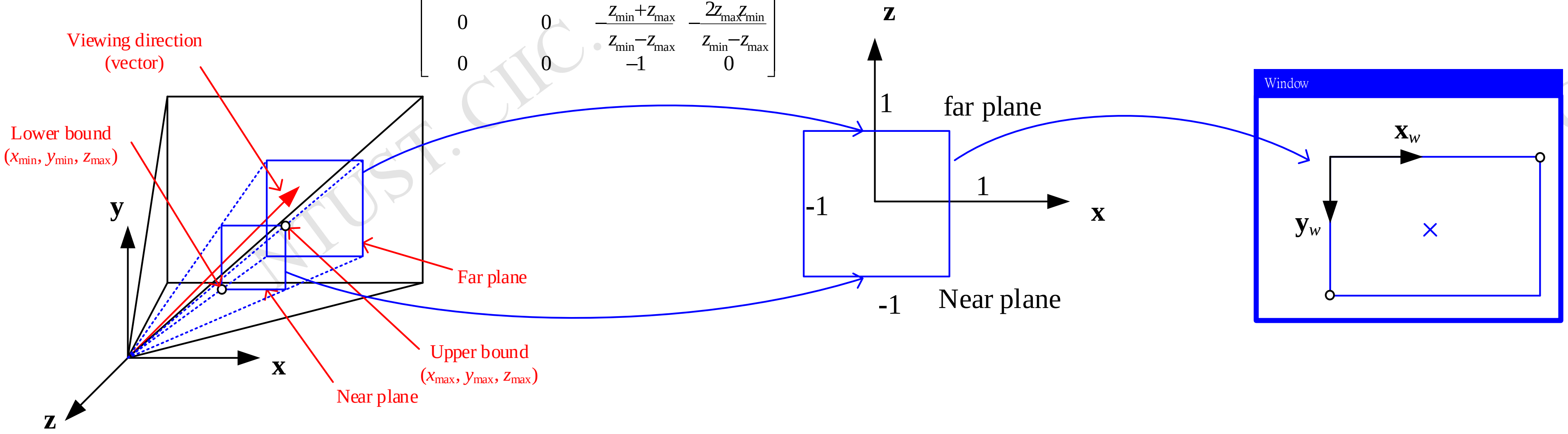


NTUST. CIIC.

Z-buffering (depth buffering)

- The Z-value is from "viewing frustum" to a clamped range [0,1]. An example of perspective projection.
- Note: the ray directions change.

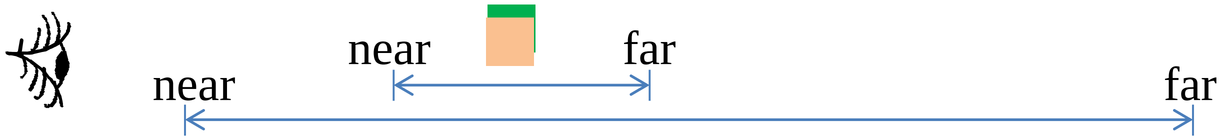
$$\begin{bmatrix} \frac{-2z_{max}}{x_{max}-x_{min}} & 0 & \frac{x_{max}+x_{min}}{x_{max}-x_{min}} & 0 \\ 0 & \frac{-2z_{max}}{y_{max}-y_{min}} & \frac{y_{max}+y_{min}}{y_{max}-y_{min}} & 0 \\ 0 & 0 & -\frac{z_{min}+z_{max}}{z_{min}-z_{max}} & -\frac{2z_{max}z_{min}}{z_{min}-z_{max}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

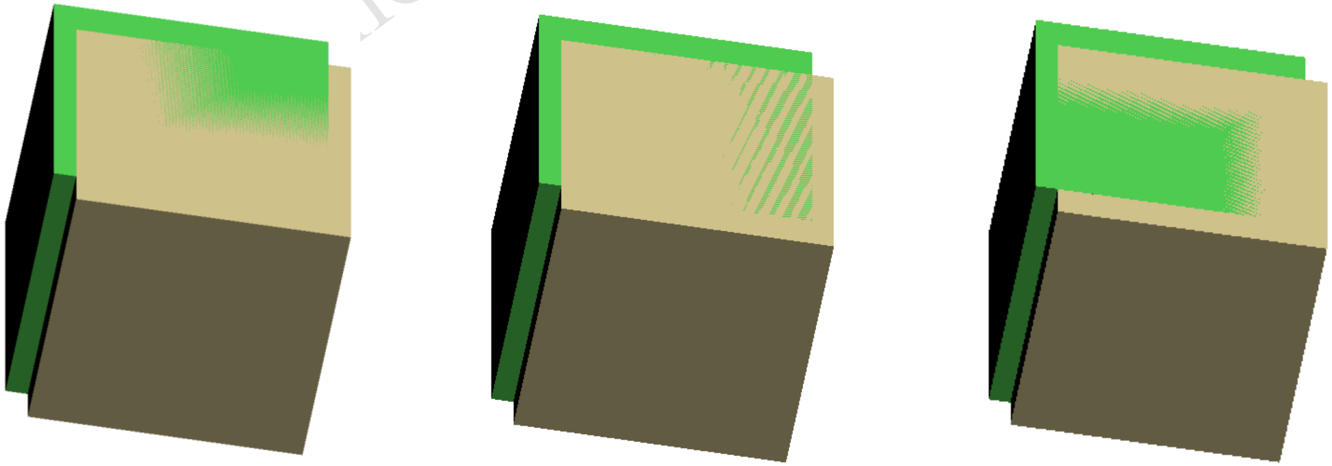


NTUST.CIIC.

Z-buffering (depth buffering)

- How to display two planes which have almost the same depth value?
  - There is no good solution. The way you may try is to shorten the distance between "near" and "far" plane

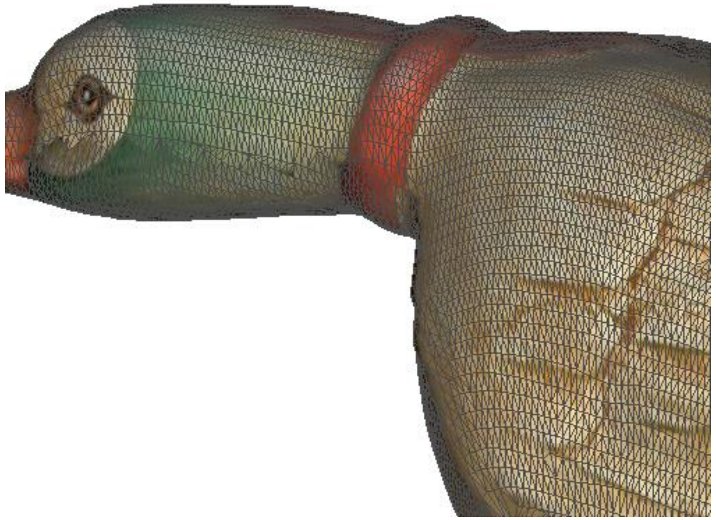


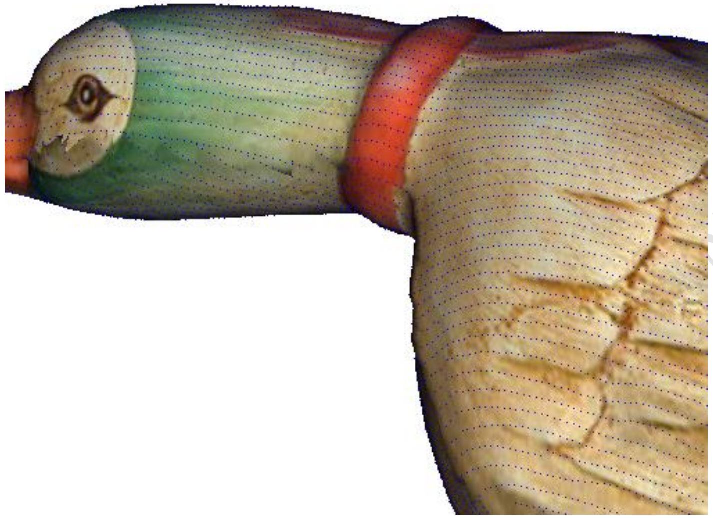


NTUST.CIIC.

Z-buffering (depth buffering)

- How to display two different types objects, for example: vertexes on surfaces, lines on surface.
  - Slightly shift the positions of vertexes (or lines) to the camera, by software or API (openGL, GL_POLYGON_OFFSET_FILL).
  - Shorten the distance between “near” and “far” planes, (and enlarge vertex’s size, line’s width)

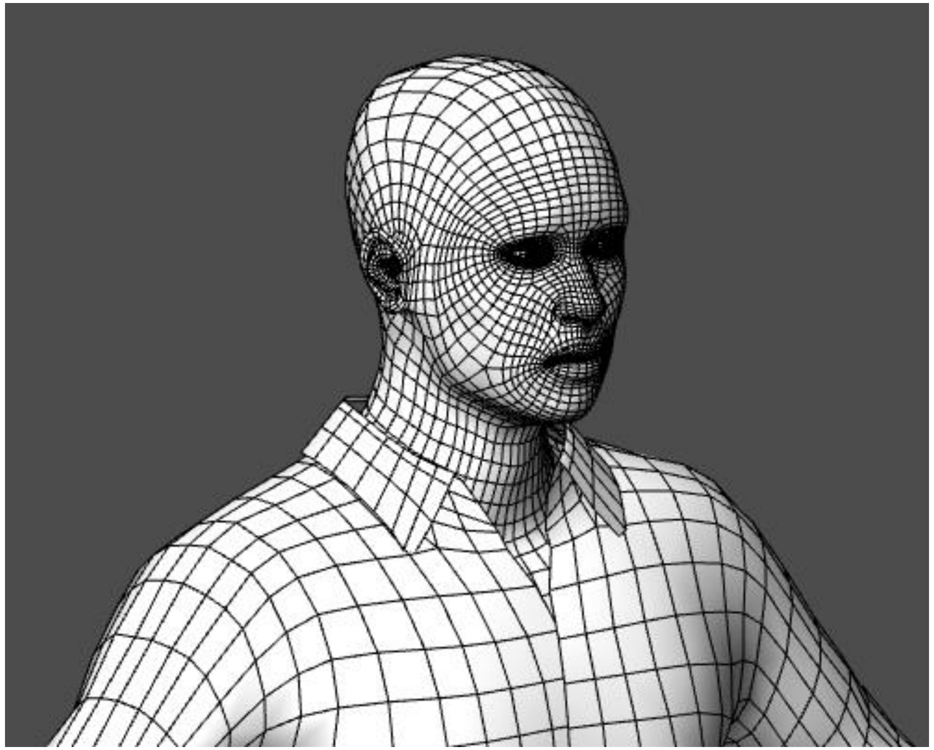
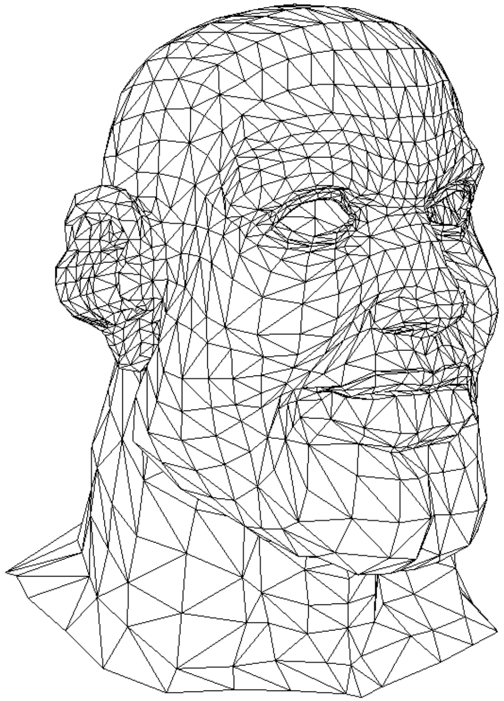




NTUST. CIIC.

Z-buffering (depth buffering)

- Hidden line rendering
  - How to perform it?

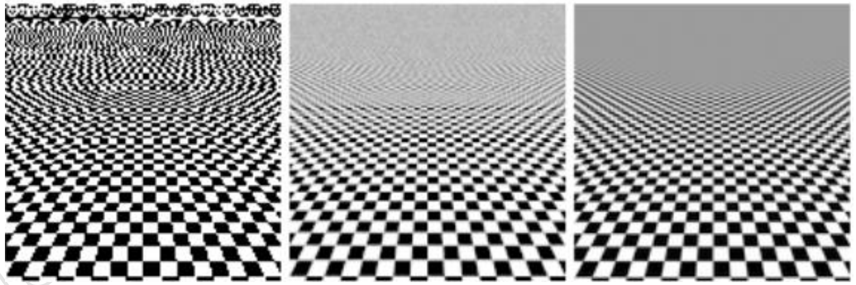



NTUST. CIIC.
NTUST. CIIC.
NTUST. CIIC.

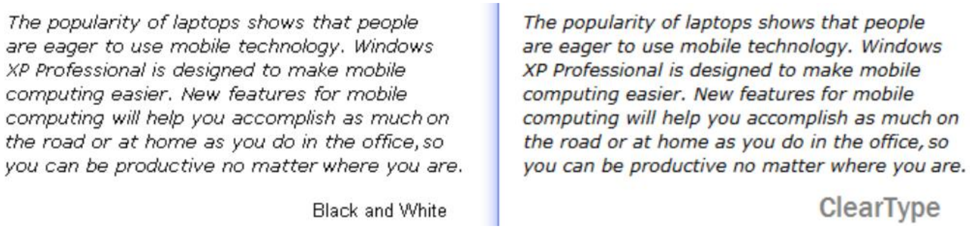
Aliasing and anti-aliasing in shading

- Anti-aliasing is a kind of sub-pixel rendering.

In boundary of polygon



In dense pattern



In text rendering
(ex. Clear type)

NTUST. CIIC.

CIT 色彩與照明科技研究所
Graduate Institute of
Color and Illumination Technology

Find us on:
facebook

NTUST. CIIC.
NTUST. CIIC.
NTUST. CIIC.

This photo is licensed under CC BY-ND