CSE-170 Computer Graphics

Lecture 14

Clipping

Dr. Renato Farias
rfarias2@ucmerced.edu

Clipping Strategies

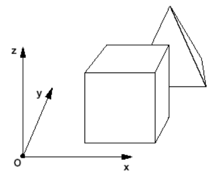
- What is it?
  - "Clip" all scene geometries to the current viewing frustum, so that what cannot be seen will not be rasterized

Remembering the Rendering Pipeline

Modeling Transformations ← ∀ Objects

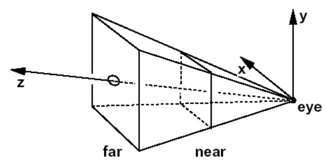
↓

Lighting



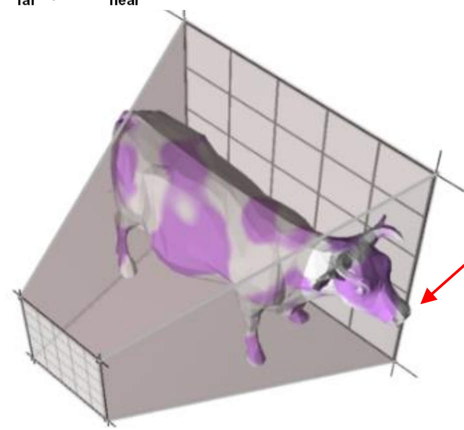
↓

Viewing/Camera Transformations



↓

Projection Transformation



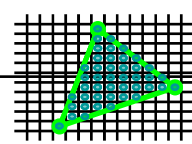
Non-visible triangles do not need to be rasterized

↓

Clipping

↓

Scan Conversion →



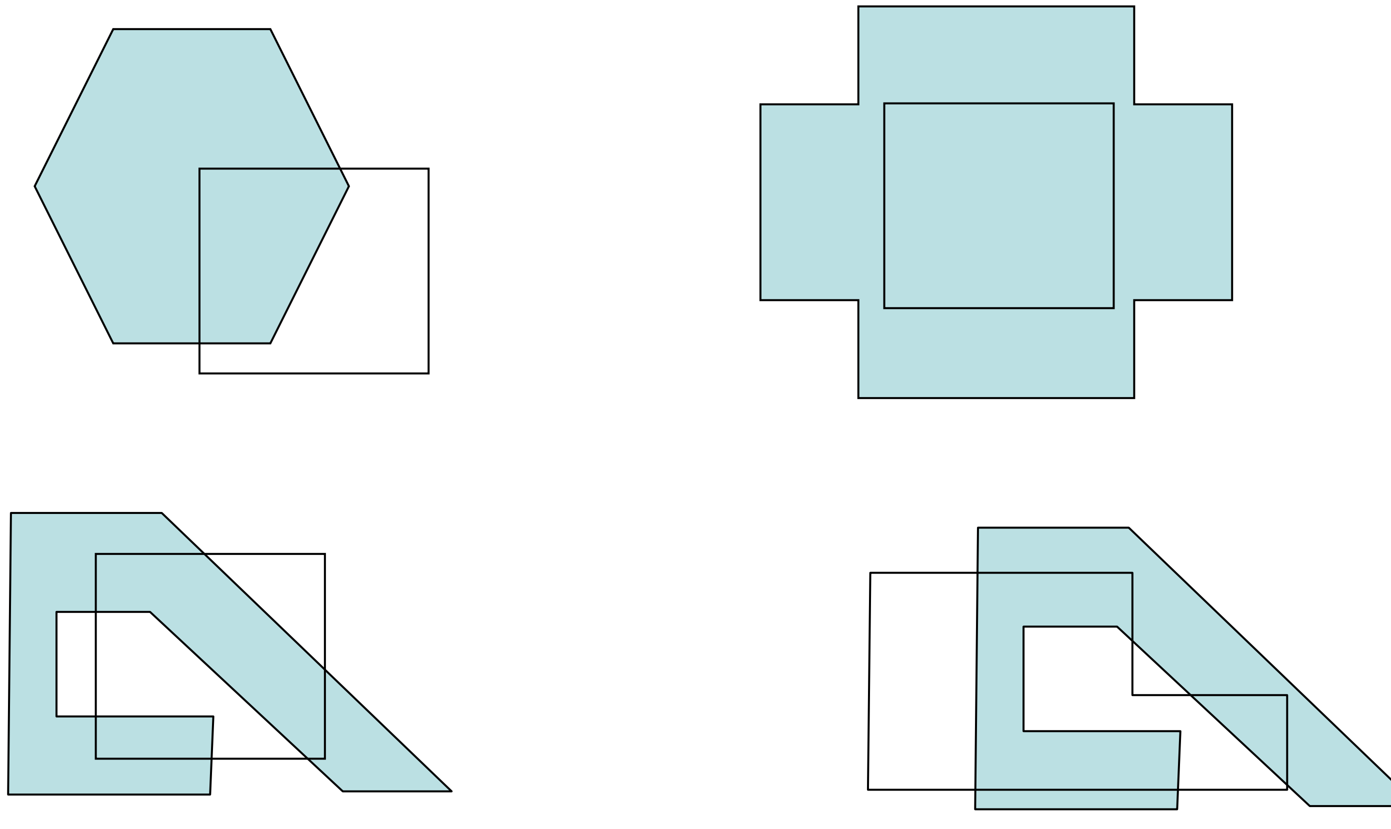
→ Image

Clipping Strategies

- Clipping can be done
  - In 3D, before rasterization (usually best)
    - Primitives are clipped in 3D
    - Good for when the camera is looking to a small area of a big environment, for ex., inside a room
  - In 2D, before rasterization
    - Project all primitives to the viewing plane, clip them in 2D and then rasterize them
  - During rasterization
    - Rasterize all primitives, and paint only the pixels inside the 2D clipping plane
    - Easier to implement, but usually slower

Clipping in 2D

- How would you clip generic shapes to a given region?



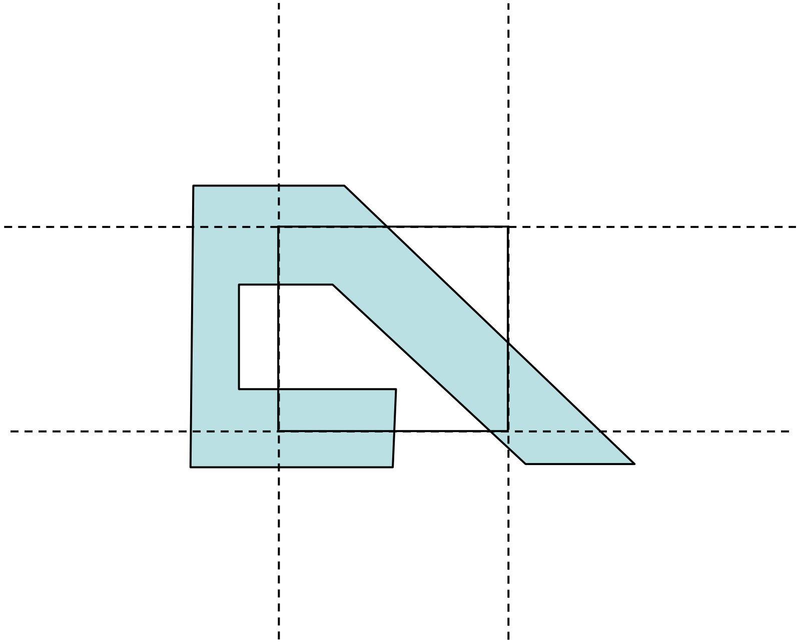
Clipping

- Main strategy to remember about clipping:

Break down a complex clipping problem
into several simple clipping procedures!

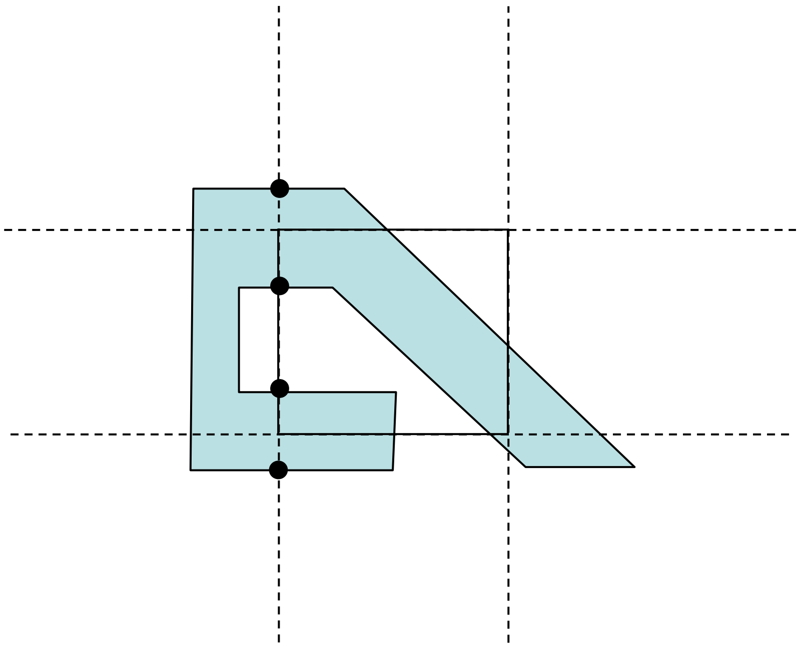
Clipping in 2D

- Sutherland-Hodgman Polygon Clipping Algorithm
  - Clip each polygon edge against each clip line
  - Keep track of the new boundary



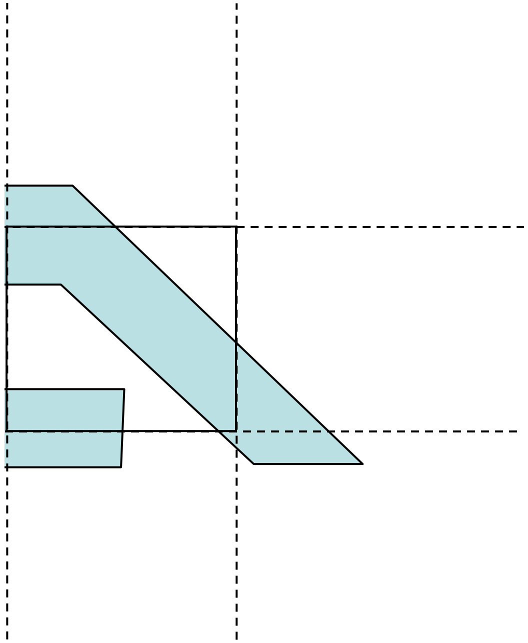
Clipping in 2D

- Sutherland-Hodgman Polygon Clipping Algorithm
  - Clip each polygon edge against each clip line
  - Keep track of the new boundary



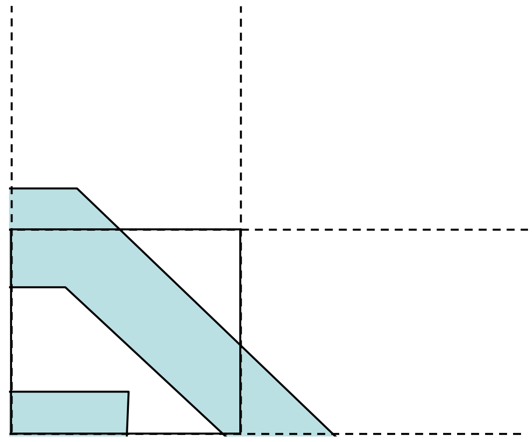
Clipping in 2D

- Sutherland-Hodgman Polygon Clipping Algorithm
  - Clip each polygon edge against each clip line
  - Keep track of the new boundary



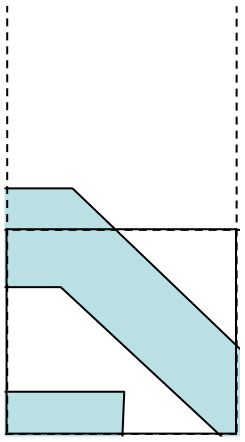
Clipping in 2D

- Sutherland-Hodgman Polygon Clipping Algorithm
  - Clip each polygon edge against each clip line
  - Keep track of the new boundary



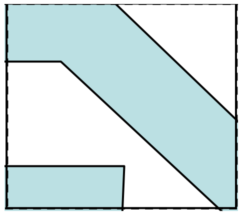
Clipping in 2D

- Sutherland-Hodgman Polygon Clipping Algorithm
  - Clip each polygon edge against each clip line
  - Keep track of the new boundary



Clipping in 2D

- Sutherland-Hodgman Polygon Clipping Algorithm
  - Clip each polygon edge against each clip line
  - Keep track of the new boundary

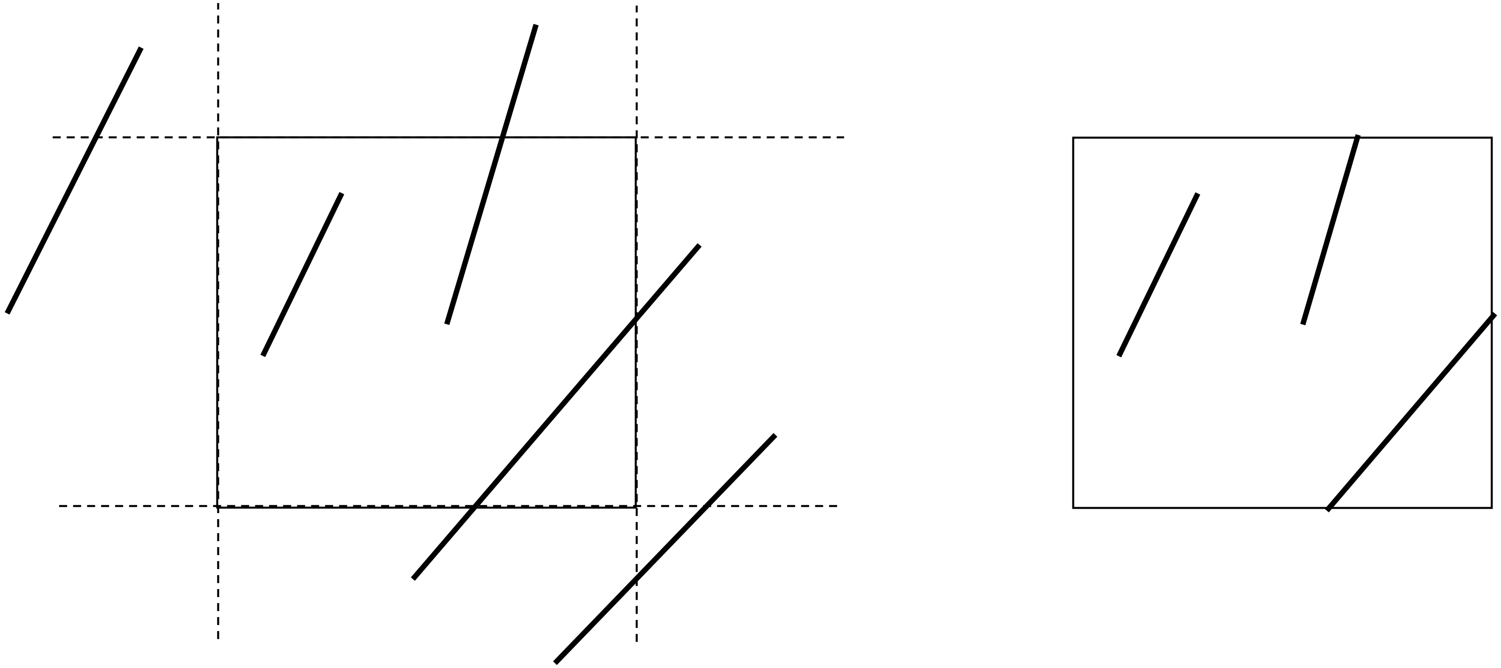


Clipping in 2D

- Clipping polygons can thus be (mainly) reduced to clipping lines

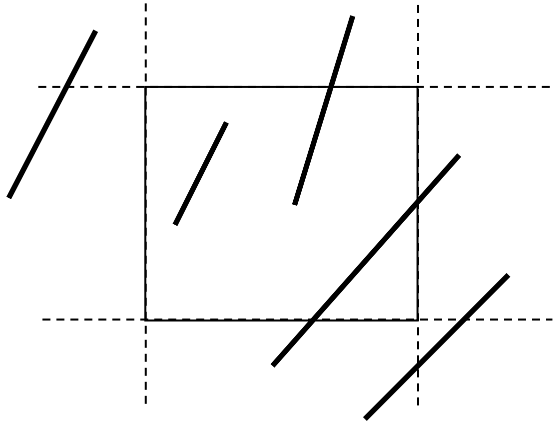
Clipping Lines

- Line clipping: check endpoints and compute appropriate intersections for each case
  - Brute force: compute/test all intersections



Clipping Lines

- Cohen-Sutherland Line-Clipping Algorithm
  1. Trivial region checks
     - Test if endpoints are trivially accepted (both inside)
     - Test trivial endpoint rejection (same side of a clip edge)
  2. Divide segment in two at a clip edge
     - One part is trivially rejected
     - Start algorithm again with the other part



Clipping Lines

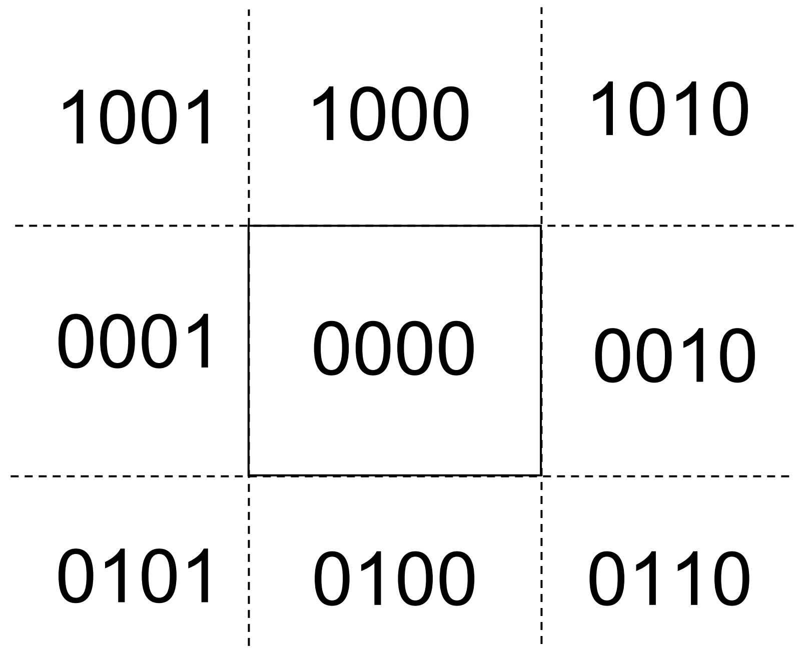
- Cohen-Sutherland Line-Clipping Algorithm
  - Bit code used to classify endpoints and recursively treat each case

1st bit: $y>y_{max}$

2nd bit: $y<y_{min}$

3rd bit: $x>x_{max}$

4th bit: $x<x_{min}$



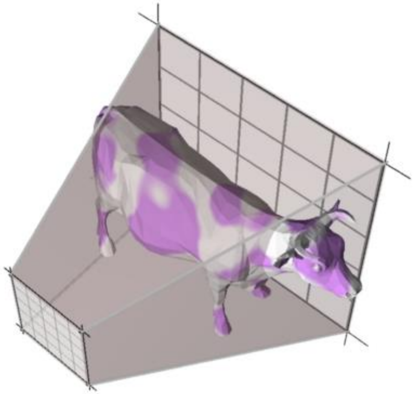
Clipping Lines

- Cyrus-Beck Parametric Clipping (1978)
  - More efficient
  - Based on parametric line representation
  - For each line to be clipped, computes *t* for all intersections with the four clip lines
    - Four *t* values are computed (in most cases)
  - A series of simple tests are enough to determine if there are actual intersections that have to be computed
  - Avoids repetitive looping of the Cohen-Sutherland algorithm

3D Line Clipping

- Clipping against our viewing frustum (truncated regular pyramid)

  – Both Cohen-Sutherland and Cyrus-Beck algorithms can be extended to 3D



3D Line Clipping

- Overall process
  - Compute clipping points for the 3 lines forming each triangle
  - Eliminate portions that are not visible
    - Triangles are simple to process
    - Result: (sub-)triangles completely visible
  - Send visible (sub-)triangles for rasterization

Midterm

Topics the midterm will cover

- Rendering pipeline (also comparison with ray tracing)
- Painter's algorithm, BSP trees, Z-buffer
- Vector and math algebra (meaning of operations)
- Transformations
- Barycentric coordinates
- Phong Illumination Model
- Flat and Gouraud shading, generation of normals
- Texture, Environment, Bump, and Disp. mapping
- Midpoint algorithm
- Use of scan lines for polygon rasterization
- Polygon and line clipping

Exam

- Questions
  - Most will be variations of the exercises
  - Some will ask about concepts seen in class
  - T/F questions at the end covering many concepts