

Lecture Set 4

## Line Clipping Algorithms

CS5600 Computer Graphics

Rich Riesenfeld  
Spring 2006

### Cases for Clipping Lines

Spring 2006      Utah School of Computing      2

### Cases for Clipping Lines

Spring 2006      Utah School of Computing      3

### Clipping Targets

- Points
- Lines
- Polygons
- Text
- Other Objects

Spring 2006      Utah School of Computing      4

## Clipping

- One of oldest problems in graphics
- We will study problem of Clip Line against an axis-oriented rectangular window
- Many variations of this problem

Spring 2006

Utah School of Computing

5

## General Problem

- Clip one object wrt another object
- Requires intersection calculations
- Interesting problem
- Floating point causes difficulties
- Relates to many other problems
- *Graphics does things fast...*

Spring 2006

Utah School of Computing

6

## Related Problem

- Classification Problem  
 $(x, y) \in \Omega$  ?
- Doing geometric calculations in floating point is tricky

Spring 2006

Utah School of Computing

7

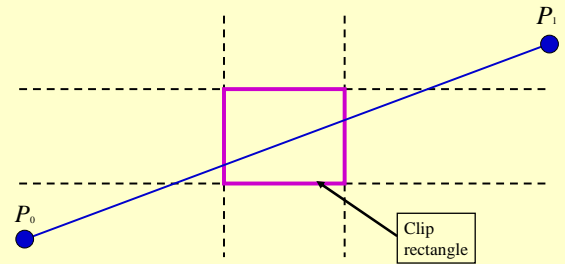
## *Recursive Subdivision Clipping*

### One of the Earliest Algorithms

- Recursive Subdivision (split line at midpoint)
- Eliminate pieces completely outside
- Good for binary processing
- Bounded number (10 or 12) of steps (by pixel size)

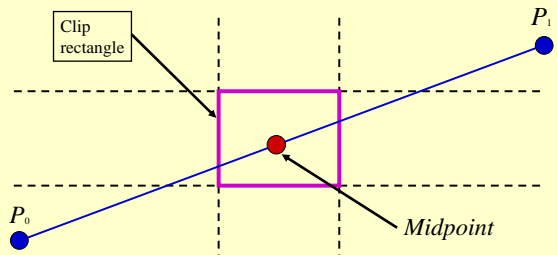
Spring 2006 Utah School of Computing 9

### Recursive Subdivision Clipping



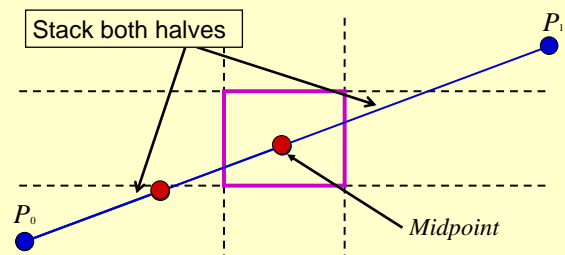
Spring 2006 Utah School of Computing 10

### Recursive Subdivision Clipping

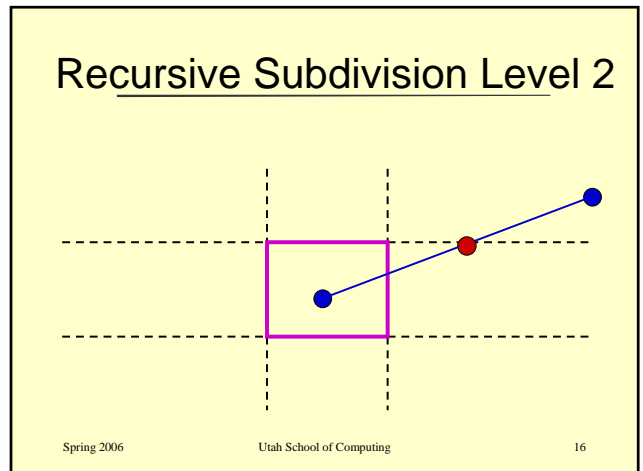
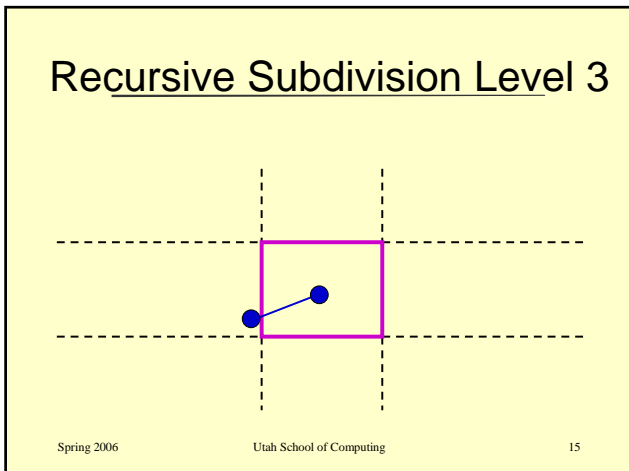
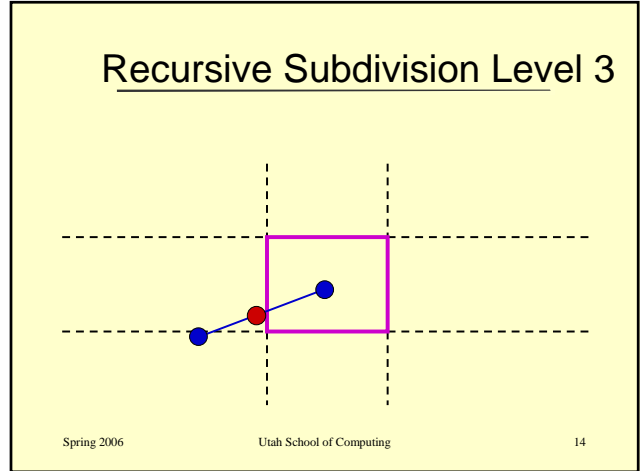
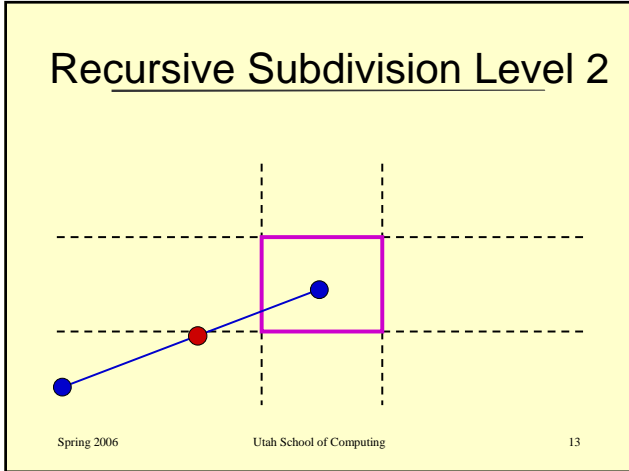


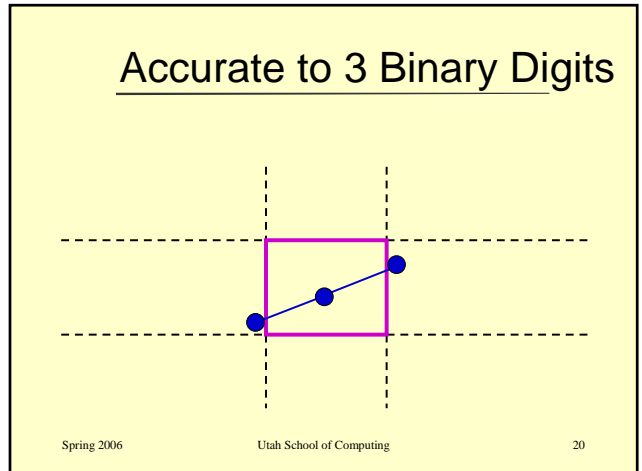
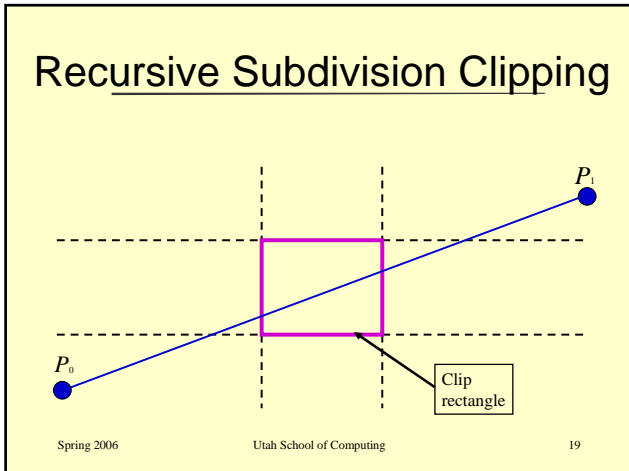
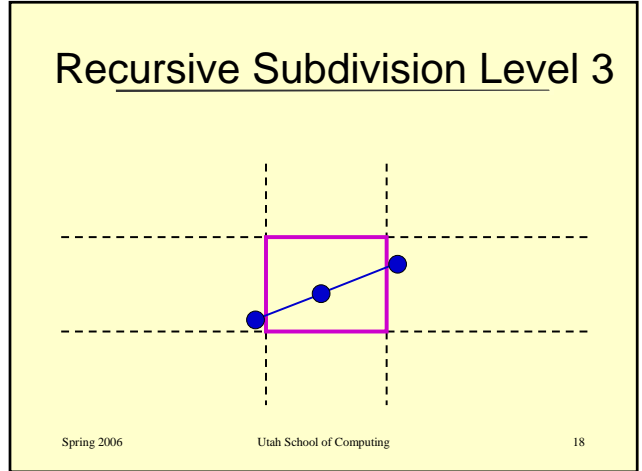
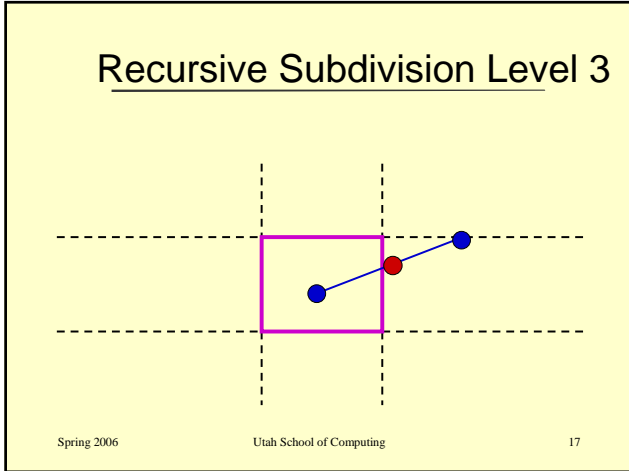
Spring 2006 Utah School of Computing 11

### Recursive Subdivision Level 1



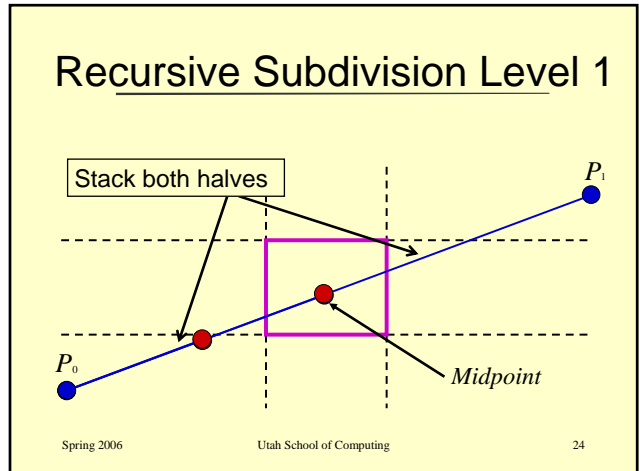
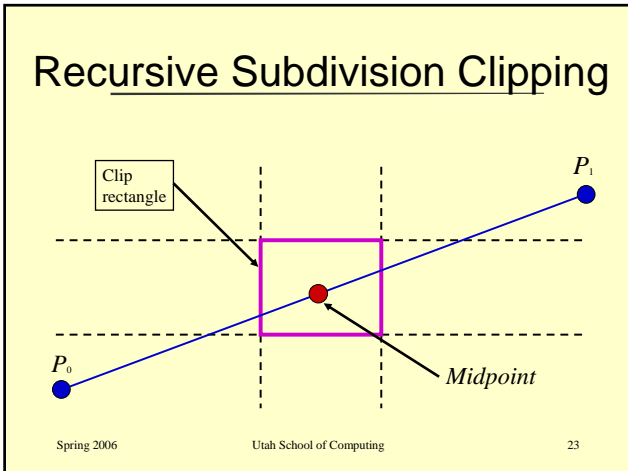
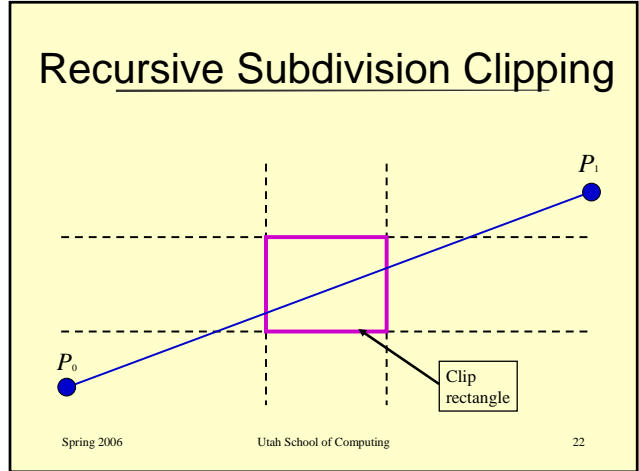
Spring 2006 Utah School of Computing 12

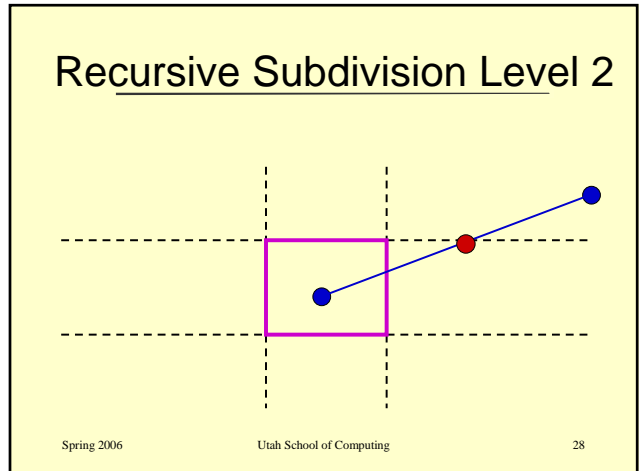
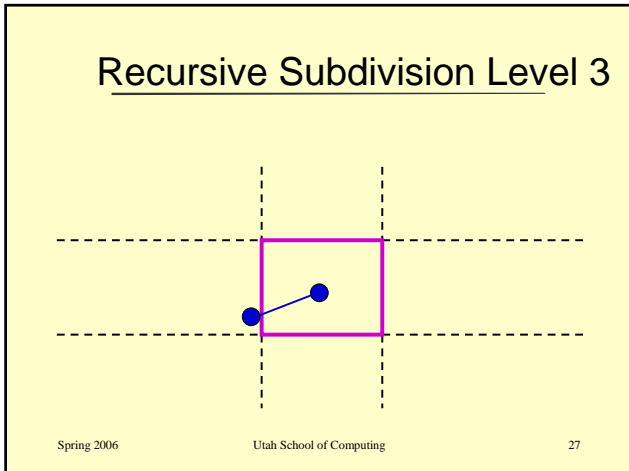
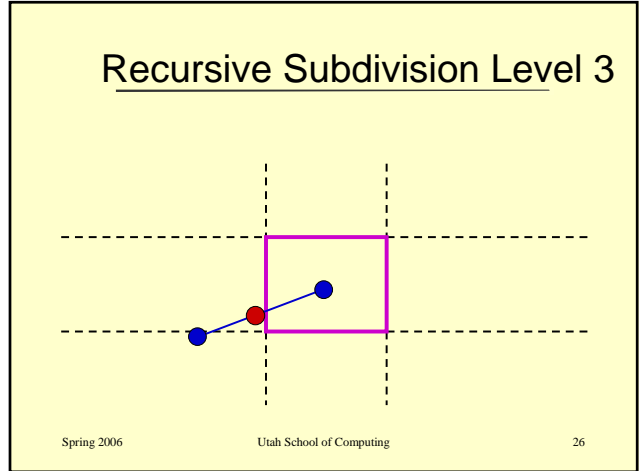
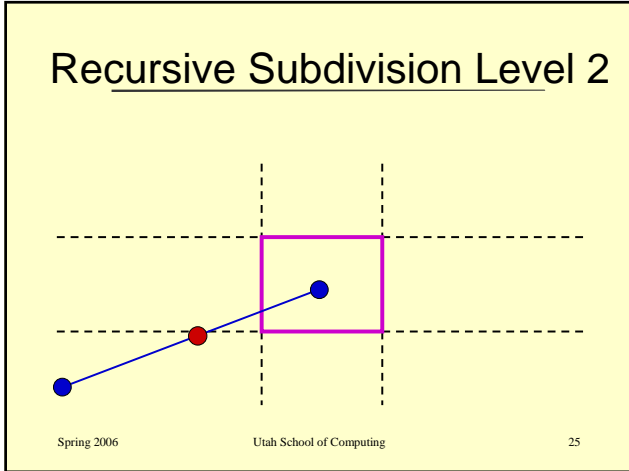


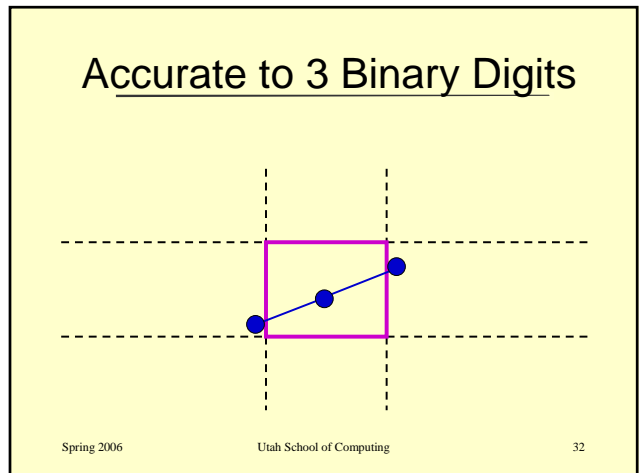
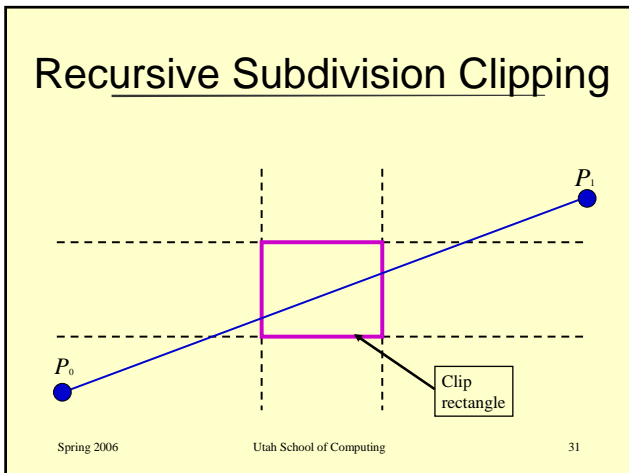
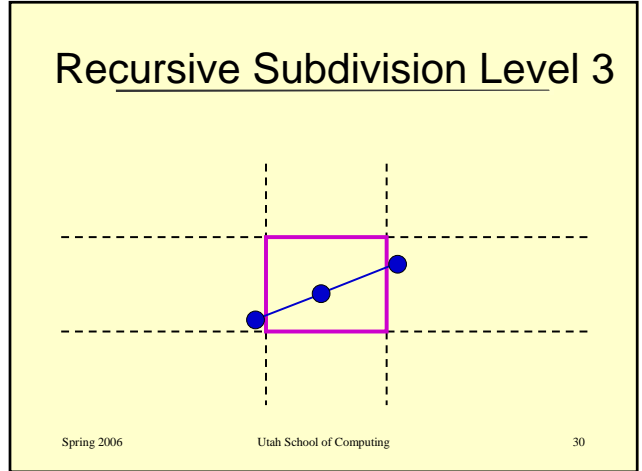
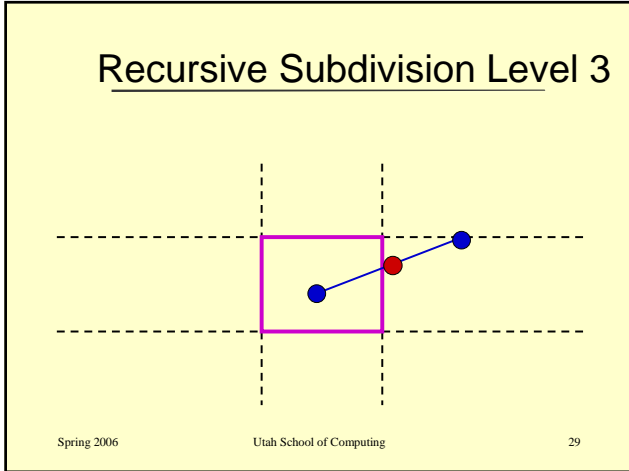


*Begin Animated Algorithm*

---







## End Animation

Spring 2006

Utah School of Computing

33

## Recursive Subdivision

- Linear Convergence Algorithm
- Computes 1 binary digit each time through loop
- Originally done with *shift* register

Spring 2006

Utah School of Computing

34

## Recursive Subdivision

- Old reliable method called *subdivision* in numerical analysis
- Stable
- Useful for nonrectangular shaped windows, e.g.

Spring 2006

Utah School of Computing

35

## Recall Parametric Line Eq

$$P(t) = (1-t)P_0 + tP_1$$

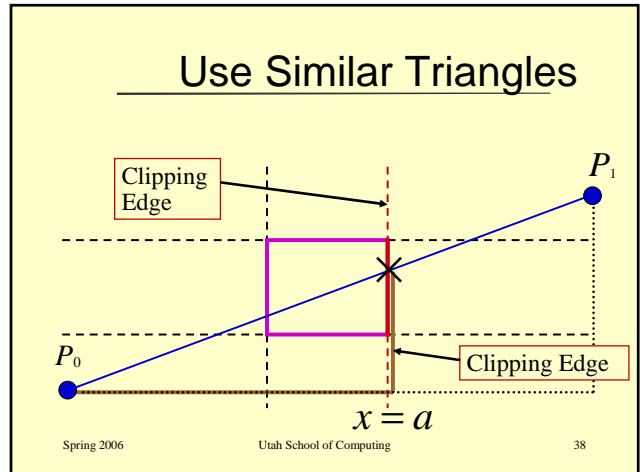
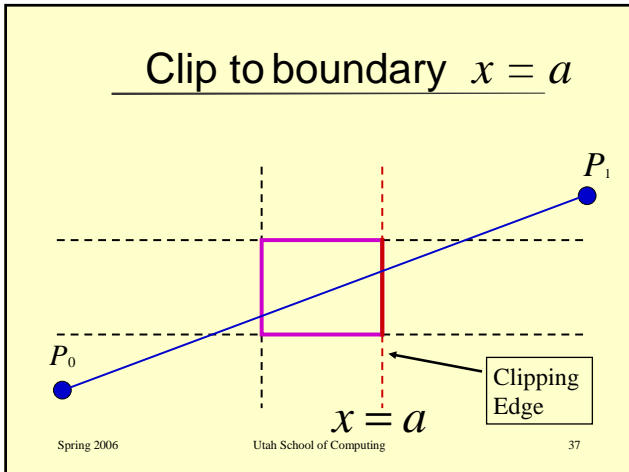
where,

$$P(0) = P_0 ; \quad P(1) = P_1$$

Spring 2006

Utah School of Computing

36



### Use Similar Triangles

Use ratio of these lines,

{ }

That is, 
$$t' = \frac{a - x_0}{x_1 - x_0}$$

And, similarly for explicit lines

Spring 2006 Utah School of Computing 39

## Cohen-Sutherland Line Clipping

---

### Region Outcodes

$x_{\min}$                        $x_{\max}$   
 $y_{\min}$                        $y_{\max}$

Spring 2006                      Utah School of Computing                      41

### Look at Sign Bits ( $neg \Rightarrow 1$ )

- Bit 1  $\leftarrow sign(y_{\max} - y)$
- Bit 2  $\leftarrow sign(y - y_{\min})$
- Bit 3  $\leftarrow sign(x_{\max} - x)$
- Bit 4  $\leftarrow sign(x - x_{\min})$

Spring 2006                      Utah School of Computing                      42

### Need to *Classify* Endpoint

- Look at  $C_0 \wedge C_1$
- What does it tell us?
- $C_0 \wedge C_1 \neq 0 \Rightarrow$  “trivial reject”
- Both ends in a row or column outside

Spring 2006                      Utah School of Computing                      43

### Region Outcodes

<i>Bit</i> <sub>1</sub>	<i>Bit</i> <sub>2</sub>	<i>Bit</i> <sub>3</sub>	<i>Bit</i> <sub>4</sub>
-------------------------	-------------------------	-------------------------	-------------------------

- *Bit* 1 = *t*  $\Rightarrow$  above window
- *Bit* 2 = *t*  $\Rightarrow$  below window
- *Bit* 3 = *t*  $\Rightarrow$  right of window
- *Bit* 4 = *t*  $\Rightarrow$  left of window

Spring 2006                      Utah School of Computing                      44

### Classify Endpoint

$\{C_0 \wedge C_1 = 0\} \Rightarrow$  Endpoints may not be in window

Clip an end for which  $C_i \neq 0$

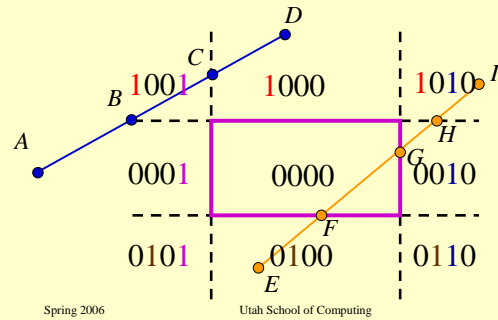
Test again. May need more.

Spring 2006

Utah School of Computing

45

### Cohen-Sutherland Line Clipping

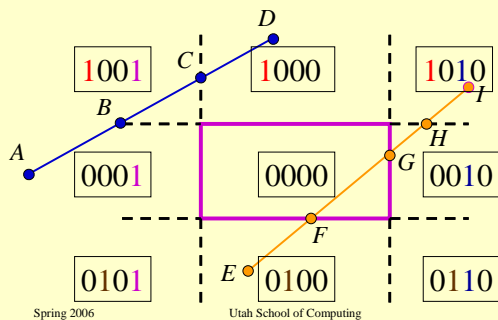


Spring 2006

Utah School of Computing

46

### Cohen-Sutherland Line Clipping



Spring 2006

Utah School of Computing

47

### Initial Outcode Calculations

- $OC(D)=1000; OC(A)=0001$   
 $1000 \wedge 0001 = 0000$
- $OC(E)=0100; OC(I)=1010$   
 $0100 \wedge 1010 = 0000$

Spring 2006

Utah School of Computing

48

### Clip and Continue

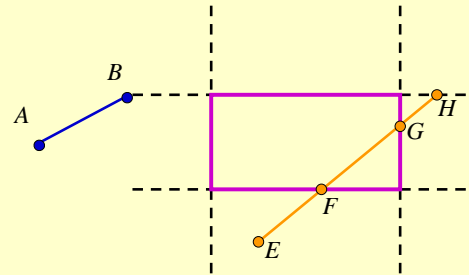
- Clip against the top boundary  
 $y = y_{\max}$
- Calculate  $B$ . Keep  $AB$
- Calculate  $H$ . Keep  $EH$

Spring 2006

Utah School of Computing

49

### Cohen-Sutherland Line Clipping



Spring 2006

Utah School of Computing

50

### Clip and Continue

- Clip against the bottom boundary  
 $y = y_{\min}$
- Now test and reject  $AB$  because
- $OC(A)=0001$  and  $OC(B)=0001$ ;  
 $0001 \wedge 0001 = 0001 \neq 0$
- Reject  $AB$  on outcode basis

Spring 2006

Utah School of Computing

51

### Outcode Calculations

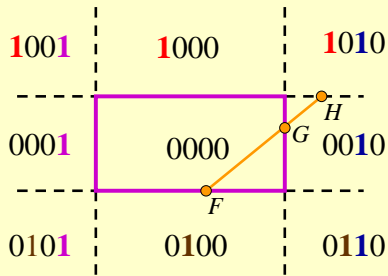
- $OC(H)=0010$ ;  $OC(E)=0100$   
 $0010 \wedge 0100 = 0000$
- Since product is 0, process  
 $HE$  to get  $FH$

Spring 2006

Utah School of Computing

52

### Cohen-Sutherland Line Clipping



Spring 2006

Utah School of Computing

53

### Outcode Calculations

- $OC(F)=0000$ ;  $OC(H)=0010$   
 $0010 \wedge 0100 = 0000$
- Since product is 0, process  $HF$  to get  $GF$

Spring 2006

Utah School of Computing

54

### Clip and Continue

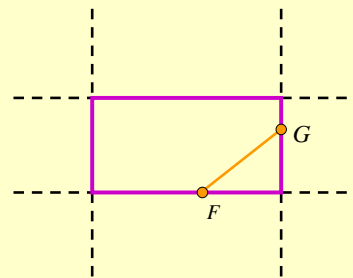
- Clip against the right boundary  
 $x = x_{\max}$
- Get  $GF$
- Done

Spring 2006

Utah School of Computing

55

### Cohen-Sutherland Line Clipping



Spring 2006

Utah School of Computing

56

### Process Ends

- Do not have to do  $x_{min}$
- Normally it would continue to last stage

Spring 2006

Utah School of Computing

57

### When is this algorithm good?

- If it *trivially rejects(accepts)* most cases
- Good if window large wrt to data
- Good if window small wrt to data
- Eg, it works well in *extreme* cases with mostly trivial decisions

Spring 2006

Utah School of Computing

58

### Another Digital Line Issue

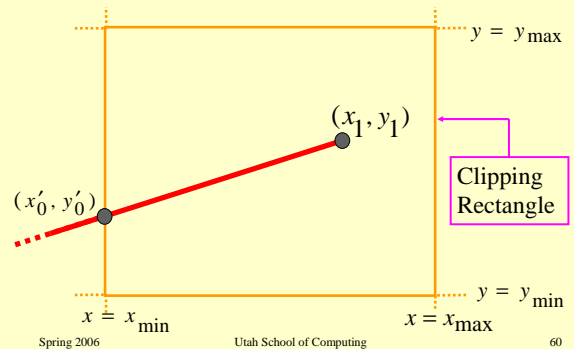
- Clipping Bresenham lines
- The integer slope is not the true slope
- Have to be careful
- More issues to follow

Spring 2006

Utah School of Computing

59

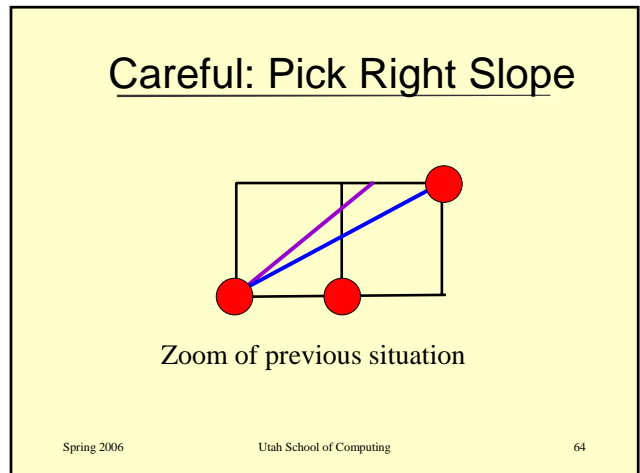
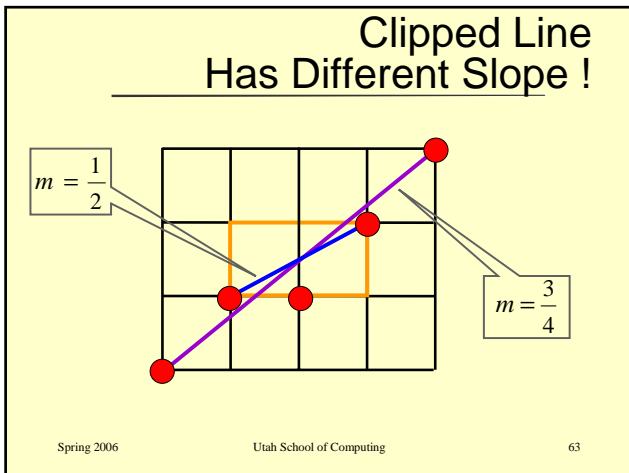
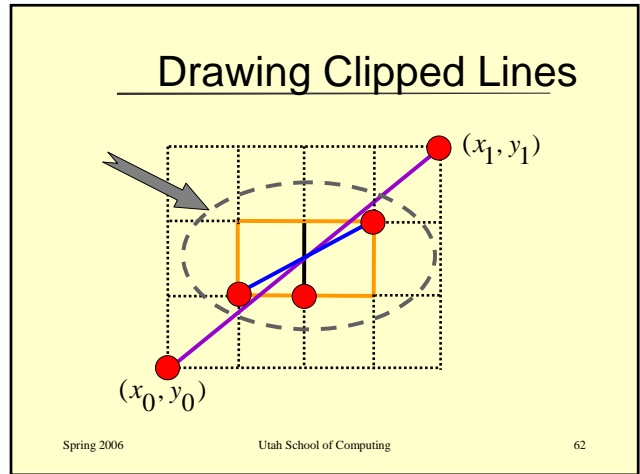
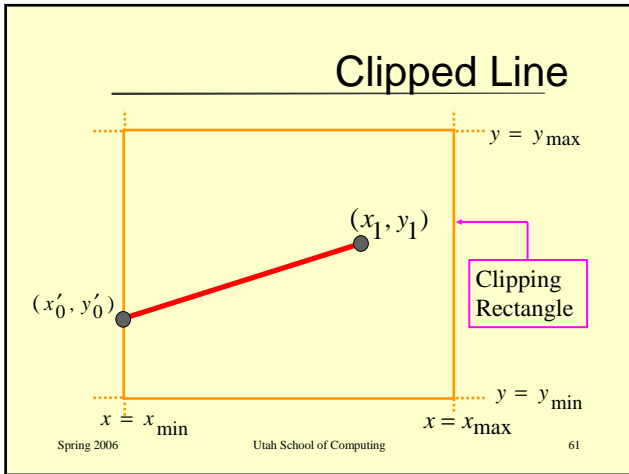
### Line Clipping Problem

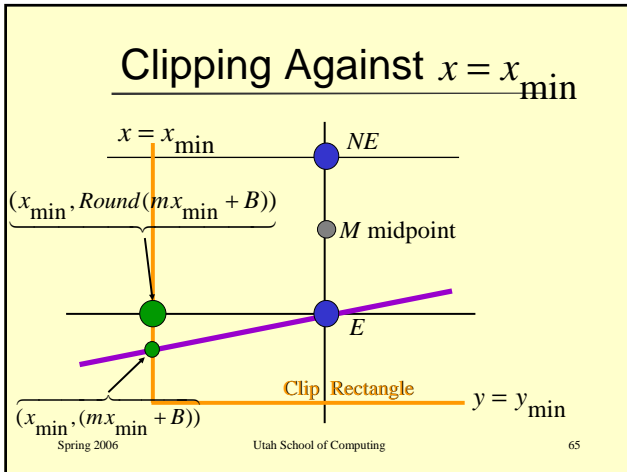


Spring 2006

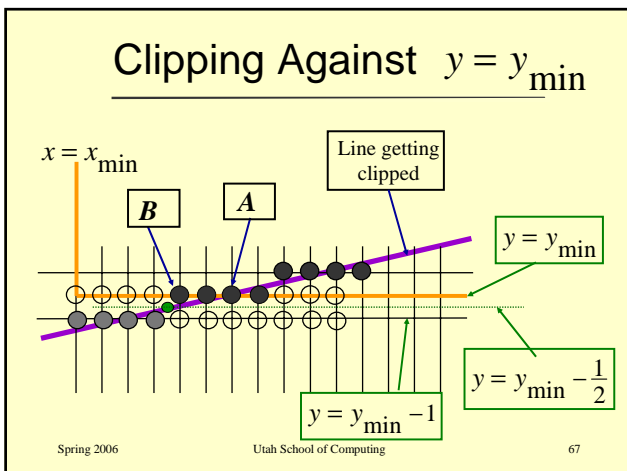
Utah School of Computing

60





- ### Generating Exact Segment
- Clipped line segment must share same entry point as unclipped Bresenham line
  - Clipped line must also employ the identical  $d$ -value as unclipped line
- Spring 2006      Utah School of Computing      66



- ### Clipping Against $y = y_{\min}$
- Situation is complicated
  - Multiple pixels involved at  $y = y_{\min}$
  - Want all of those pixels as "in"
  - Analytic  $\cap$ , rounding  $x$  gives  $A$
  - We want point  $B$
- Spring 2006      Utah School of Computing      68

## Clipping Against: $y = y_{\min} -$

---

- Use  $\text{Line} \cap y = y_{\min} - \frac{1}{2}$
- Round *up* to nearest integer  $x$
- This yields point  $B$ , the desired result

Spring 2006

Utah School of Computing

69

The End  
*Line Clipping*

---

Lecture Set 4