

Costruzione di Interfacce Lezione 10 Clipping e HSR

cignoni@iei.pi.cnr.it
<http://vcq.iei.pi.cnr.it/~cignoni>

Oggi parleremo di...

- ❖ Clipping di segmenti
- ❖ Algoritmo di Cohen-Sutherland
- ❖ Clipping di poligoni
- ❖ Algoritmo di Sutherland-Hodgman
- ❖ Eliminazione linee nascoste

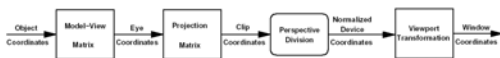
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

2

Clipping

- ❖ Sino ad adesso abbiamo ignorato che lo schermo (la finestra dell'applicazione) fosse una superficie discreta ma di dimensione finita
- ❖ Questo comporta la necessità di fare **clipping** delle primitive che si rasterizzano
- ❖ Fare **clipping** significa identificare le porzioni della primitiva che sono visibili e quelle che non lo sono



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

3

Clipping

- ❖ Il problema del clipping è genericamente risolvibile per qualunque superficie chiusa
- ❖ A noi però interessa risolvere solo il problema di fare clipping su rettangoli, dato che le porzioni di schermo che l'applicazione gestisce sono rettangoli

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

4

Clipping di un punto

- ❖ Un punto si trova all'interno del rettangolo di clipping se sono soddisfatte le 4 disuguaglianze:

$$x_{\min} \leq x \leq x_{\max}, y_{\min} \leq y \leq y_{\max}$$

Qualora una qualsiasi di queste disuguaglianze non valesse il punto è al di fuori del rettangolo di clipping

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

5

Clipping di un segmento

- ❖ Per fare il clipping di un segmento, si considera la posizione dei suoi punti estremi:
 - ❖ Se i due punti sono entrambi dentro, l'intero segmento lo è
 - ❖ Se un punto è dentro e uno fuori il segmento interseca il rettangolo e si deve calcolare l'intersezione
 - ❖ Se entrambi i punti sono fuori dal rettangolo il segmento può o non può intersecare il rettangolo e si devono fare altri calcoli per determinare se le intersezioni ci sono, e se ci sono dove sono

23 Ott 2002

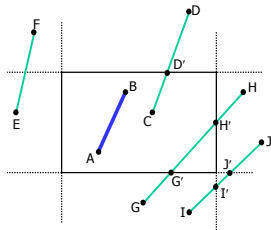
Costruzione di Interfacce - Paolo Cignoni

6

Clipping di un segmento

Per fare il clipping di un segmento, si considera la posizione dei suoi punti estremi:

- ❖ Se i due punti sono entrambi dentro, l'intero segmento (AB) lo è



23 Ott 2002

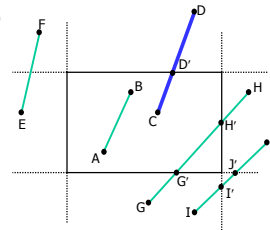
Costruzione di Interfacce - Paolo Cignoni

7

Clipping di un segmento

Per fare il clipping di un segmento, si considera la posizione dei suoi punti estremi:

- ❖ Se un punto è dentro e uno fuori il segmento (CD) interseca il rettangolo e si deve calcolare l'intersezione (D')



23 Ott 2002

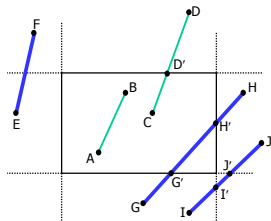
Costruzione di Interfacce - Paolo Cignoni

8

Clipping di un segmento

Per fare il clipping di un segmento, si considera la posizione dei suoi punti estremi:

- ❖ Se entrambi i punti sono fuori dal rettangolo (EF, GH, IJ) il segmento può o non può intersecare il rettangolo e si devono fare altri calcoli per determinare se le intersezioni ci sono, e se ci sono dove sono



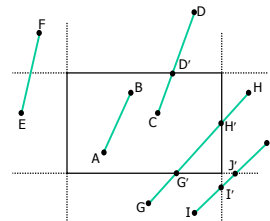
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

9

Clipping di un segmento

- ❖ L'approccio più diretto alla soluzione del problema sarebbe quello di fare il calcolo delle intersezioni tra la retta su cui giace il segmento e le 4 rette su cui giacciono i lati del rettangolo di clipping



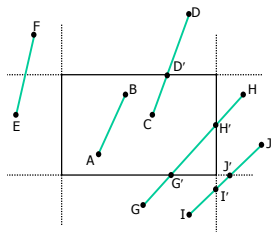
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

10

Clipping di un segmento

- ❖ Una volta individuati i punti di intersezione verificare poi se essi appartengono effettivamente al segmento ed al lato (G' e H') oppure no (I' e J').



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

11

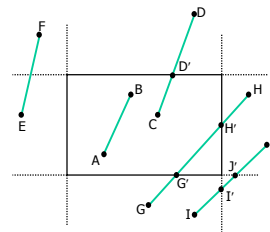
Clipping di un segmento

- ❖ Questo calcolo si può facilmente compiere utilizzando l'equazione parametrica della retta:

$$x = x_0 + t(x_1 - x_0)$$

$$y = y_0 + t(y_1 - y_0)$$

$$t \in [0,1]$$



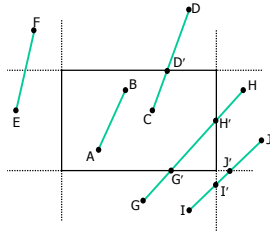
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

12

Clipping di un segmento

- Le rette si intersecano se, dopo aver risolto contemporaneamente e i due insiemi di equazioni che descrivono il segmento ed il lato in t_{lato} e t_{segm} i due valori rientrano entrambi nell'intervallo $[0, 1]$



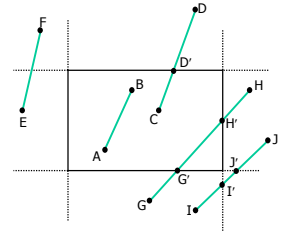
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

13

Clipping di un segmento

- Si dovrebbe inoltre verificare in anticipo se per caso le linee sono parallele prima di calcolare l'intersezione
- Un algoritmo del genere funziona ma è costoso e quindi inefficiente



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

14

Algoritmo di Cohen-Sutherland

- L'algoritmo di clipping di Cohen-Sutherland si basa su un approccio completamente diverso
- La considerazione di base che si fa è che le rette che delimitano il rettangolo che definisce la regione di clipping suddividono il piano in nove regioni

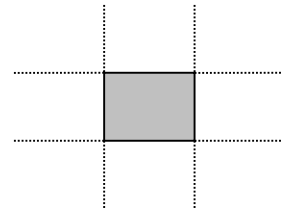
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

15

Algoritmo di Cohen-Sutherland

- La considerazione di base che si fa è che le rette che delimitano il rettangolo che definisce la regione di clipping suddividono il piano in nove regioni



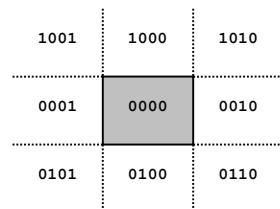
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

16

Algoritmo di Cohen-Sutherland

- Ad ogni regione viene associato un codice numerico di quattro cifre binarie



23 Ott 2002

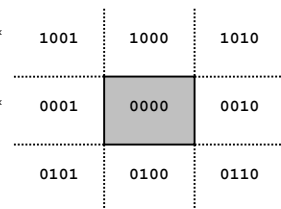
Costruzione di Interfacce - Paolo Cignoni

17

Algoritmo di Cohen-Sutherland

- Il codice è formato da 4 bit, vero o falso:

- bit 1 sopra l'edge in alto $y > y_{\text{max}}$
- bit 2 sotto l'edge in basso $y < y_{\text{min}}$
- bit 3 a dx dell'edge di dx $x > x_{\text{max}}$
- bit 4 a sx dell'edge di sx $x < x_{\text{min}}$



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

18

Algoritmo di Cohen-Sutherland

❖ Il codice è formato da 4 bit, vero o falso:

❖ bit 1 sopra l'edge in alto $y > y_{max}$	1001	1000	1010
❖ bit 2 sotto l'edge in basso $y < y_{min}$			
❖ bit 3 a dx dell'edge di dx $x > x_{max}$	0001	0000	0010
❖ bit 4 a sx dell'edge di sx $x < x_{min}$	0101	0100	0110

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

19

Algoritmo di Cohen-Sutherland

❖ Il codice è formato da 4 bit, vero o falso:

❖ bit 1 sopra l'edge in alto $y > y_{max}$	1001	1000	1010
❖ bit 2 sotto l'edge in basso $y < y_{min}$			
❖ bit 3 a dx dell'edge di dx $x > x_{max}$	0001	0000	0010
❖ bit 4 a sx dell'edge di sx $x < x_{min}$	0101	0100	0110

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

20

Algoritmo di Cohen-Sutherland

❖ Il codice è formato da 4 bit, vero o falso:

❖ bit 1 sopra l'edge in alto $y > y_{max}$	1001	1000	1010
❖ bit 2 sotto l'edge in basso $y < y_{min}$			
❖ bit 3 a dx dell'edge di dx $x > x_{max}$	0001	0000	0010
❖ bit 4 a sx dell'edge di sx $x < x_{min}$	0101	0100	0110

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

21

Algoritmo di Cohen-Sutherland

❖ Il codice è formato da 4 bit, vero o falso:

❖ bit 1 sopra l'edge in alto $y > y_{max}$	1001	1000	1010
❖ bit 2 sotto l'edge in basso $y < y_{min}$			
❖ bit 3 a dx dell'edge di dx $x > x_{max}$	0001	0000	0010
❖ bit 4 a sx dell'edge di sx $x < x_{min}$	0101	0100	0110

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

22

Algoritmo di Cohen-Sutherland

❖ Con queste premesse l'operazione di clipping si risolve con opportune codifiche e confronti tra codici numerici

1001	1000	1010
0001	0000	0010
0101	0100	0110

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

23

Algoritmo di Cohen-Sutherland

❖ Per fare il clipping di un segmento si codificano i suoi punti estremi sulla base della regione del piano a cui appartengono e poi si confrontano i due codici

1001	1000	1010
0001	0000	0010
0101	0100	0110

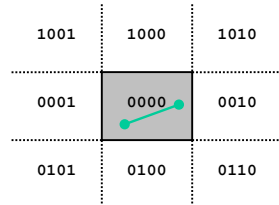
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

24

Algoritmo di Cohen-Sutherland

- Se il codice di entrambi i punti estremi è 0000, allora si può banalmente decidere che il segmento è interamente all'interno



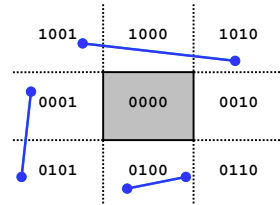
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

25

Algoritmo di Cohen-Sutherland

- Altrettanto banalmente si può decidere che un segmento è tutto all'esterno quando l'operazione di AND logico tra i codici dei due punti ha un risultato diverso da 0



$$0001 \wedge 0100 = 0000$$

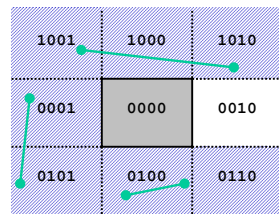
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

26

Algoritmo di Cohen-Sutherland

- In questo caso, infatti, entrambi i punti stanno in uno stesso semipiano (quello identificato dal bit a 1 del risultato) e quindi il segmento non interseca il rettangolo di clipping



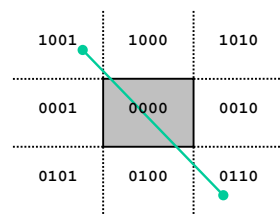
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

27

Algoritmo di Cohen-Sutherland

- Se il risultato dell'AND è invece 0000



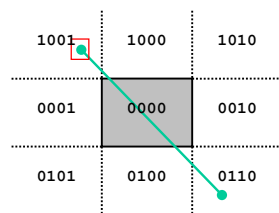
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

28

Algoritmo di Cohen-Sutherland

- Se il risultato dell'AND è invece 0000
 - si cerca quale dei due punti estremi giace fuori dal rettangolo (almeno uno lo è)



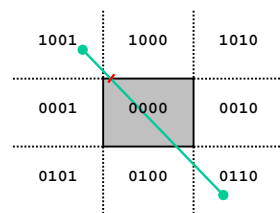
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

29

Algoritmo di Cohen-Sutherland

- Se il risultato dell'AND è invece 0000
 - si cerca quale dei due punti estremi giace fuori dal rettangolo (almeno uno lo è)
 - si suddivide il segmento in due parti calcolando l'intersezione del segmento con i bordi del rettangolo



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

30

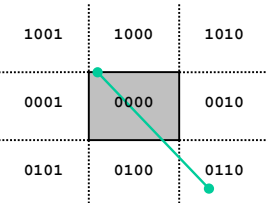
Algoritmo di Cohen-Sutherland

❖ Se il risultato dell'AND è invece 0000

❖ si cerca quale dei due punti estremi giace fuori dal rettangolo (almeno uno lo è)

❖ si suddivide il segmento in due parti calcolando l'intersezione del segmento con i bordi del rettangolo

❖ una si scarta (poiché giace fuori dal rettangolo) e si itera il procedimento



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

31

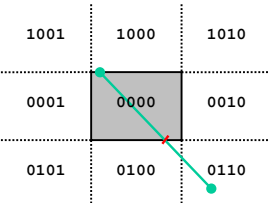
Algoritmo di Cohen-Sutherland

❖ Se il risultato dell'AND è invece 0000

❖ si cerca quale dei due punti estremi giace fuori dal rettangolo (almeno uno lo è)

❖ si suddivide il segmento in due parti calcolando l'intersezione del segmento con i bordi del rettangolo

❖ una si scarta (poiché giace fuori dal rettangolo) e si itera il procedimento



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

32

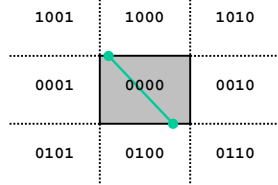
Algoritmo di Cohen-Sutherland

❖ Se il risultato dell'AND è invece 0000

❖ si cerca quale dei due punti estremi giace fuori dal rettangolo (almeno uno lo è)

❖ si suddivide il segmento in due parti calcolando l'intersezione del segmento con i bordi del rettangolo

❖ una si scarta (poiché giace fuori dal rettangolo) e si itera il procedimento



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

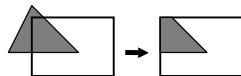
33

Clipping di poligoni

Clipping di poligoni

❖ Fare clipping di un poligono è un'operazione più complessa che farlo di un segmento dato che i casi da trattare sono vari e diversi tra loro

❖ Poligono convesso



23 Ott 2002

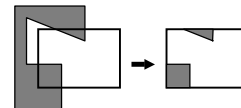
Costruzione di Interfacce - Paolo Cignoni

35

Clipping di poligoni

❖ Fare clipping di un poligono è un'operazione più complessa che farlo di un segmento dato che i casi da trattare sono vari e diversi tra loro

❖ Poligono concavo che clippato si divide



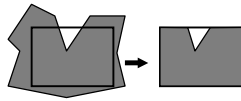
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

36

Clipping di poligoni

- ❖ Fare clipping di un poligono è un'operazione più complessa che farlo di un segmento dato che i casi da trattare sono vari e diversi tra loro
- ❖ Poligono con molti spigoli esterni



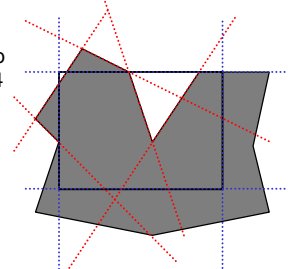
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

37

Clipping di poligoni

- ❖ L'approccio iniziale di soluzione potrebbe essere quello di confrontare ogni lato del poligono con le 4 rette che delimitano il rettangolo di clipping



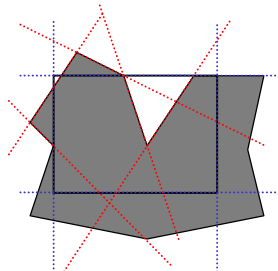
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

38

Clipping di poligoni

- ❖ Un approccio di questo tipo porta a compiere molte operazioni (costose quali i calcoli di intersezioni) di cui una gran parte possono essere inutili



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

39

Algoritmo di Sutherland-Hodgman

- ❖ L'idea alla base dell'algoritmo di Sutherland-Hodgman è applicare una strategia risolutiva del tipo *divide et impera*
- ❖ Il problema viene così ricondotto a quello di calcolare il clipping di un poligono qualsiasi rispetto ad una retta

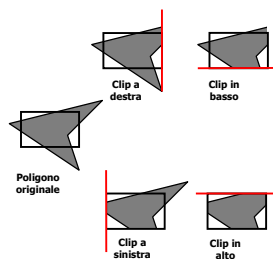
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

40

Algoritmo di Sutherland-Hodgman

- ❖ L'applicazione, in successione, della procedura alle 4 rette che definiscono il rettangolo di clipping avrà come risultato il clipping del poligono sul rettangolo



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

41

Algoritmo di Sutherland-Hodgman

- ❖ L'algoritmo riceve in ingresso una serie di vertici v_1, v_2, \dots, v_n che definiscono n spigoli: gli $n-1$ da v_i a v_{i+1} e quello da v_n a v_1
- ❖ Dopo aver fatto clipping su una retta ritorna in output un'altra serie di vertici che definiscono il poligono clippato

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

42

Algoritmo di Sutherland-Hodgman

- ❖ Il confronto si effettua scandendo il poligono in senso orario partendo dal vertice v_n fino a v_1 per poi tornare a v_n
- ❖ Per ogni passo si confronta la relazione esistente tra due vertici successivi e la retta di clipping
- ❖ Per ogni caso i vertici indicati con \square verranno inseriti nella lista di output

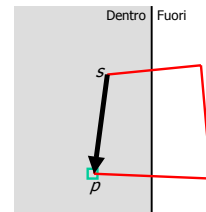
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

43

Algoritmo di Sutherland-Hodgman

- ❖ Le relazioni possono essere di 4 tipi:
 - ❖ Spigolo tutto interno



Caso 1

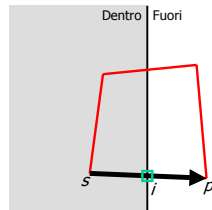
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

44

Algoritmo di Sutherland-Hodgman

- ❖ Le relazioni possono essere di 4 tipi:
 - ❖ Spigolo tutto interno
 - ❖ Spigolo uscente



Caso 2

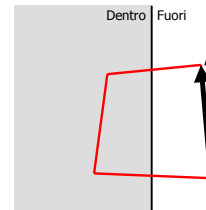
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

45

Algoritmo di Sutherland-Hodgman

- ❖ Le relazioni possono essere di 4 tipi:
 - ❖ Spigolo tutto interno
 - ❖ Spigolo uscente
 - ❖ Spigolo tutto esterno



Caso 3

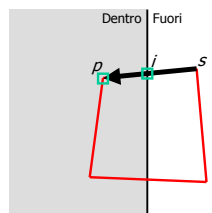
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

46

Algoritmo di Sutherland-Hodgman

- ❖ Le relazioni possono essere di 4 tipi:
 - ❖ Spigolo tutto interno
 - ❖ Spigolo uscente
 - ❖ Spigolo tutto esterno
 - ❖ Spigolo entrante



Caso 4

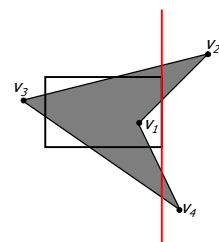
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

47

Esempio

- ❖ Vediamo come opera l'algoritmo nel caso in figura
- ❖ I vertici in nero sono gli originali, quelli in arancio sono originati dall'algoritmo di clipping sulla retta rossa



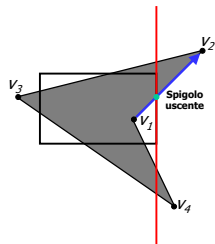
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

48

Esempio

- ❖ Vediamo come opera l'algoritmo nel caso in figura
- ❖ I vertici in nero sono gli originali, quelli in arancio sono originati dall'algoritmo di clipping sulla retta rossa



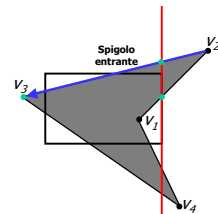
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

49

Esempio

- ❖ Vediamo come opera l'algoritmo nel caso in figura
- ❖ I vertici in nero sono gli originali, quelli in arancio sono originati dall'algoritmo di clipping sulla retta rossa



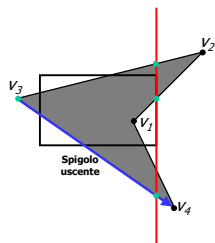
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

50

Esempio

- ❖ Vediamo come opera l'algoritmo nel caso in figura
- ❖ I vertici in nero sono gli originali, quelli in arancio sono originati dall'algoritmo di clipping sulla retta rossa



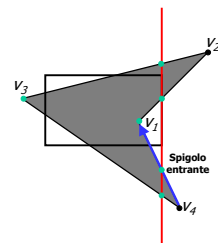
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

51

Esempio

- ❖ Vediamo come opera l'algoritmo nel caso in figura
- ❖ I vertici in nero sono gli originali, quelli in arancio sono originati dall'algoritmo di clipping sulla retta rossa



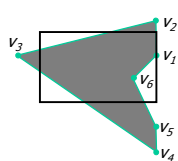
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

52

Esempio

- ❖ Vediamo come opera l'algoritmo nel caso in figura
- ❖ I vertici in nero sono gli originali, quelli in arancio sono originati dall'algoritmo di clipping sulla retta rossa



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

53

Algoritmo di Sutherland-Hodgman

- ❖ Una cosa da tenere di conto sarà l'eventualità che in output dall'algoritmo si possano ottenere dei lati che si sovrappongono ai bordi del rettangolo di clipping
- ❖ Tali lati si possono generare quando un poligono si divide in due
- ❖ È necessario allora aggiungere una fase di post-processing all'algoritmo per eliminarli

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

54

Rimozione delle superfici nascoste

Rimozione delle superfici nascoste

- ❖ Il problema della **rimozione delle superfici nascoste** consiste nel determinare se un oggetto è visibile all'osservatore, oppure rimane oscurato da altri oggetti della scena
- ❖ Non è quindi un problema legato solo alla disposizione degli oggetti nella scena, ma alla relazione che esiste tra oggetti e posizione dell'osservatore

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

56

Rimozione delle superfici nascoste

- ❖ Gli algoritmi per la rimozione delle superfici nascoste si possono dividere in due classi:
 - ❖ gli algoritmi **object-space** determinano, **per ogni oggetto**, quali parti dell'oggetto non sono oscurate da altri oggetti nella scena
 - ❖ gli algoritmi **image-space** determinano, **per ogni pixel**, quale è l'oggetto più vicino all'osservatore

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

57

Object-space

- ❖ Data una scena tridimensionale composta da n poligoni piatti ed opachi, si può derivare un generico algoritmo di tipo object-space considerando gli oggetti a coppie

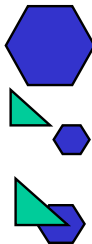
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

58

Object-space

- ❖ Data una coppia di poligoni, ad esempio A e B, ci sono quattro casi da considerare:
 - ❖ A oscura B: visualizzeremo solo A
 - ❖ B oscura A: visualizzeremo solo B
 - ❖ A e B sono completamente visibili: visualizzeremo sia A che B
 - ❖ A e B si oscurano parzialmente l'un l'altro: dobbiamo calcolare le parti visibili di ciascun poligono



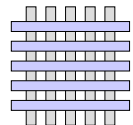
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

59

Object-space

- ❖ Si prende uno dei n poligoni e lo si confronta con tutti i restanti $n - 1$
- ❖ In questo modo si determina quale parte del poligono sarà visibile
- ❖ Questo processo è ripetuto con gli altri poligoni
- ❖ La complessità di questo approccio risulta di ordine $O(n^2)$
- ❖ L'approccio object-space è consigliabile solo quando gli oggetti nella scena sono pochi



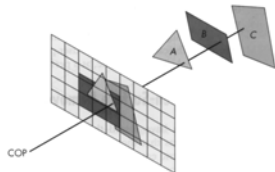
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

60

Image-space

- ❖ Per ogni pixel, si considera un raggio che parte dal centro di proiezione e passa per quel pixel
- ❖ Il raggio è intersecato con ciascuno dei piani determinati dai k poligoni per determinare per quali piani il raggio attraversa un poligono
- ❖ L'intersezione più vicina al centro di proiezione è quella visibile



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

61

Image-space

- ❖ L'operazione fondamentale dell'approccio image-space è il calcolo delle intersezioni dei raggi con i poligoni
- ❖ Per un display $w \times h$, questa operazione deve essere eseguita $w \cdot h \cdot n$ volte, e la complessità risulta di ordine $O(n)$, considerando quindi la risoluzione dello schermo una costante.

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

62

L'algoritmo *depth sort*

L'algoritmo *depth sort*

- ❖ Consideriamo una scena composta da poligoni planari
- ❖ L'algoritmo *depth sort* è una variante di un algoritmo ancora più semplice, chiamato **algoritmo del pittore**
- ❖ Supponiamo che i poligoni siano ordinati sulla base della loro distanza dall'osservatore

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

64

L'algoritmo *depth sort*

- ❖ Per rappresentare correttamente la scena, potremmo individuare la parte visibile del poligono più distante, e predisporla nel frame buffer
- ❖ Se i poligoni sono solo due, questa operazione richiede l'esecuzione del clipping di un poligono rispetto all'altro

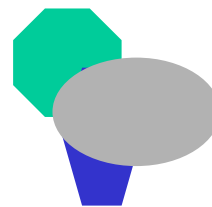
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

65

L'algoritmo *depth sort*

- ❖ Un'altra possibilità è invece quella di seguire un approccio analogo a quello usato da un pittore:
- ❖ Dipingere prima il poligono più lontano interamente, e poi dipingere il poligono più vicino, dipingendo sopra le parti del poligono più lontano non visibili all'osservatore



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

66

L'algoritmo *depth sort*

- ❖ I problemi da risolvere per implementare questo approccio riguardano
 - ❖ l'ordinamento in profondità dei poligoni
 - ❖ la situazione di sovrapposizione tra poligoni

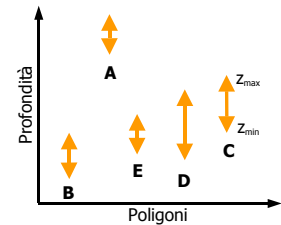
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

67

L'algoritmo *depth sort*

- ❖ Devo ordinare i poligoni in base alla distanza della loro coordinata z massima dall'osservatore
- ❖ Più precisamente, si considera l'estensione nella direzione z di ogni poligono



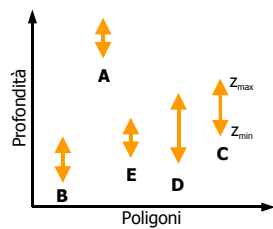
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

68

L'algoritmo *depth sort*

- ❖ Se la profondità minima di ogni poligono è maggiore della profondità massima del poligono situato sul retro, possiamo visualizzare i poligoni partendo da quello più in profondità



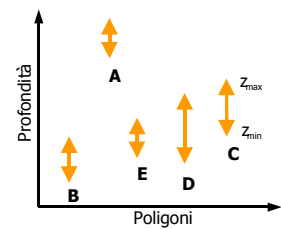
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

69

L'algoritmo *depth sort*

- ❖ E' il caso del poligono A, che è situato dietro a tutti gli altri poligoni e può essere visualizzato per primo



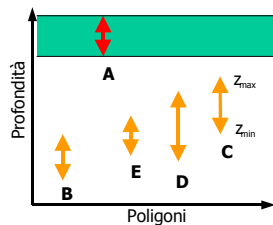
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

70

L'algoritmo *depth sort*

- ❖ E' il caso del poligono A, che è situato dietro a tutti gli altri poligoni e può essere visualizzato per primo



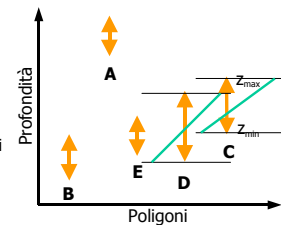
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

71

L'algoritmo *depth sort*

- ❖ Gli altri poligoni, tuttavia, non possono essere visualizzati basandosi solo sulla loro estensione lungo z
- ❖ Se le estensioni z di due poligoni si sovrappongono, dobbiamo determinare un ordine per visualizzarli individualmente che permetta di ottenere l'immagine corretta



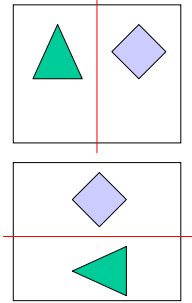
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

72

L'algoritmo *depth sort*

- ❖ Il test più semplice consiste nel controllare le estensioni lungo x e lungo y
- ❖ Se non c'è sovrapposizione in almeno una delle due direzioni, allora sicuramente nessuno dei due poligoni può oscurare l'altro, ed essi possono essere visualizzati in un ordine qualsiasi



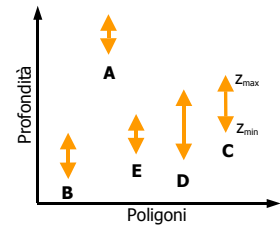
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

73

L'algoritmo *depth sort*

- ❖ Se anche questo test fallisce, può essere ancora possibile trovare un ordine corretto per visualizzare i poligoni, ad esempio se tutti i vertici di un poligono cadono dalla stessa parte del piano determinato dall'altro poligono



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

74

L'algoritmo *depth sort*

- ❖ Rimangono da considerare due situazioni problematiche, per cui non esiste un ordine corretto di rappresentazione

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

75

L'algoritmo *depth sort*

- ❖ La prima si verifica quando tre o più poligoni si sovrappongono ciclicamente



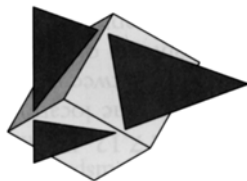
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

76

L'algoritmo *depth sort*

- ❖ La seconda situazione si verifica invece quando un poligono penetra nell'altro



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

77

L'algoritmo *depth sort*

- ❖ In entrambi i casi, è necessario spezzare i poligoni in corrispondenza dei segmenti di intersezione, e cercare l'ordine corretto di rappresentazione del nuovo insieme di poligoni

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

78

L'algoritmo z-buffer

L'algoritmo z-buffer

- ❖ L'algoritmo **z-buffer** è un algoritmo di tipo image-space, basato su una logica molto semplice, e facile da implementare
- ❖ Lavora in accoppiamento con l'algoritmo di scan conversion, e necessita, oltre alla memoria di display (frame buffer), anche di un'area di memoria in cui memorizzare le **informazioni di profondità** relative ad ogni pixel
- ❖ Quest'area addizionale di memoria è chiamata **z-buffer**

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

80

L'algoritmo z-buffer

- ❖ La rasterizzazione dei poligoni avviene subito dopo la proiezione sul piano di vista in NDC
- ❖ Ad ogni pixel dello schermo possono coincidere 0, 1 o più primitive
- ❖ Nel corso dell'esecuzione della scan conversion, possiamo pensare al processo di proiezione come al calcolo del colore da associare ad ogni punto di intersezione tra una retta che passa dal centro di proiezione ed un pixel e le primitive



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

81

L'algoritmo z-buffer

- ❖ Nell'effettuare questa operazione, si può facilmente controllare se il punto di intersezione è visibile o meno (dall'osservatore), secondo la regola che stabilisce che *il punto è visibile se è il punto di intersezione più vicino al centro di proiezione*

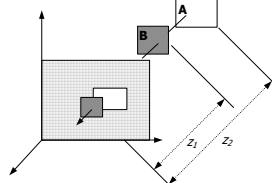
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

82

L'algoritmo z-buffer

- ❖ Quando si esegue la scan conversion del poligono B, il suo colore apparirà sullo schermo poiché la distanza z_1 è minore della distanza z_2 relativa al poligono A
- ❖ Al contrario, quando esegue la scan conversion del poligono A, il pixel che corrisponde al punto di intersezione non apparirà sul display



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

83

L'algoritmo z-buffer

- ❖ Poiché si procede poligono per poligono, non è possibile disporre di tutte le informazioni relative agli altri poligoni
- ❖ Dobbiamo disporre di una memoria, detta appunto z-buffer, che abbia la stessa risoluzione del frame buffer e con una profondità sufficiente per memorizzare le informazioni sulla risoluzione che si vuole ottenere per le distanze
- ❖ Ogni elemento dello z-buffer è inizializzato al valore della distanza massima dal centro di proiezione (il back-clipping plane)

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

84

L'algoritmo z-buffer

- ❖ Con questo approccio i poligoni possono essere rasterizzati in qualsiasi ordine (non è necessario alcun ordinamento preventivo dei poligoni in object-space, ovvero in 3D)

```

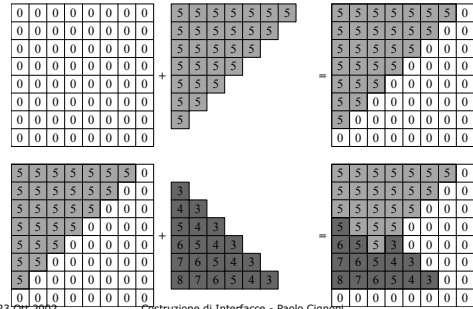
for y:=0 YMAX
  for x:=0 XMAX
    writePixel(x,y, colore del background);
    writeZ(x,y,0);
  for ogni poligono
    for ogni pixel nella proiezione del poligono
      pz := valore della z nel pixel di coordinate (x,y)
      if pz >= ReadZ(x,y) then
        writeZ(x,y,pz)
        writePixel(x,y, colore del poligono nel pixel di
          coordinate (x,y))
  
```

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

85

L'algoritmo z-buffer



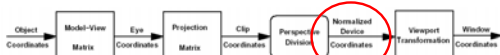
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

86

L'algoritmo z-buffer

- ❖ Se ripensiamo a come avviene la proiezione da 3 a 2 dimensioni, una volta trasformato il view frustum nel cubo in NDC possiamo pensare all'algoritmo di z-buffer come a quel metodo che anziché scartare la z al momento della proiezione la memorizza (nello z-buffer appunto) assieme all'informazione sul colore (nel frame buffer)



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

87

Eliminazione delle back face

- ❖ Oltre alla rimozione delle superfici nascoste esistono metodi opzionali che consentono di eliminare dalla pipeline di rendering primitive che si può decidere che saranno comunque invisibili
- ❖ Se un oggetto è rappresentato da un poliedro solido chiuso, le facce poligonali del poliedro delimitano completamente il volume del solido
- ❖ Modelliamo i poligoni in maniera tale che le normali alle loro superfici siano tutte dirette verso l'esterno del poligono

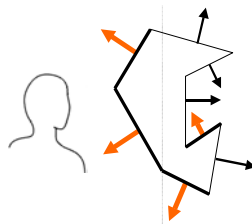
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

88

Eliminazione delle back face

- ❖ Se nessuna parte del poliedro viene tagliata dal front clipping plane:
 - ❖ le facce che hanno una normale che punta verso l'osservatore possono essere visibili



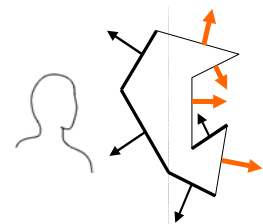
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

89

Eliminazione delle back face

- ❖ Se nessuna parte del poliedro viene tagliata dal front clipping plane:
 - ❖ quelle con normale che punta via dall'osservatore **sicuramente non lo sono**



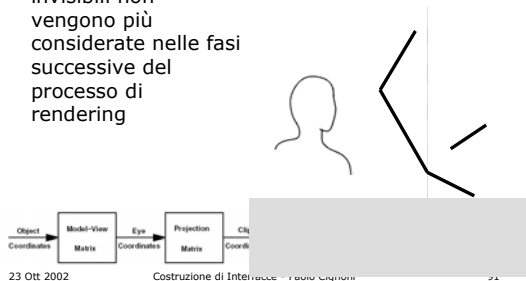
23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

90

Eliminazione delle back face

- ❖ Le facce sicuramente invisibili non vengono più considerate nelle fasi successive del processo di rendering



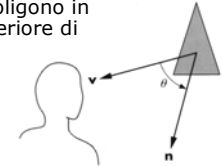
Eliminazione delle *back face*

- ❖ Per determinare se un poligono deve essere eliminato, dobbiamo verificare se la sua normale è diretta o meno verso l'osservatore
- ❖ Se indichiamo con θ l'angolo tra la normale e l'osservatore, il poligono in esame definisce la parte anteriore di un oggetto se e solo se

$$-90^\circ \leq \theta \leq 90^\circ, \\ \cos \theta \geq 0$$

- ❖ Invece di calcolare il coseno, si usa il prodotto scalare

$$n \cdot v \geq 0$$



23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

92

Eliminazione delle *back face*

- ❖ Questo procedimento (detto di solito *back-face culling*) consente, in media, di dimezzare il tempo necessario a fare il rendering degli oggetti solidi dato che, sempre in media, circa metà delle facce di un poliedro saranno *back-facing* e quindi il loro rendering sarebbe comunque inutile

23 Ott 2002

Costruzione di Interfacce - Paolo Cignoni

93