


# CS5182 Computer Graphics

## The Rendering Pipeline

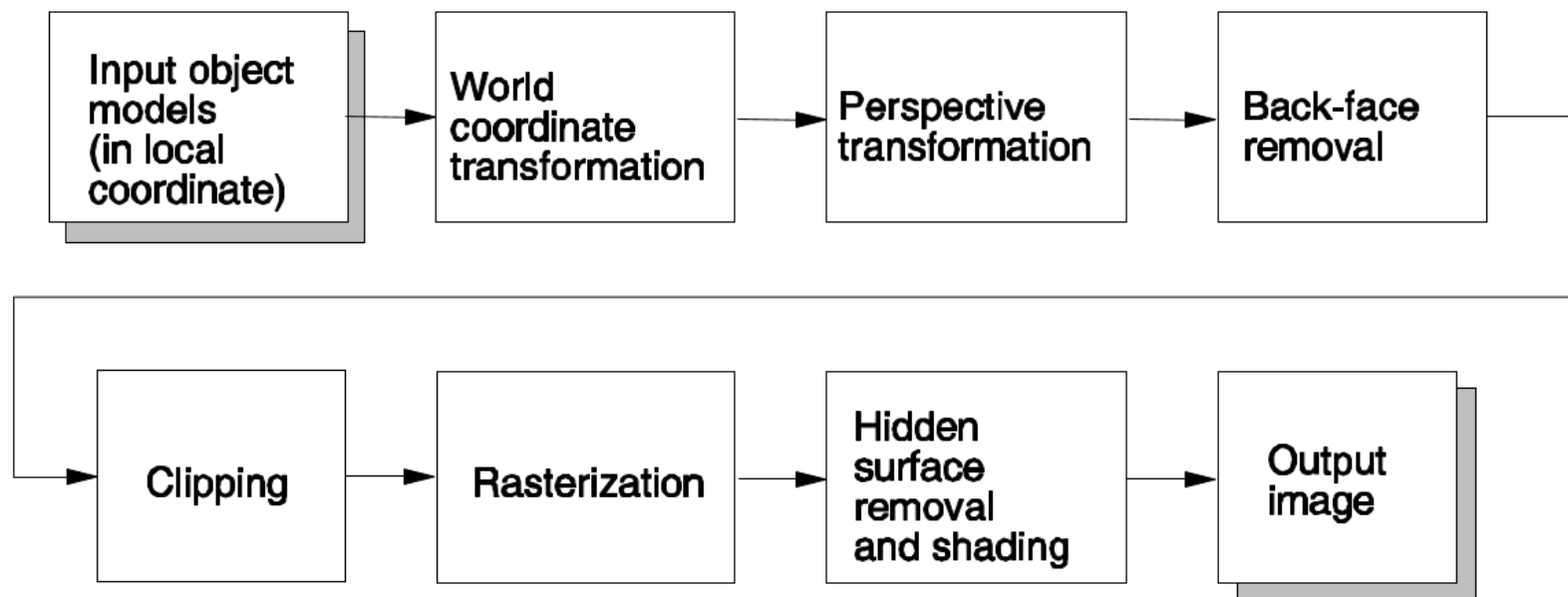


2024/25 Semester A

City University of Hong Kong (DG)

# The Rendering Pipeline

- Surface primitives (such as polygons) representing a continuous object are processed as a collection of discrete pixels one by one, typically in a random order. This process is composed of many stages called the *rendering pipeline*.



Note that the rendering pipeline may be implemented with slightly different order of the operations to optimize the performance.

# The Rendering Pipeline

## □ Object models and primitives

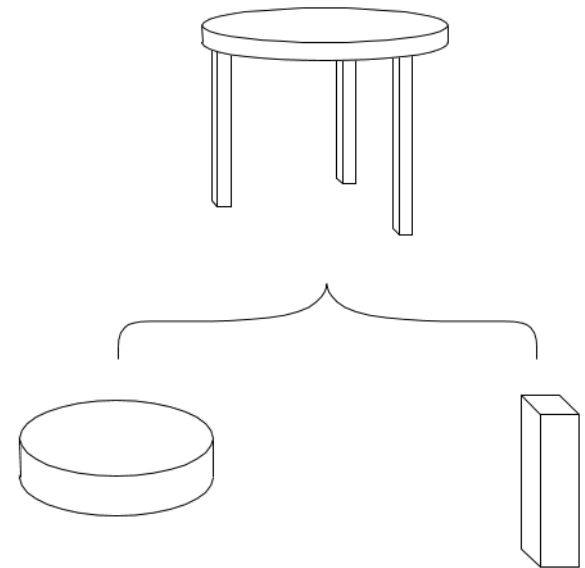
- Each object in the scene is first created using a software program such as *3D Studio Max*. An object model may have the following properties:

- it may be composed of many primitives, such as points, lines, planes, circles, ...

- primitives may need to be scaled, rotated and translated in order for them to form the object.

- each object model is in its own coordinate system called *local coordinate*.

- to render an image, primitives are sent to the rendering pipeline one by one, typically in random order.



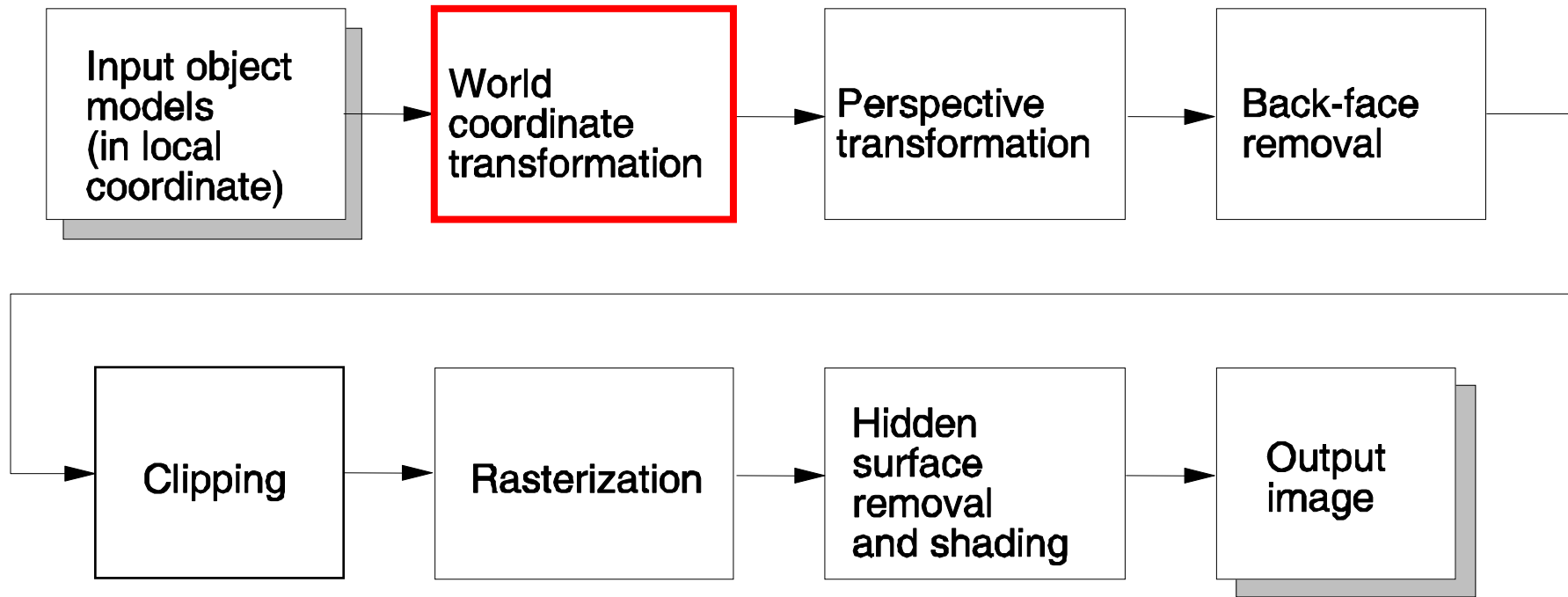
# Object Modeling (recall)

---

- ❑ 2D drawing: a line, a circle
- ❑ Man-made 3D modelling using software
- ❑ 3D Scanning
- ❑ 3D Generation
- ❑ 3D representations: point clouds, polygonal mesh, subdivision surface, ...

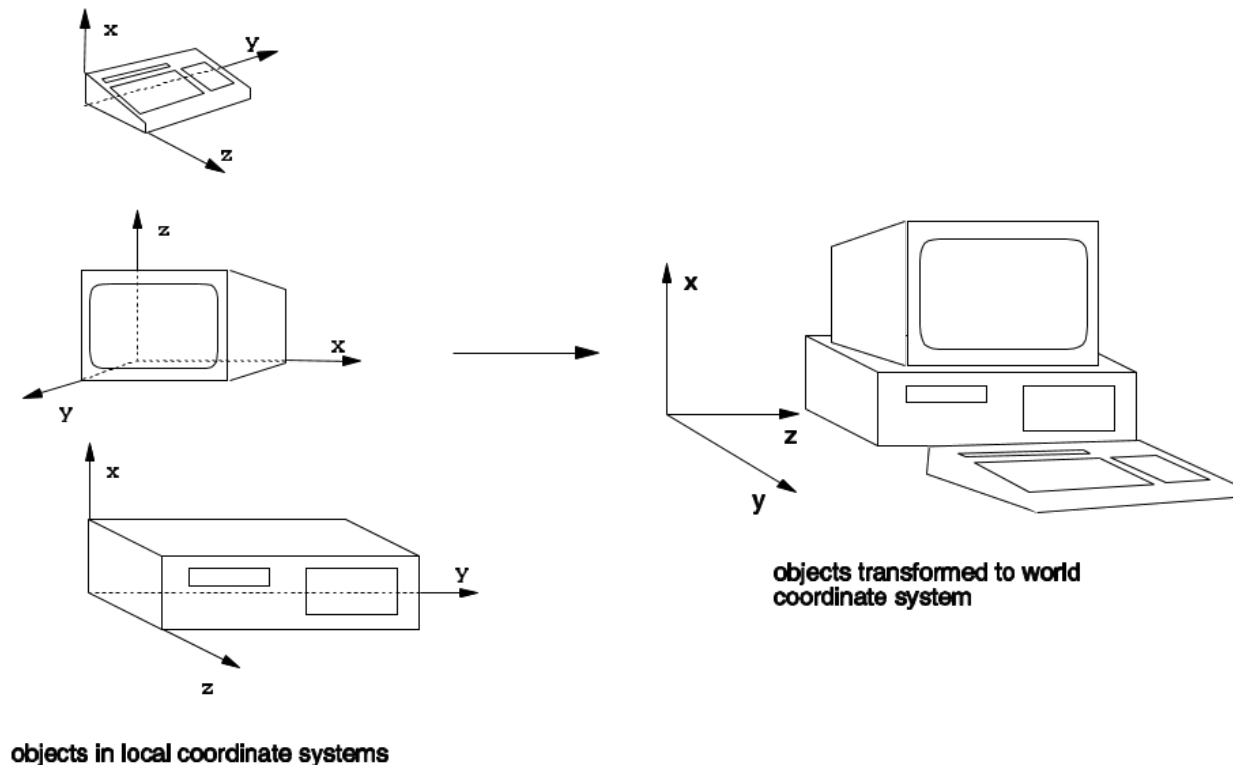
# The Rendering Pipeline

---



# The Rendering Pipeline

- Word coordinate transformation
  - This step transforms each object from its local coordinate system to a common coordinate system called the world coordinate system – creating a geometric relationship among all objects to form the scene.



# Transformation (recall)

---

## □ Transformation

- Homogeneous coordinates
- 2D transformations
  - Translation
  - Scaling
    - uniform/non-uniform, simple/general case
  - Rotation
    - rotation orientation, simple/general case, three-step trick
  - Shearing
- 3D transformations
  - Translation, scaling, rotation around x, y or z-axis
- Inverse transformation
- The order of transformations

# Exercise Questions

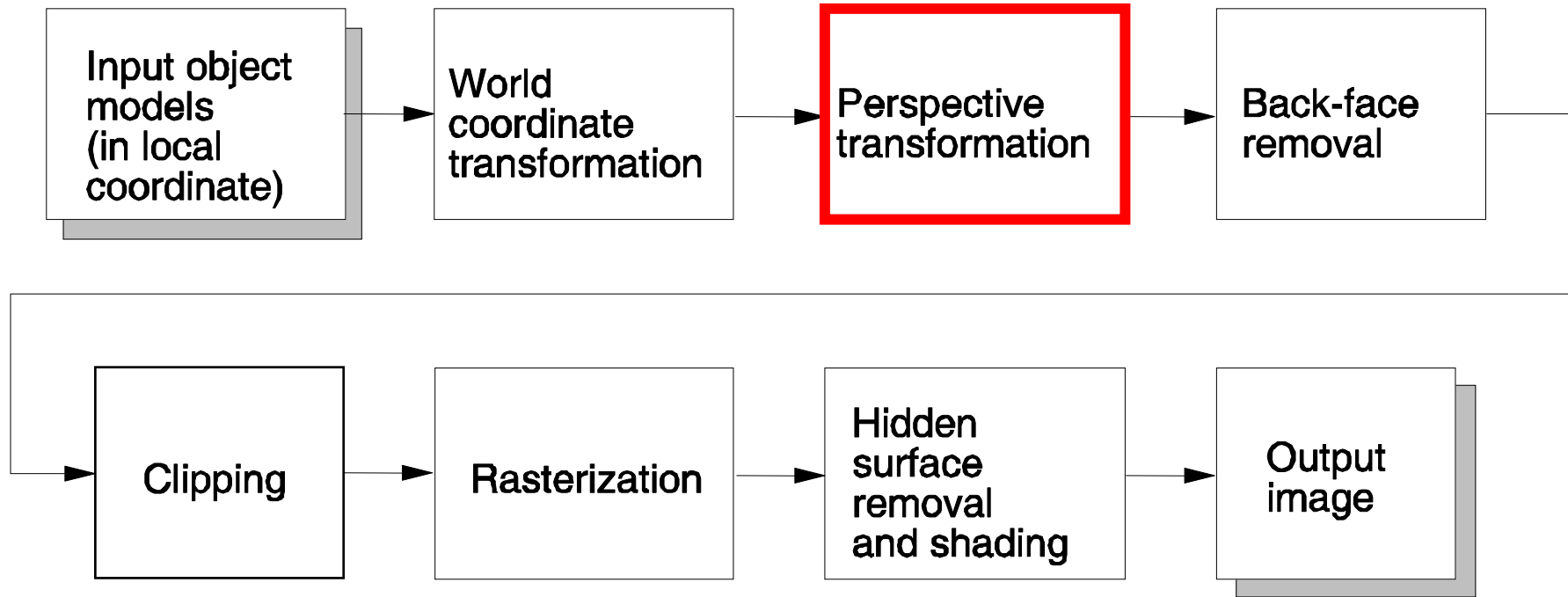
- A 3D object is rotated by 90 degrees about an axis passing from (0, 1, 1) to (2, 1, 1). Write out the transformation matrix.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



# The Rendering Pipeline

---



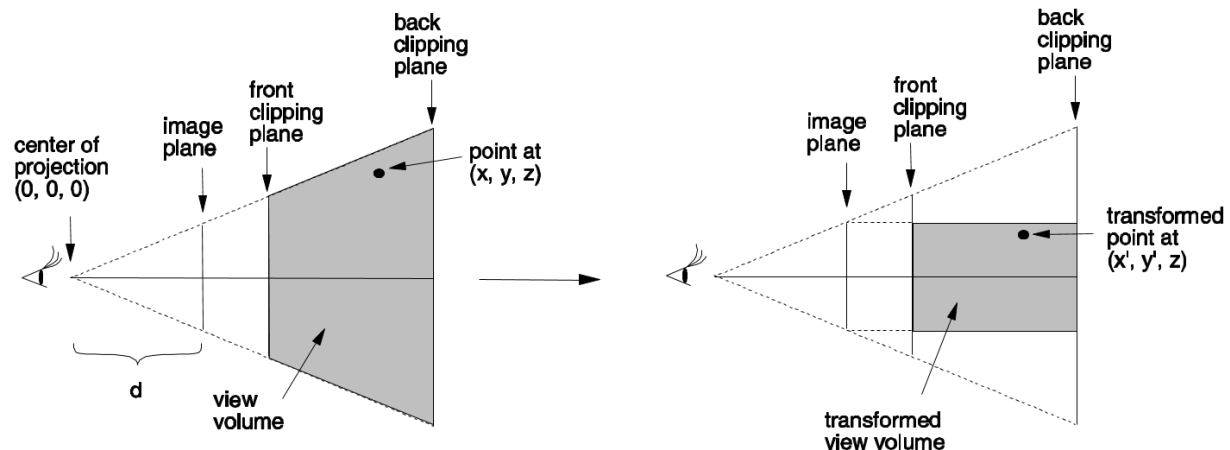
# The Rendering Pipeline

## □ Perspective transformation

- This step perspectively transforms each object such that distant object will appear smaller. After the transformation, the view volume in the shape of a frustum will become a regular parallelepiped.
- Given the position of an input vertex  $(x, y, z)$ , the transformed position of the vertex  $(x', y', z')$  are computed as

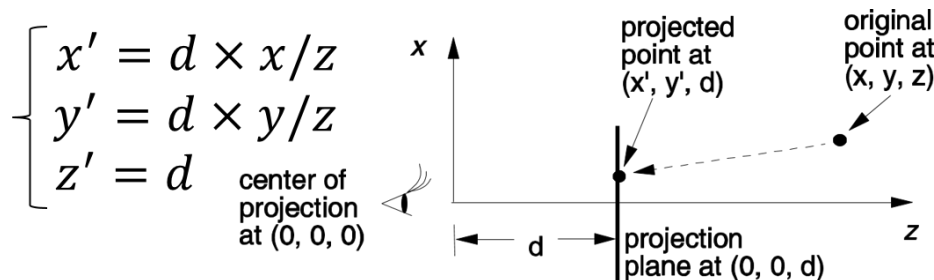
$$x' = d * x / z, y' = d * y / z, z' = z$$

where  $d$  is the distance of the image plane from the center of projection.

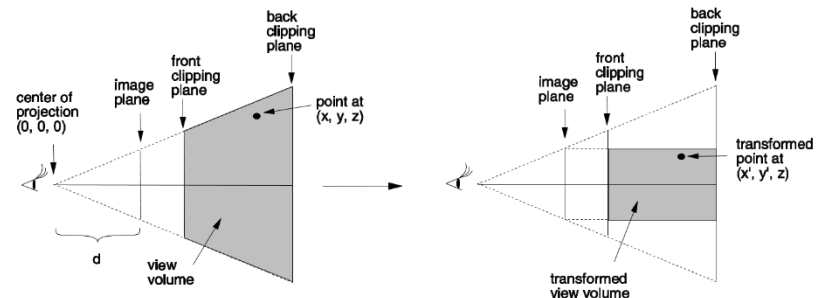


# The Rendering Pipeline

- Perspective transformation vs. Perspective projection
  - Perspective transformation is different from perspective projection in that perspective transformation retains the depth value of each transformed vertex, while perspective projection does not.
  - Before perspective transformation, all the projection lines converge to the center of projection. After the transformation, all the lines are parallel to each others.
  - The front clipping plane and the back clipping plane are used to remove objects which are outside our interest.



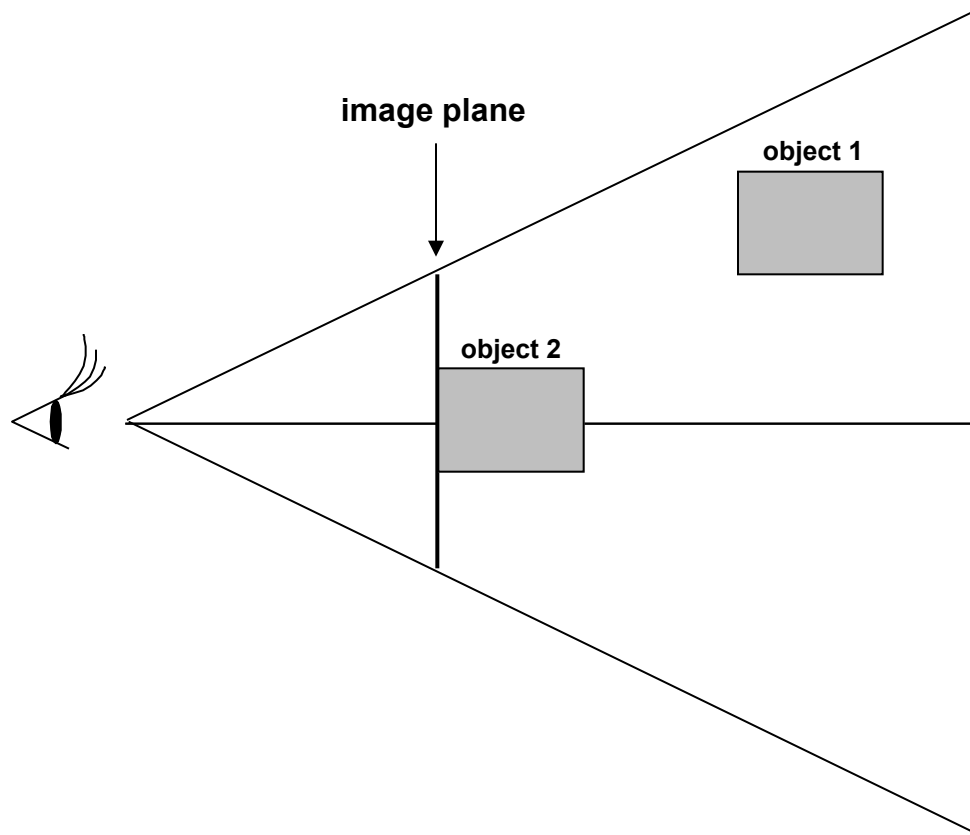
Perspective projection



Perspective transformation

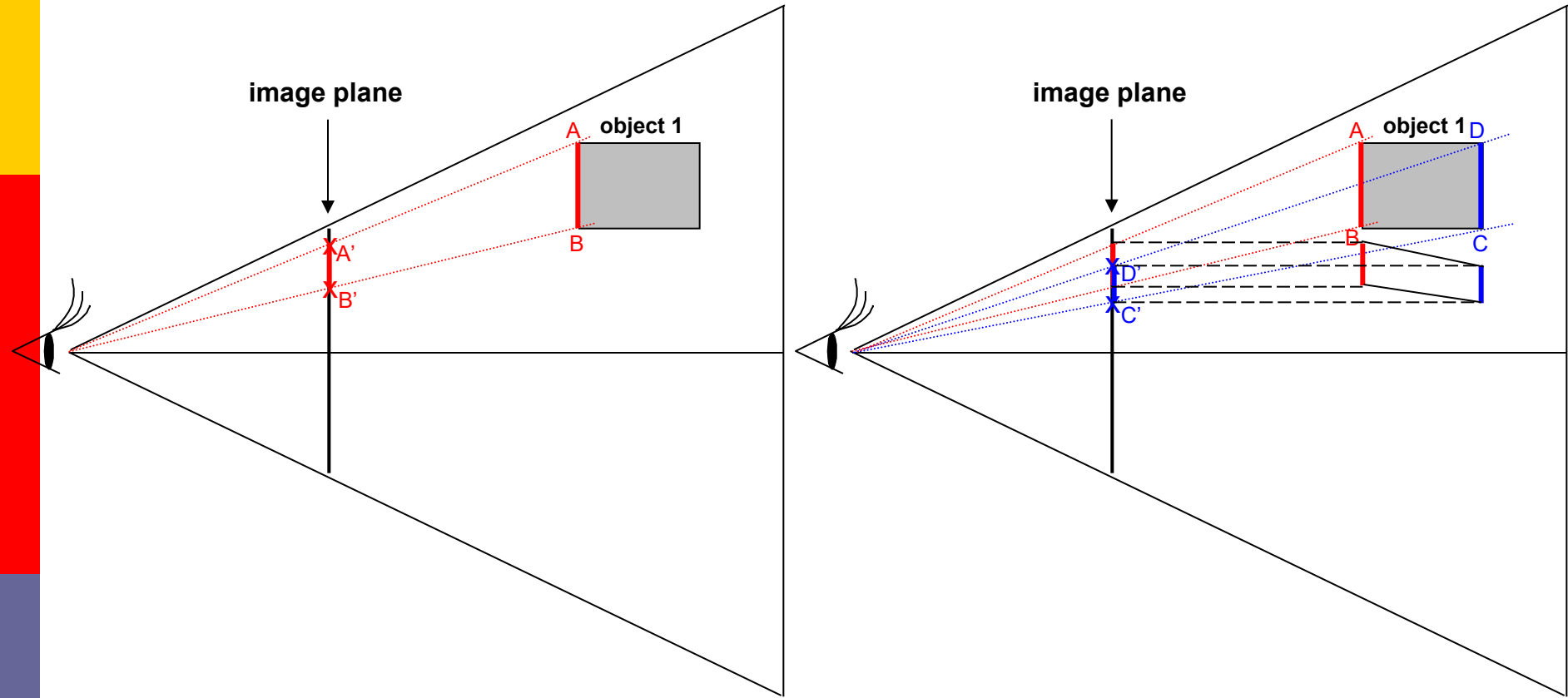
# Exercise Question

- Perspective transform the following two objects graphically:



# Exercise Question

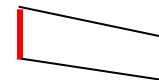
## To transform object 1



Ans: The edge AB is projected on the image plane as edge A'B'.

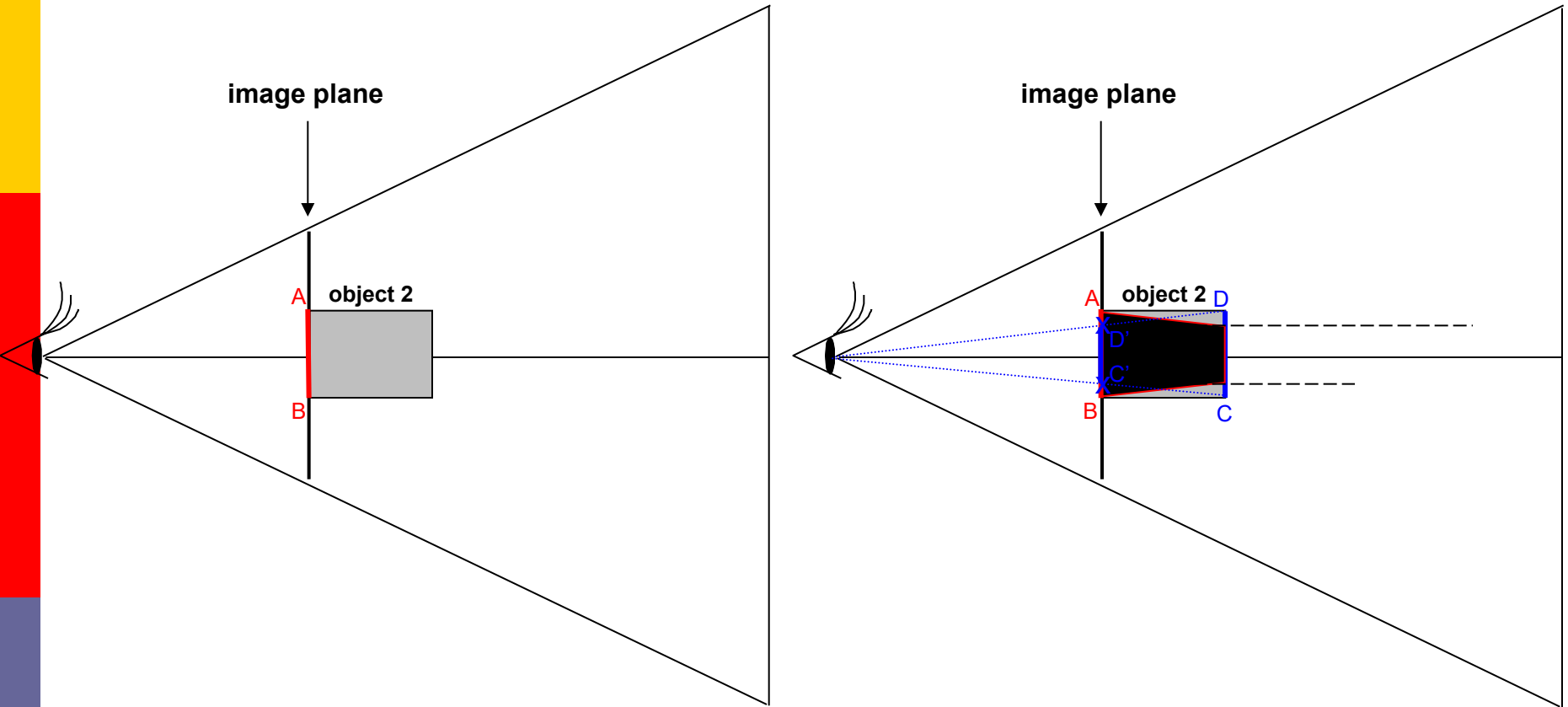
The edge CD is projected on the image plane as edge C'D'.

By considering the depths of the vertices, the result is:



# Exercise Question

## To transform object 2

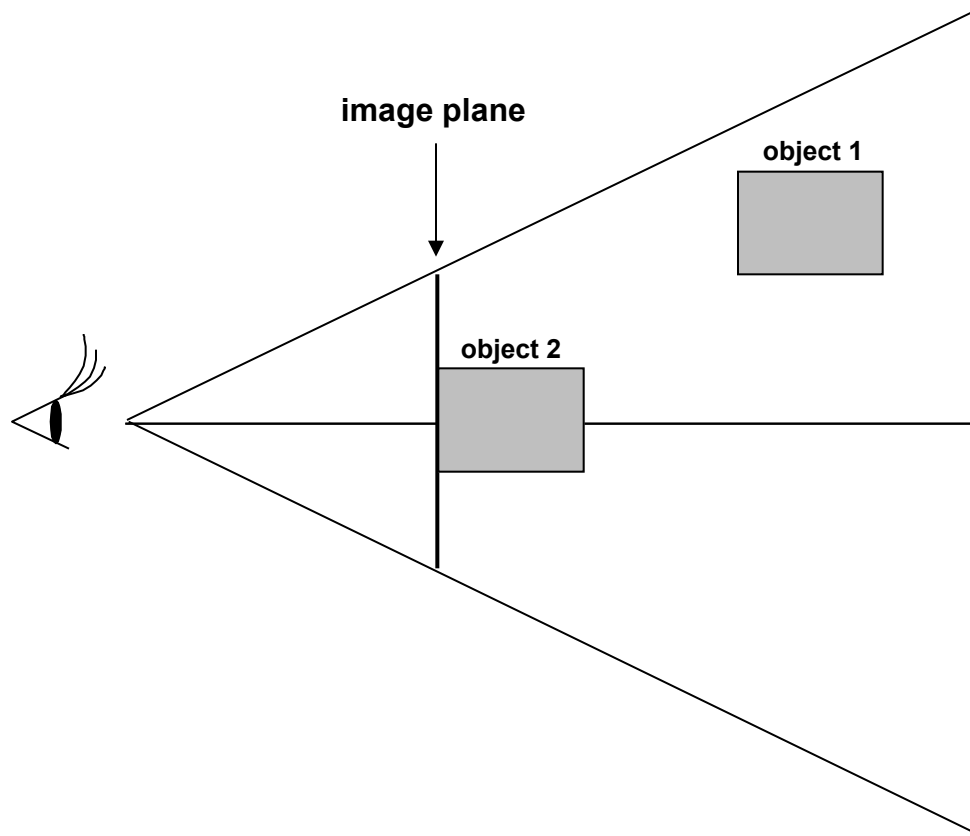


Ans: The edge AB itself already lies on the image plane.  
The edge CD is projected on the image plane as edge C'D'.  
By considering the depths of the vertices, the result is:



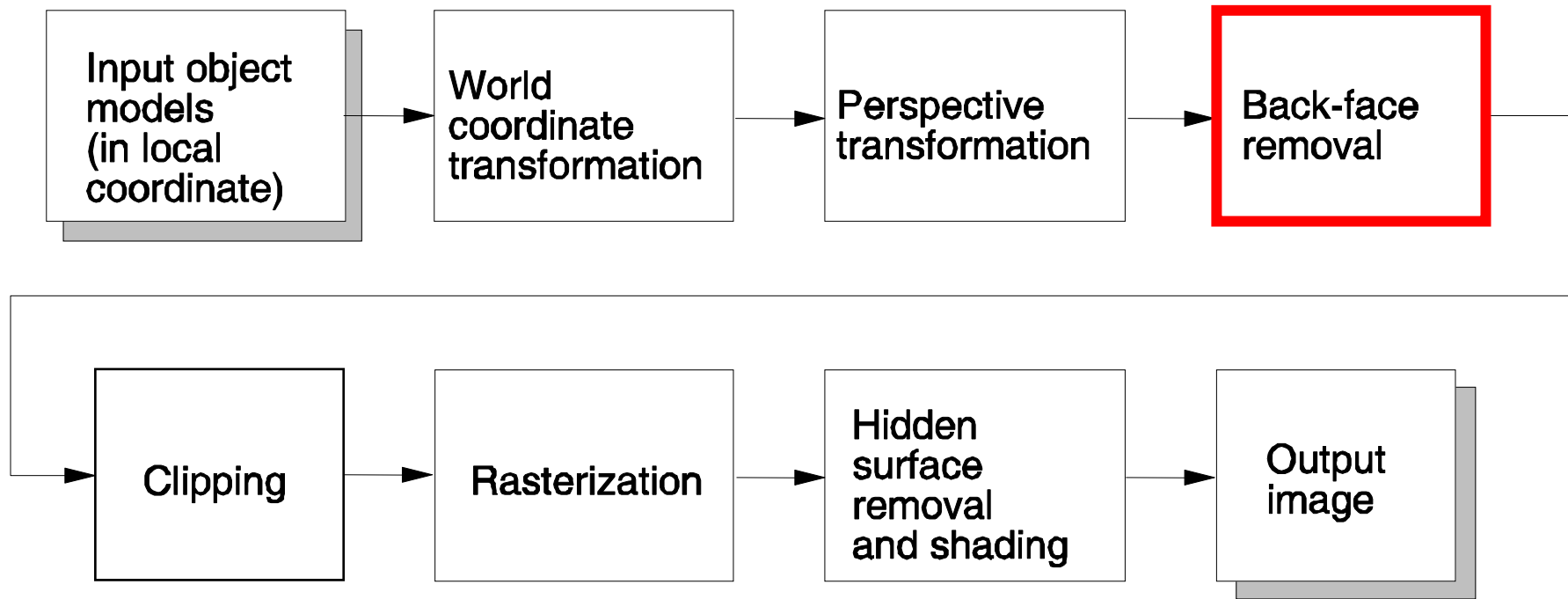
# Exercise Question

- Perspective transform the following two objects graphically:



# The Rendering Pipeline

---





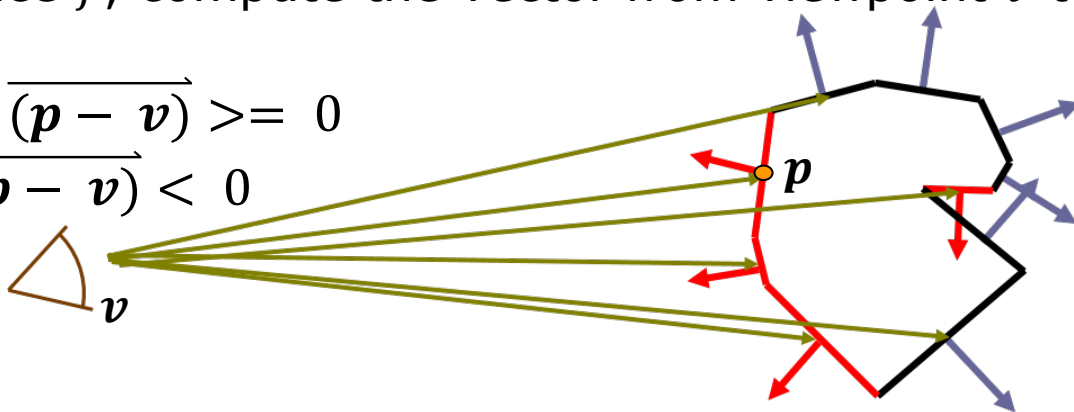
# The Rendering Pipeline

## □ Back-face removal

- In a solid object, some surfaces may be facing the viewer (front faces); others may be facing away from the viewer (back faces). These back faces contribute to approximately half of the total number of surfaces. As we cannot see these surfaces anyway, we would like to remove them to save processing time.
- Back-face culling algorithm

For each polygonal face  $f$ , compute the vector from viewpoint  $v$  to any point  $p$  on  $f$

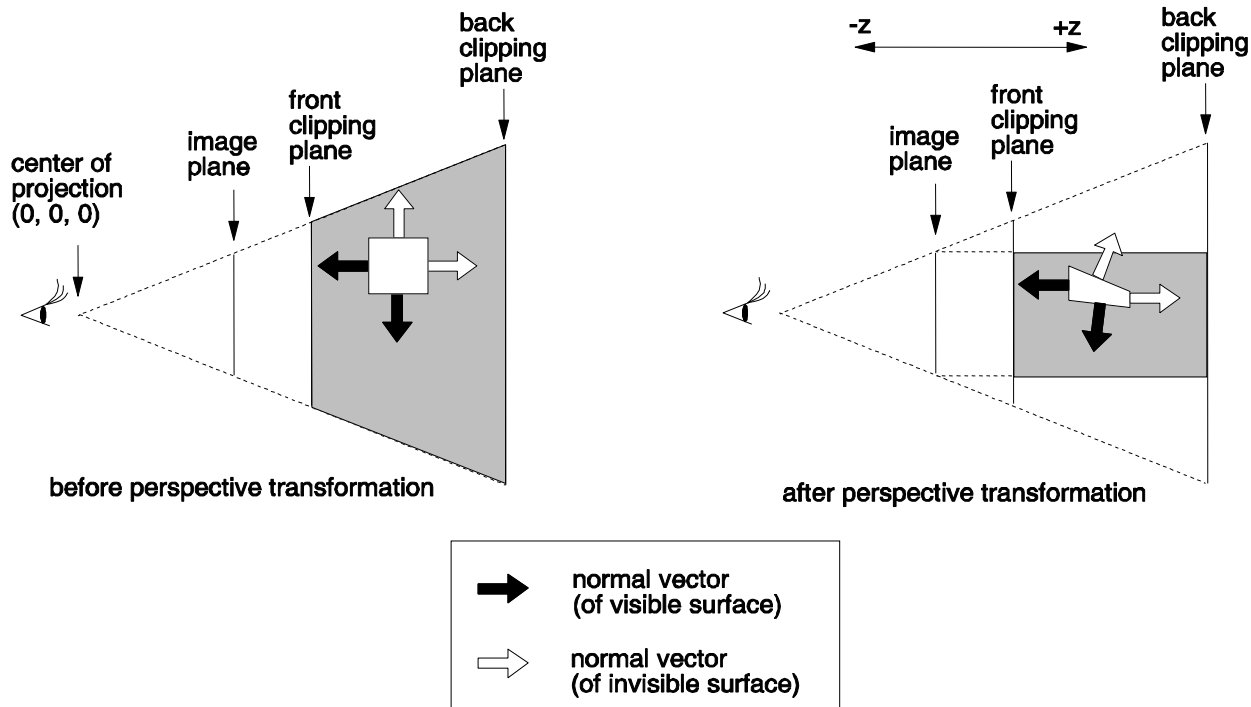
- Invisible if  $\vec{n} \cdot \overrightarrow{(p - v)} \geq 0$
- Visible if  $\vec{n} \cdot \overrightarrow{(p - v)} < 0$



# The Rendering Pipeline

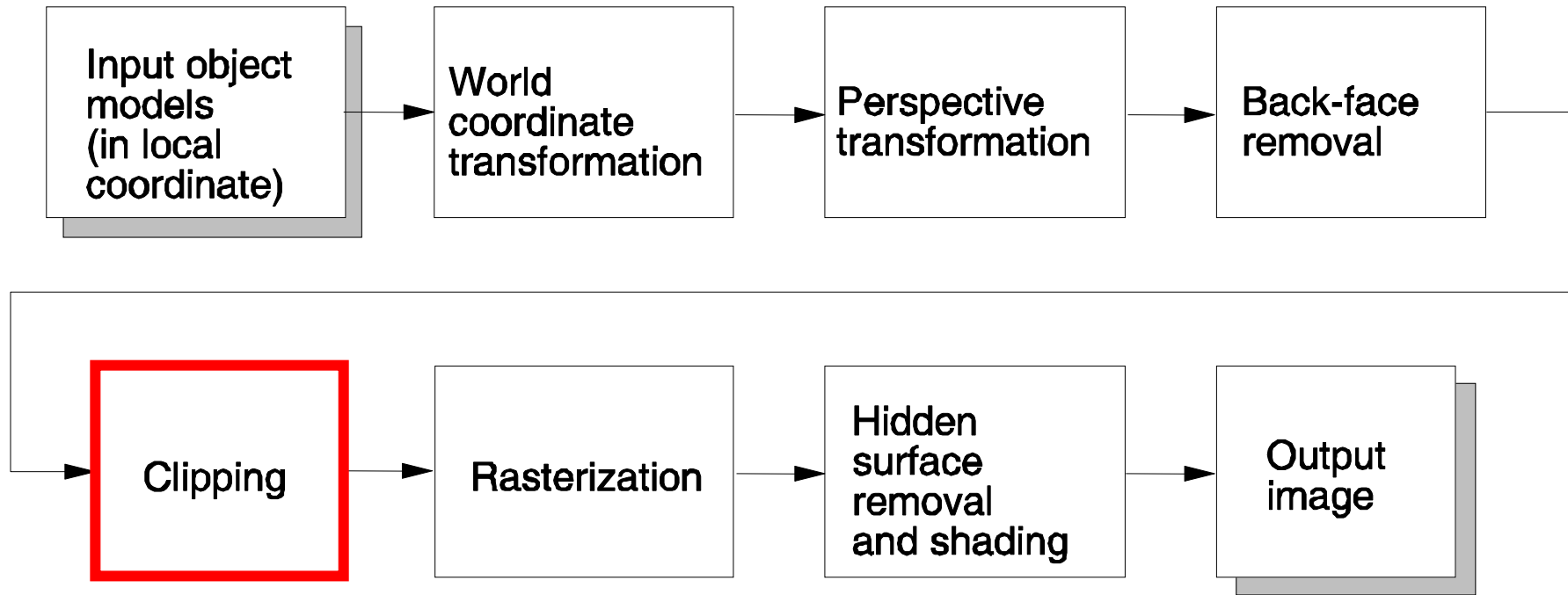
## □ Back-face removal

- As mentioned earlier, after perspective transformation, the projection becomes a parallel projection. The test becomes very simple. If the  $z$  component of the normal vector is positive, it is a back face. If it is negative, it is a front face.



# The Rendering Pipeline

---



# The Rendering Pipeline

---

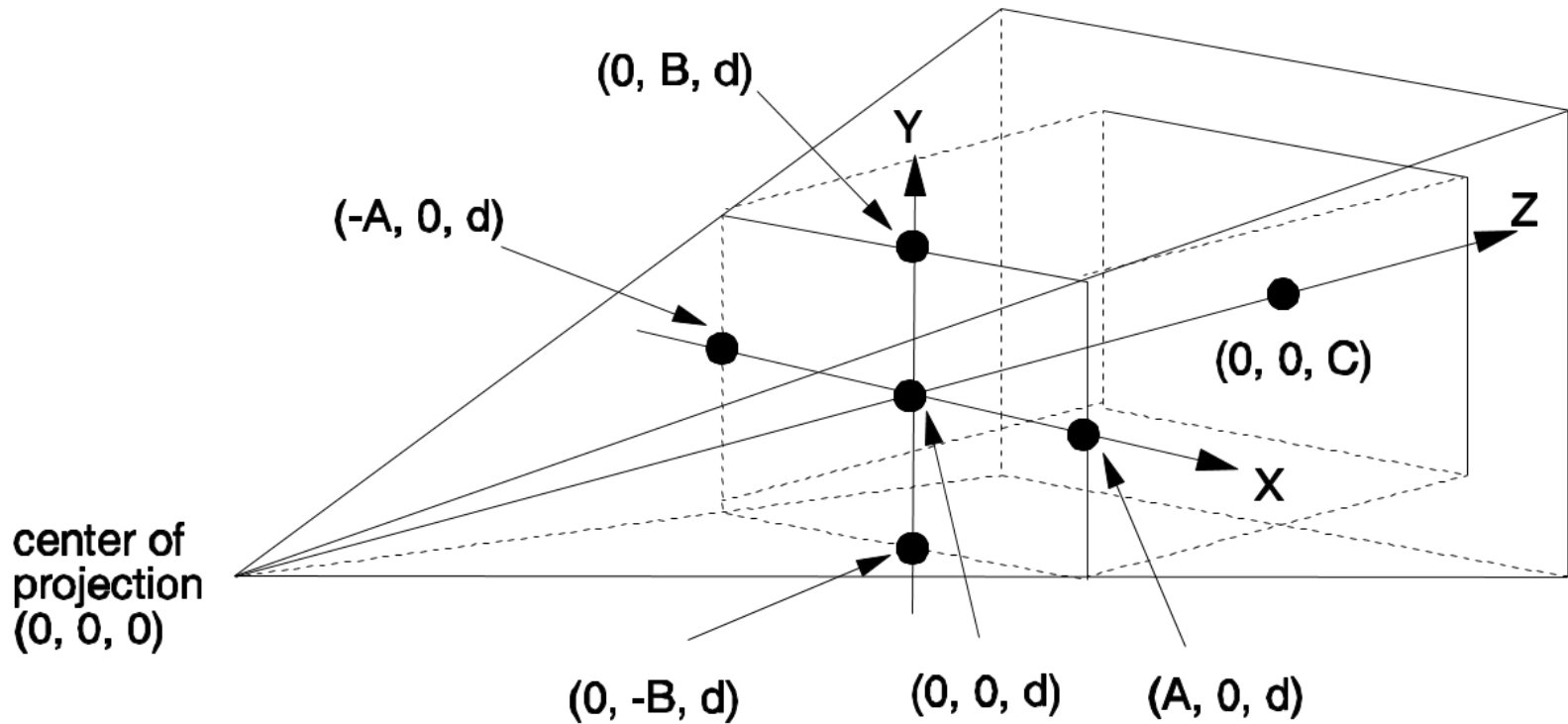
## □ Clipping

- This step removes all primitives and parts of primitives which are outside the view volume.
- If it is performed before perspective transformation, this operation will have to clip primitives against arbitrary planes. However, after perspective transformation, the 6 clipping planes, which form the parallelepiped, become parallel to the three axes. Thus clipping is straightforward and can be performed in 2D.
- Assuming that the origin is at the center of projection, the six clipping planes after perspective transformation becomes:

$$x = -A, \quad x = A, \quad y = -B, \quad y = B, \quad z = d, \quad z = C$$

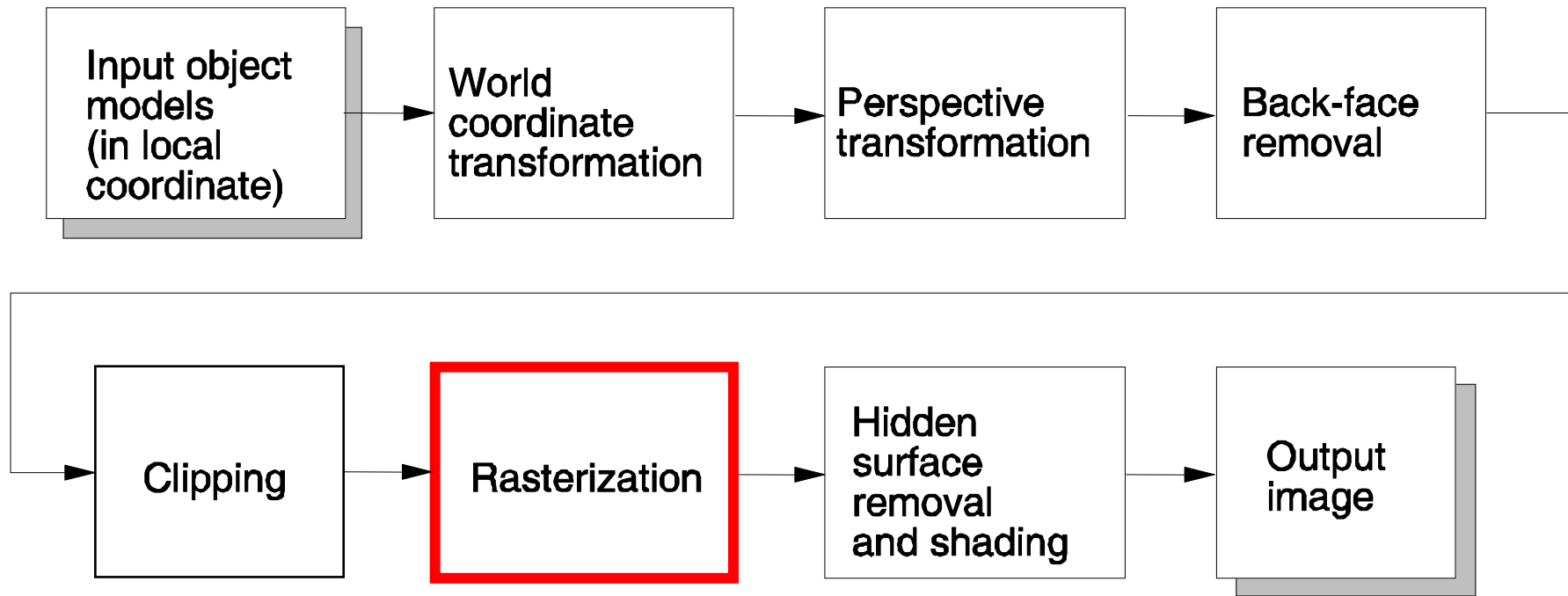
# The Rendering Pipeline

## □ Clipping



# The Rendering Pipeline

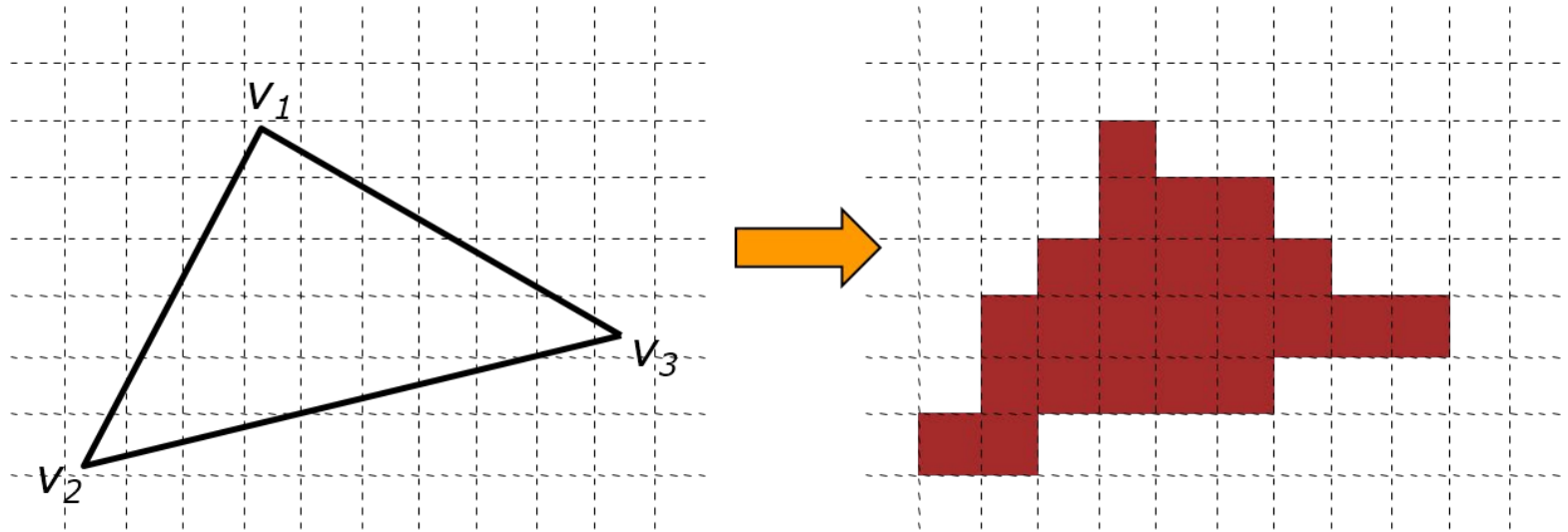
---



# The Rendering Pipeline

## □ Rasterization

- The process of taking a primitive and figuring out which pixels it covers is called rasterization.
- Consider rasterizing a triangle ( $v_1, v_2, v_3$ ).



# The Rendering Pipeline

## □ Rasterization

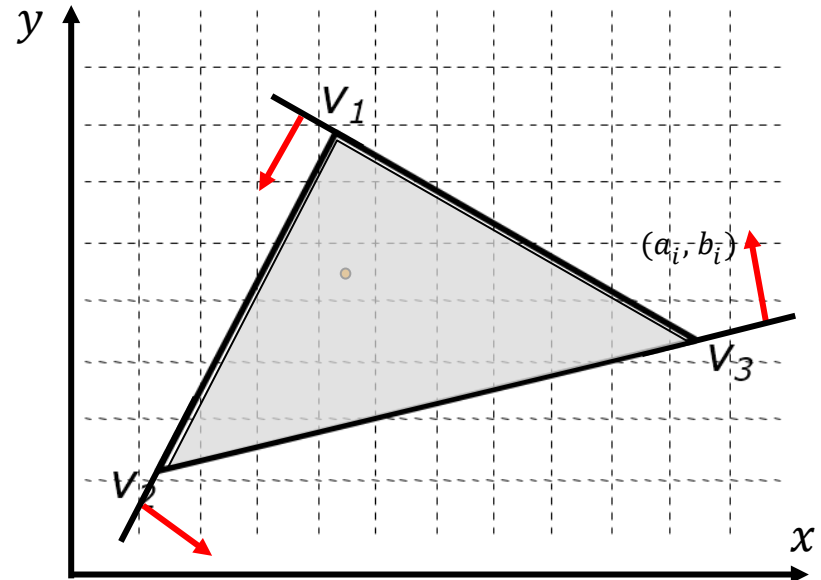
- The interior of the triangle is the set of points that is inside all three half-spaces defined by these lines

$$E_i(x, y) = a_i x + b_i y + c_i \quad (i = 1, 2, 3)$$

$(x, y)$  within triangle



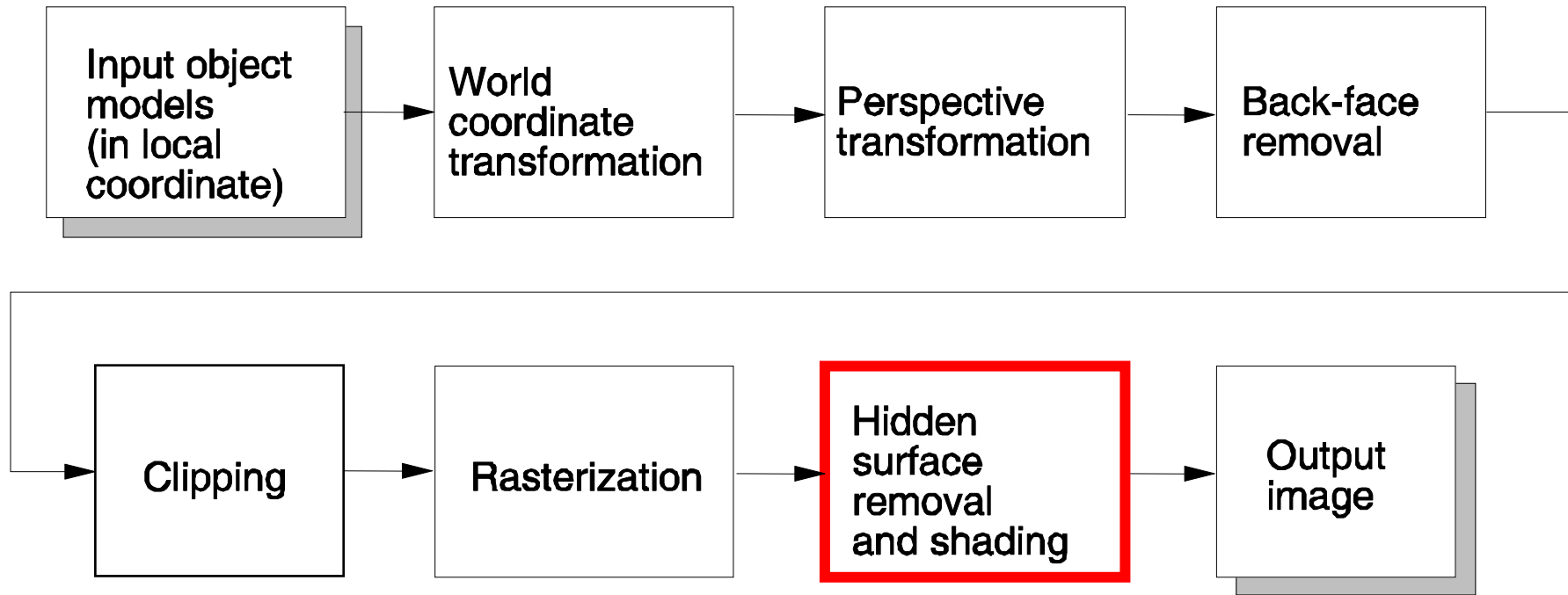
$$E_i(x, y) \geq 0 \quad (i = 1, 2, 3)$$





# The Rendering Pipeline

---



# The Rendering Pipeline

---

## □ Hidden surface removal

- The hidden surface removal step removes primitives or parts of primitives that are obscured by other primitives.
- Many methods have been developed for this purpose.
  - depth-sorting: one frame buffer, fail for conditions with intersecting polygons or cyclic overlap
  - z-buffering: two buffers (frame and depth buffers)
- The most popular one is the **z-buffer** method, which is implemented by majority of existing graphics accelerators.

# The Rendering Pipeline

## □ Shading

- uses the color calculation from an illumination model to determine the pixel colors for all projected positions in a scene.
- Flat shading
  - compute one intensity value for the whole polygon
- Smooth shading
  - Gouraud shading: interpolate color across triangles
  - Phong shading: interpolate normal across triangles

