Dentro o fuori il triangolo?

Livello scolare: 1° biennio.

Abilità interessate	Conoscenze	Nuclei coinvolti	Collegamenti esterni
Utilizzare in modo consapevole gli strumenti di calcolo automatico.	I polinomi e le loro operazioni.	Numeri e algoritmi Spazio e figure Relazioni e funzioni Laboratorio di matematica	

Contesto
Geometria analitica.

L'uso del computer è sempre più facilitato da interfacce grafiche che aiutano l'utente nello svolgimento delle attività. Tali interfacce, se da una parte facilitano l'utente, dall'altra rischiano di renderlo inconsapevole delle problematiche, non solo tecniche, che stanno alla base delle attività. Si può discutere se questo sia un bene o un male; certamente può essere un elemento utile - anche per attirare l'attenzione e l'interesse degli allievi - a poter rivisitare argomenti della didattica tradizionale della matematica alla luce delle implicazioni che possono avere con le nuove tecnologie. Un esempio può essere lo studio del piano cartesiano con riferimento alla grafica del computer. Nell'attività presentata si propone lo studio delle disequazioni sul piano cartesiano come strumento per distinguere sullo schermo del computer, con un opportuno algoritmo, i punti interni da quelli esterni di un triangolo.

Descrizione dell'attività
Lo scopo dell'attività è la definizione di un algoritmo, e la relativa codifica in un linguaggio di programmazione, che riconosca quando un punto è interno, esterno o sul bordo di un triangolo. Il punto ed i vertici del triangolo sono dati attraverso le loro coordinate cartesiane. L'unità inizia mostrando come si possa riconoscere il semipiano d'appartenenza del punto rispetto alla retta determinata passante per due punti dati. Si discute poi come stabilire con una formula, che è una forma d'algoritmo, quando due punti appartengono allo stesso semipiano. La formula è la base dell'algoritmo che caratterizza i punti del piano rispetto ad un dato triangolo. L'unità si conclude con un gioco grafico: dato un triangolo si simula una passeggiata casuale all'interno del triangolo controllando che il punto in movimento, all'inizio coincidente con il baricentro, rimanga dentro.

Dentro o fuori il triangolo: che domanda è?
Preso un foglio di carta, disegnare un triangolo, segnare un punto e chiedere se è dentro o fuori il triangolo: questa sembra proprio la classica domanda che toglie agli alunni ogni dubbio sulla salute mentale del proprio insegnante. Eppure la domanda può avere un qualche interesse, anche per gli alunni più smaliziati. Il triangolo è dato sul piano cartesiano come terna di punti (i suoi tre vertici); ci si chiede come si colloca un punto, note le sue coordinate, rispetto al triangolo.
Si può anche dare al problema una veste più accattivante: "In un parco a forma di triangolo (Yellowstone) è stato introdotto un nuovo animale (Jogi), dotato di collare con radiocomando, che deve essere tenuto sotto controllo: quando Jogi è all'interno del parco (dentro il triangolo) va tutto bene, quando invece si avvicina al bordo in centrale si accende un lampeggiante giallo di avvertimento. Infine se Jogi esce dal parco scatta l'allarme".

Per far funzionare il sistema di controllo occorre un algoritmo che, a partire dalle coordinate dei tre vertici di un triangolo, calcoli quando un punto è interno, esterno o sul bordo dello stesso.

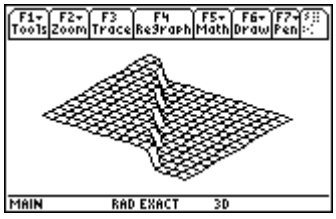
Segno di un punto rispetto ad una retta.
Una retta divide il piano in tre insiemi di punti: due semipiani (senza bordo) e la retta stessa. Dal punto di vista algebrico, i punti del piano cartesiano sono divisi nei tre insiemi per i quali il polinomio $P(x, y) = a\,x + b\,y + c$ produce rispettivamente un valore positivo, negativo o nullo.
Dati due punti $A(x_0, y_0)$ e $B(x_1, y_1)$, l'equazione canonica (o implicita) della retta passante per i due punti è: $x\,(y_0 - y_1) - y\,(x_0 - x_1) + x_0\,y_1 - x_1\,y_0 = 0$. Il polinomio, primo membro dell'equazione, si annulla sulla retta mentre darà valori positivi su un semipiano e negativi sull'altro.

Osservazione: nell'indicare i punti sul piano cartesiano si userà l'abituale notazione delle lettere latine maiuscole, come si è fatto nel precedente capoverso. Nei programmi invece le variabili, comprese quelle che indicano coppie di coordinate e quindi punti nel piano cartesiano, sono indicate per consuetudine con lettere minuscole. Nel seguito per i punti sarà usata indifferentemente l'una o l'altra notazione quando, in base al contesto, non ci sia ambiguità.

```
segno(a,b,p)
Func
Local  c,t
10^(-6)→t
p[1]*(a[2]-b[2])-p[2]*(a[1]-b[1])+
a[1]*b[2]-a[2]*b[1]→c
If abs(c)<t Then
   Return  0
Else
   If c>0 Then
      Return  1
   Else
      Return  -1
   EndIf
EndIf
EndFunc
```

Possiamo definire una funzione usando l'equazione sopra descritta. Nella Figura 1 la funzione è descritta nel linguaggio di una calcolatrice grafica programmabile.
La funzione *segno*(*a,b,p*) restituisce il valore 0 se il punto *p* è allineato con i punti *a* e *b*, mentre darà 1 se *p* appartiene ad un semipiano o −1 se appartiene a quello opposto. Si può anche osservare che il segno determina l'ordine della sequenza dei tre punti: se i punti si succedono (nel perimetro del triangolo di cui sono vertici) in verso antiorario allora la funzione *segno* produce 1, altrimenti (verso orario) −1. Se il triangolo è degenere il risultato è 0.

Figura 1



La Figura 2 mostra il grafico in 3 dimensioni della funzione $z1(x, y) = segno(a,b,\{x, y\})$, con *a* e *b* due punti fissati nel piano cartesiano. Nella figura si osserva che il piano è diviso in due semipiani: il primo a quota 1 e l'altro a −1. Le coordinate del punto generico (x, y) sono introdotte nella funzione *segno* come lista e, quindi nella calcolatrice, sono racchiuse in parentesi graffe.

Figura 2

Data la retta *AB,* due punti *P* e *Q* (non allineati con *A*, *B*) appartengono allo stesso semipiano se la funzione segno produce lo stesso valore. Per stabilire se i due punti sono nello stesso semipiano basta studiare il prodotto delle funzioni $segno(a,b,p) \times segno(a,b,q)$.

Dentro o fuori del triangolo?
La funzione *test*(*a,b,c,p*) controlla se un punto *p* è esterno, sul bordo o interno al triangolo di vertici *a*, *b* e *c*. Si descrive dapprima l'algoritmo della funzione con un diagramma di flusso.

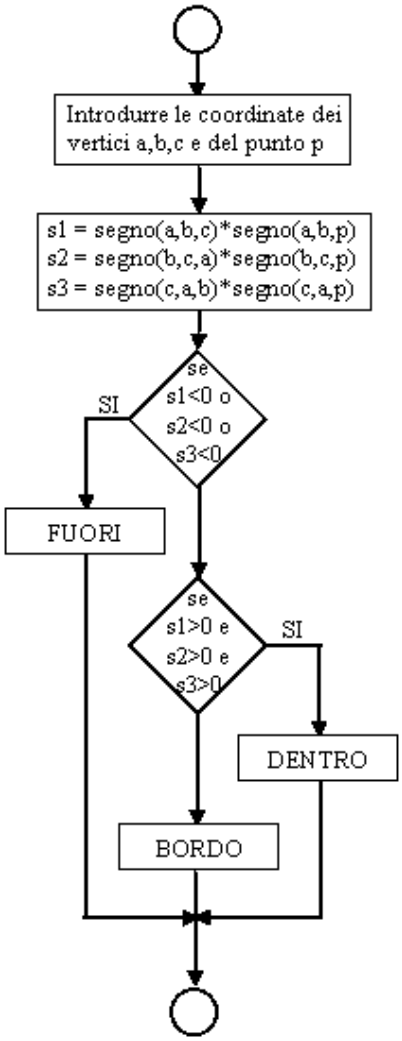


Figura 3

La funzione *test* riceve come argomenti le coordinate dei vertici *a*, *b*, *c* del triangolo e quelle del punto *p* da controllare.

Con la funzione *segno*, definita in precedenza, si verifica se il punto *p* appartiene in ogni caso allo stesso semipiano che contiene il terzo vertice rispetto alla retta che passa per gli altri due: il prodotto è positivo quando *p* e il terzo vertice appartengono allo stesso semipiano. Le variabili *s*1, *s*2, *s*3 contengono la posizione di *p* rispetto ai tre lati.

Se il valore delle variabili di controllo è in almeno un caso negativo, il punto *p* è esterno al triangolo.

Se i valori di controllo sono tutti positivi, il punto è interno al triangolo.

Infine, quando nessun valore è negativo, ma in qualche caso è nullo, il punto si trova sul bordo del triangolo.

Figura 3

Si passa ora alla stesura dell'algoritmo, sotto forma di funzione, nel linguaggio di una calcolatrice grafica programmabile.

```
test(a,b,c,p)
Func
Local  c,s1,s2,s3
If segno(a,b,c)=0 Then
  Return  "Triangolo degenere"
Else
  segno(a,b,c)*segno(a,b,p)→s1
  segno(b,c,a)*segno(b,c,p)→s2
  segno(c,a,b)*segno(c,a,p)→s3
  If s1<0 or s2<0 or s3<0 Then
    Return  "FUORI"
  Else
    If s1=0 or s2=0 or s3=0 Then
      Return  "BORDO"
    Else
      Return  "DENTRO"
    EndIf
  EndIf
EndIf
EndFunc
```

La funzione inizia controllando se i tre punti sono allineati: in tal caso dichiara *degenere* il triangolo.

Per un triangolo non degenere, il programma corrisponde esattamente al diagramma di flusso della Figura 3: i prodotti delle funzioni *segno*, registrati nelle variabili di controllo *s*1, *s*2 e *s*3, sono poi usati per stabilire la posizione del punto *p* rispetto al triangolo di vertici *a*, *b* e *c*. Il risultato è riportato sotto forma di *stringa*.

Figura 4

Si può dare il risultato della funzione *test* in forma grafica. Il programma *tri*(*a*,*b*,*c*,*p*) disegna il triangolo di vertici *a*, *b* e *c* e disegna il punto *p*. Poi riporta nella parte inferiore dello schermo la posizione del punto *p* rispetto al triangolo.

```
tri(a,b,c,p)
Prgm
setGraph("Axes","Off")
ClrGraph:ClrDraw
76→ymax:0→ymin
158→xmax:0→xmin
1→yscl:1→xscl

©Triangolo
PxlLine a[2],a[1],b[2],b[1],1
PxlLine b[2],b[1],c[2],c[1],1
PxlLine c[2],c[1],a[2],a[1],1
©Punto
PxlLine p[2]-1,p[1],p[2]+1,p[1],-1
PxlLine p[2],p[1]-1,p[2],p[1]+1,-1

PxlText test(a,b,c,p),ymax-20,xmax-50

Pause
setGraph("Axes","On")
DispHome
EndPrgm
```

La prima parte del programma *tri* predispone la finestra grafica ed imposta le dimensioni dello schermo.

Lo schermo è visto come 'matrice' di punti individuati da coppie di numeri interi (riga,colonna), con coordinate 'scambiate' rispetto all'ordinaria disposizione nel piano cartesiano di ascissa e ordinata.

Gli argomenti del programma sono dati come coordinate nell'ordine usuale. L'istruzione grafica nella Figura 5 (*PxlLine r0,c0,r1,c1,–1*) disegna il segmento delimitato dai punti (riga0,colonna0) e (riga1, colonna1) invertendo il 'colore' dei punti dello schermo.
Il punto da testare rispetto al triangolo è disegnato, per renderlo più evidente, come crocetta centrata nelle coordinate del punto.
La risposta del test è scritta in basso sulla destra dello schermo.

Figura 5

Ecco due esempi di esecuzione del programma:

tri({5,5},{140,30},{85,70},{100,50})

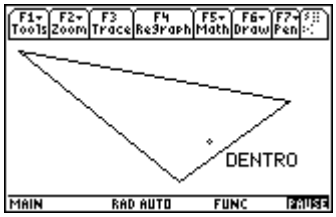


Figura 6

tri({5,5},{140,30},{85,70},{40,50})

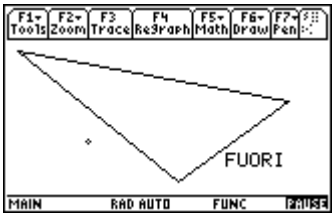


Figura 7

Jogi nel parco di Yellowstone.

La funzione *test2*, è la funzione *test* modificata in modo che restituisca un numero in base alla posizione del punto rispetto al triangolo: 1 se il punto è interno, 0 se è sul bordo oppure –1 se è esterno. Nel programma che segue la funzione *test2* è usata per un gioco grafico che offre l'occasione di qualche riflessione sulla programmazione.
Si immagina che il triangolo sia il recinto del parco di Yellowstone nel quale è rinchiuso l'orso Jogi. Jogi passeggia nel recinto, ma non può uscire. Il programma *jogi* simula la passeggiata dell'orso, che si muove a caso all'interno del recinto.

```
jogi(a,b,c)
Prgm
Local  x,y,p1,p2,k,l
setGraph("Axes","Off")
ClrGraph:ClrDraw
76→ymax:0→ymin
158→xmax:0→xmin
1→yscl:1→xscl
©Triangolo (recinto)
PxlLine  a[2],a[1],b[2],b[1],1
PxlLine  b[2],b[1],c[2],c[1],1
PxlLine  c[2],c[1],a[2],a[1],1
©Punto (jogi all'inizio)
int((a[1]+b[1]+c[1])/3+0.5)→x
int((a[2]+b[2]+c[2])/3+0.5)→y
PxlOn  y,x
PxlLine  y-2,x,y+2,x,-1
PxlLine  y,x-2,y,x+2,-1
1→p1:1→p2
0→k
While  k=0
  rand(10)-1→l
  While  l>0
    If test2(a,b,c,{x+p1,y+p2})=1 Then
       PxlOn  y,x
       PxlLine  y-2,x,y+2,x,-1
       PxlLine  y,x-2,y,x+2,-1
       x+p1→x:y+p2→y
       PxlLine  y-2,x,y+2,x,-1
       PxlLine  y,x-2,y,x+2,-1
    EndIf
    l-1→l
  EndWhile
  If rand(2)-1=0 Then
     -1*p1→p1
    Else
     -1*p2→p2
  EndIf
  getKey()→k
EndWhile
setGraph("Axes","On")
DispHome
EndPrgm
```

Il programma *jogi* ha come argomenti i tre vertici del triangolo.

Dopo aver disegnato, come nel programma *tri* descritto prima, i lati del triangolo, si pone nel baricentro del triangolo (calcolato come media dei vertici) la posizione iniziale dell'orso.

La direzione di marcia dell'orso è data dall'incremento delle coordinate, che può essere –1 o 1 e che è registrato nelle due variabili *p1* e *p2*. All'inizio la direzione è fissata, successivamente varierà in modo casuale. La variabile *k* registra il codice del tasto premuto e rimane 0 finché non si tocca la tastiera: serve ad interrompere l'esecuzione del programma premendo un tasto qualsiasi della calcolatrice.
La variabile *l* indica il numero di passi dell'orso, data la direzione: il numero di passi è un numero a caso tra 0 e 9.
Ad ogni passo, si controlla (funzione *test2*) se Jogi resta dentro al recinto: in questo caso si segna il punto in cui si trova, si cancella la 'crocetta' della posizione attuale, si fa il passo e si disegna la crocetta nella nuova posizione.
Finiti i passi nella direzione attuale, si calcola la nuova direzione: a questo scopo si inverte, dopo averlo scelto a caso, l'incremento in ascisse o in ordinate. In questo modo il cammino casuale dell'orso sarà sempre in direzione 'diagonale' rispetto allo schermo.

Figura 8

Le figure seguenti mostrano qualche percorso casuale, ma rigorosamente dentro al recinto, di Jogi.

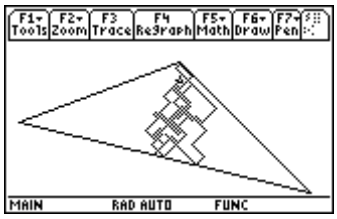


Figura 9

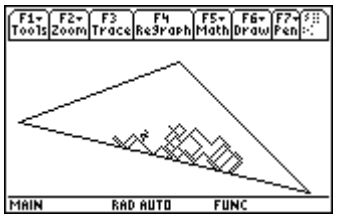


Figura 10

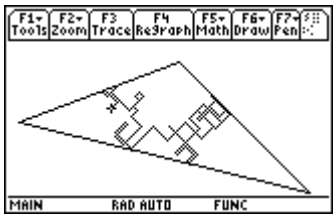


Figura 11

Possibili sviluppi

- Come si è detto, e come si vede nelle figure, il programma *jogi* fa muovere il punto in diagonale sullo schermo. Si può proporre di estendere il programma in modo che il punto si muova anche nelle direzioni nord-sud e est-ovest.
- Un secondo problema 'banale' se pensato *carta e penna*, ma più interessante se realizzato *al computer*, è il seguente: stabilire se due segmenti si intersecano o meno. I due segmenti sono

dati come coppie di punti (gli estremi) sul piano cartesiano. Una prima soluzione consiste nel calcolare l'intersezione delle rette sostegno e nel verificare se questa appartiene ad entrambi i segmenti. È questa la soluzione migliore, cioè quella che richiede meno calcoli? Una soluzione alternativa può essere trovata esprimendo i segmenti in forma parametrica: si ottiene un risultato migliore come complessità di calcolo?

- Un algoritmo è, in casi fortunati, esprimibile con una formula: un esercizio bello di geometria analitica può essere il calcolo della formula della simmetria assiale, rispetto ad una retta espressa nella forma $y = m\ x + q$. Indicati con $P(x_0, y_0)$ e $P'(x_1, y_1)$ i punti corrispondenti nella simmetria, le condizioni che danno le equazioni da porre a sistema sono che il punto medio di PP' appartenga alla retta e che la retta PP' sia perpendicolare a quella data. Avendo a disposizione un sistema di calcolo simbolico, la determinazione delle equazioni della simmetria assiale è a questo punto immediata. Con le formule ottenute, si possono poi dimostrare, per via algebrica, le ben note proprietà della composizione della simmetrie assiali.

Elementi di prove di verifica

Funzioni o programmi?

1. Qual è la differenza tra un algoritmo descritto da una funzione e quello descritto da un programma?

Se … allora … quando.

2. A cosa servono le strutture di controllo negli algoritmi? Quali sono quelle fondamentali?

Formule e algoritmi.

3. In quali condizioni un algoritmo può essere espresso da una formula?