Java 2D Rendering

Luca Grilli

Proprietà dell'oggetto Graphics

- Abbiamo visto che un oggetto Graphics permette di effettuare il rendering di una qualsiasi immagine descritta mediante una serie di primitive grafiche.
- Le primitive grafiche permettono di:
  - Riprodurre una qualsiasi forma geometrica a partire da una serie di primitive geometriche: linee, curve, poligonali, rettangoli, ellissi, poligoni, etc..
  - Impostare dei parametri legati all'apparenza, come il colore del pennello, e riprodurre immagini digitali.
  - Visualizzare delle stringhe testuali in modo da poter impostare il font e la dimensione dei caratteri.
- Nelle prossime slide analizzeremo i metodi (astratti) definiti nella classe astratta Graphics e implementati da tutti gli oggetti che manipolano dei contesti grafici.

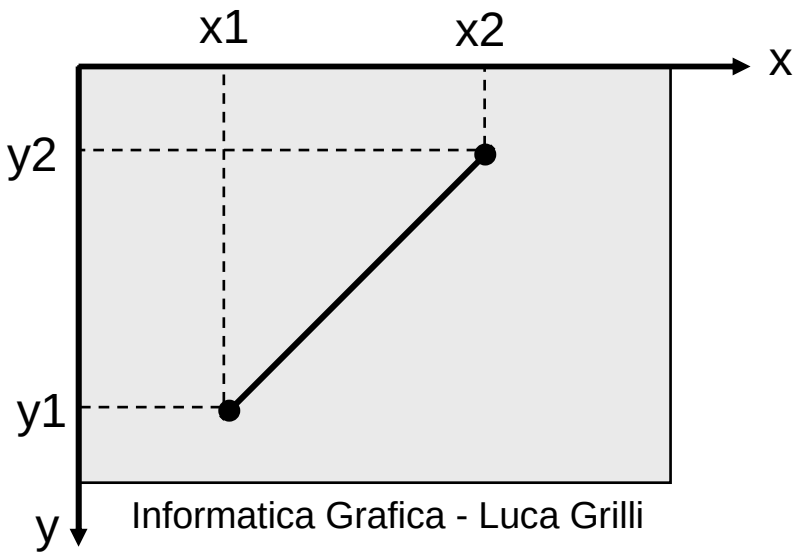
Primitive geometriche di Graphics

Primitive Geometriche di Graphics

- Le primitive geometriche di Graphics permettono di visualizzare tutte le figure geometriche ottenibili utilizzando le seguenti figure geometriche elementari.
  - Figure 1D o "forme aperte":
    - Linee ⇒ **drawLine( )**
    - Archi di ellissi/circonferenze ⇒ **drawArc( )**
    - Spezzate ⇒ **drawPolyLine( )**
  - Contorno (iniziano con "**draw**…") di Figure 2D o "forme chiuse" 2D:
    - Rettangoli ⇒ **drawRect( )**
    - Rettangoli con angoli arrotondati ⇒ **drawRoundRect( )**
    - Ovali (ellissi/circonferenze) ⇒ **drawOval( )**
    - Poligoni ⇒ **drawPolygon( )**
  - Riempimento (iniziano con "**fill**…") di Figure 2D o "forme chiuse" 2D:
    - Rettangoli ⇒ **fillRect( )**
    - Rettangoli con angoli arrotondati ⇒ **fillRoundRect( )**
    - Ovali (ellissi/circonferenze) ⇒ **fillOval( )**
    - Poligoni ⇒ **fillPolygon( )**

Disegno di linee: drawLine( )

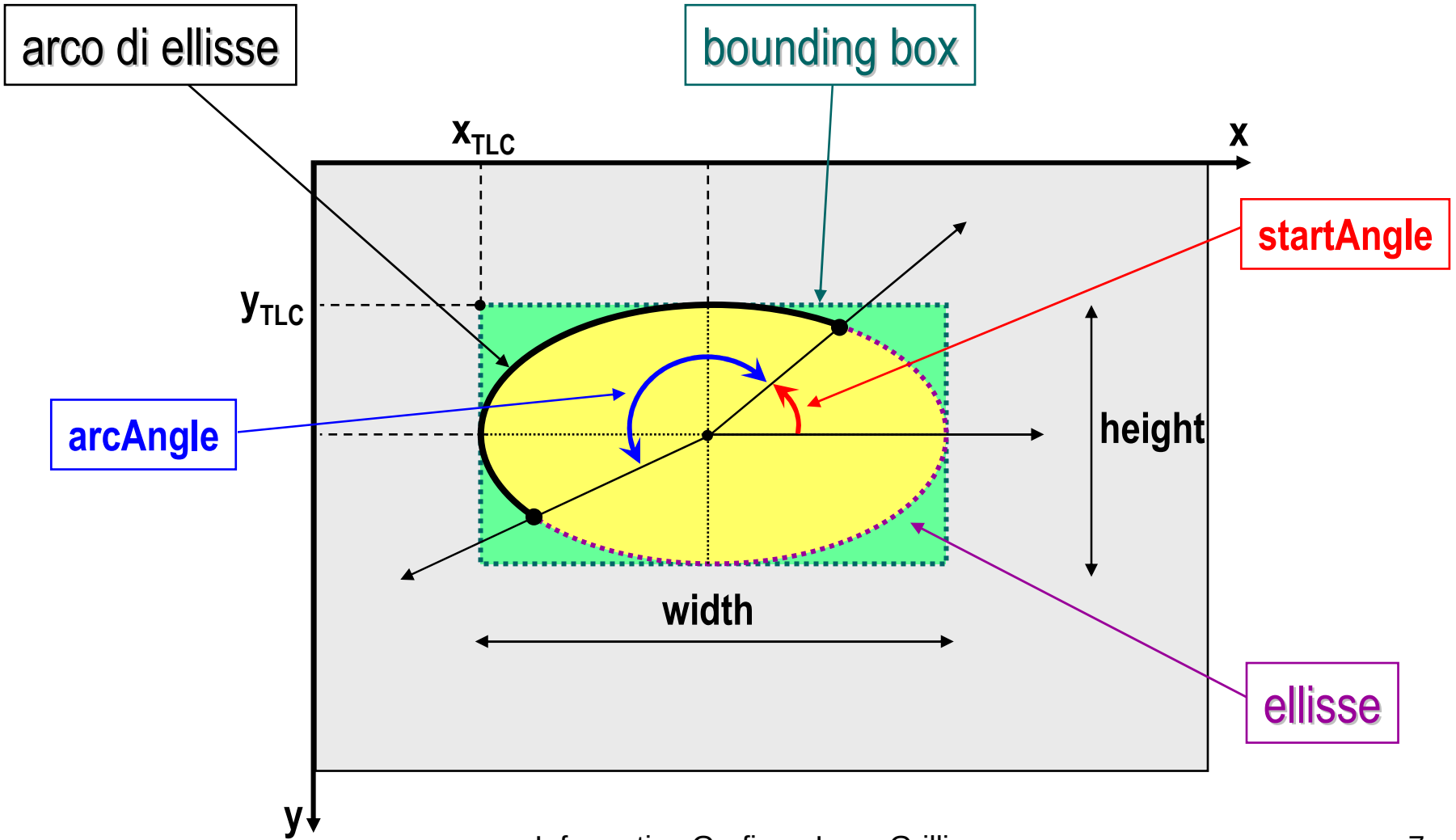
```
/* Disegna, utilizzando il colore corrente, una linea
(segmento) che unisce i punti di coordinate (x1, y1) e
(x2, y2) nel sistema di coordinate associate al contesto
grafico. */
public void drawLine(int x1, int y1, int x2, int y2)
// x1: coordinata x del primo punto
// y1: coordinata y del primo punto
// x2: coordinata x del secondo punto
// y2: coordinata y del secondo punto
```



Disegno di archi: drawArc( )

- Disegna il contorno di un arco di ellisse (circonferenza).
- L'ellisse (circonferenza) viene definita specificando la regione rettangolare (o bounding box), top-left corner, width e height, in cui è inscritta.
- Il centro dell'ellisse coincide con il centro della regione rettangolare in cui è inscritta.
- L'arco di ellisse (circonferenza) viene specificato definendo la sua estensione angolare (arcAngle) e l'offset angolare (startAngle), cioè l'angolo di rotazione in senso anti-orario rispetto la semiretta orizzontale, orientata da sinistra verso destra e che ha origine nel centro dell'ellisse.

Disegno di archi: drawArc( )

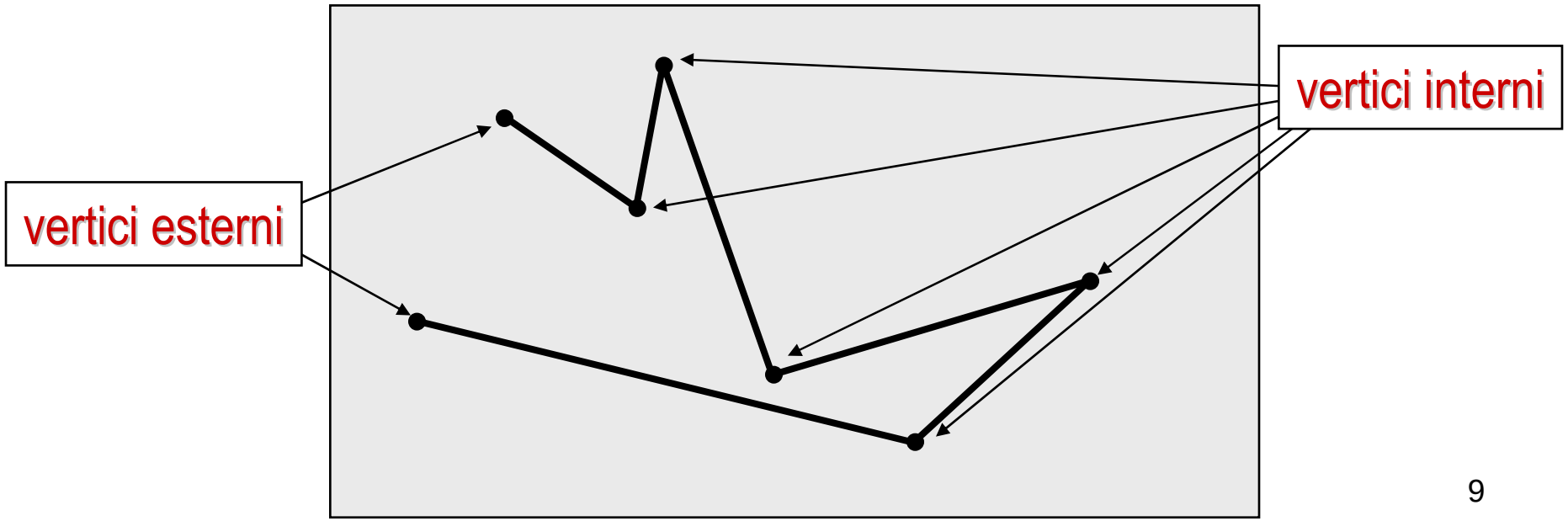


Disegno di archi: drawArc( )

```
/* Disegna, utilizzando il colore corrente, il contorno di un
arco di ellisse (circonferenza).
L'ellisse viene definita specificando la regione rettangolare
(bounding box: x, y, width, height) in cui è inscritta.
L'arco viene definito specificando la sua estensione angolare
(arcAngle) e l'offset angolare (startAngle), espressi in gradi.
Il riferimento angolare è la semiretta orizzontale, orientata da
sinistra verso destra e che ha origine nel centro dell'ellisse
(bounding box).
Gli angoli positivi (negativi) corrispondono a delle rotazioni
angolari in senso anti-orario (orario) rispetto tale semiretta.
*/
public void drawArc(int x,
                    int y,
                    int width,
                    int height,
                    int startAngle,
                    int arcAngle)
```

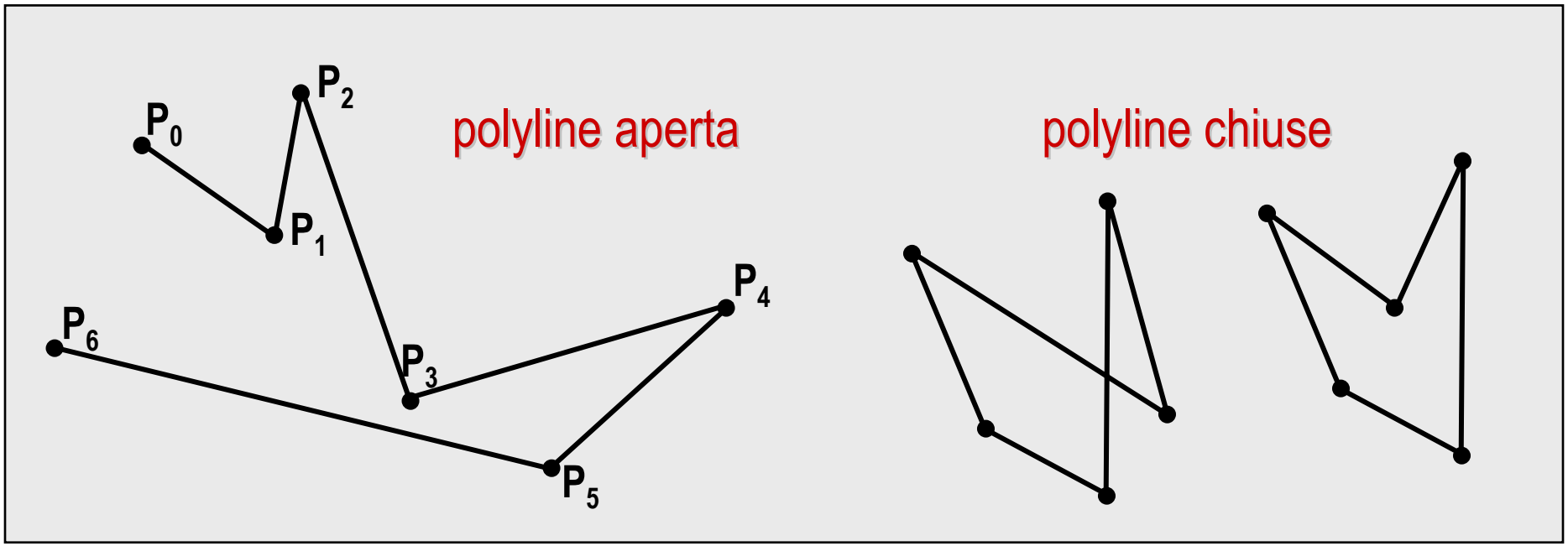
Disegno di spezzate (o polyline)

- Una spezzata o polyline, è una figura geometrica continua ottenuta connettendo un insieme di linee (segmenti).
- I punti estremi di ogni linea della polyline sono detti vertici (o semplicemente punti).
- I vertici della polyline o sono collegati ad una sola linea (vertici esterni) o sono collegati a due linee consecutive (vertici interni).



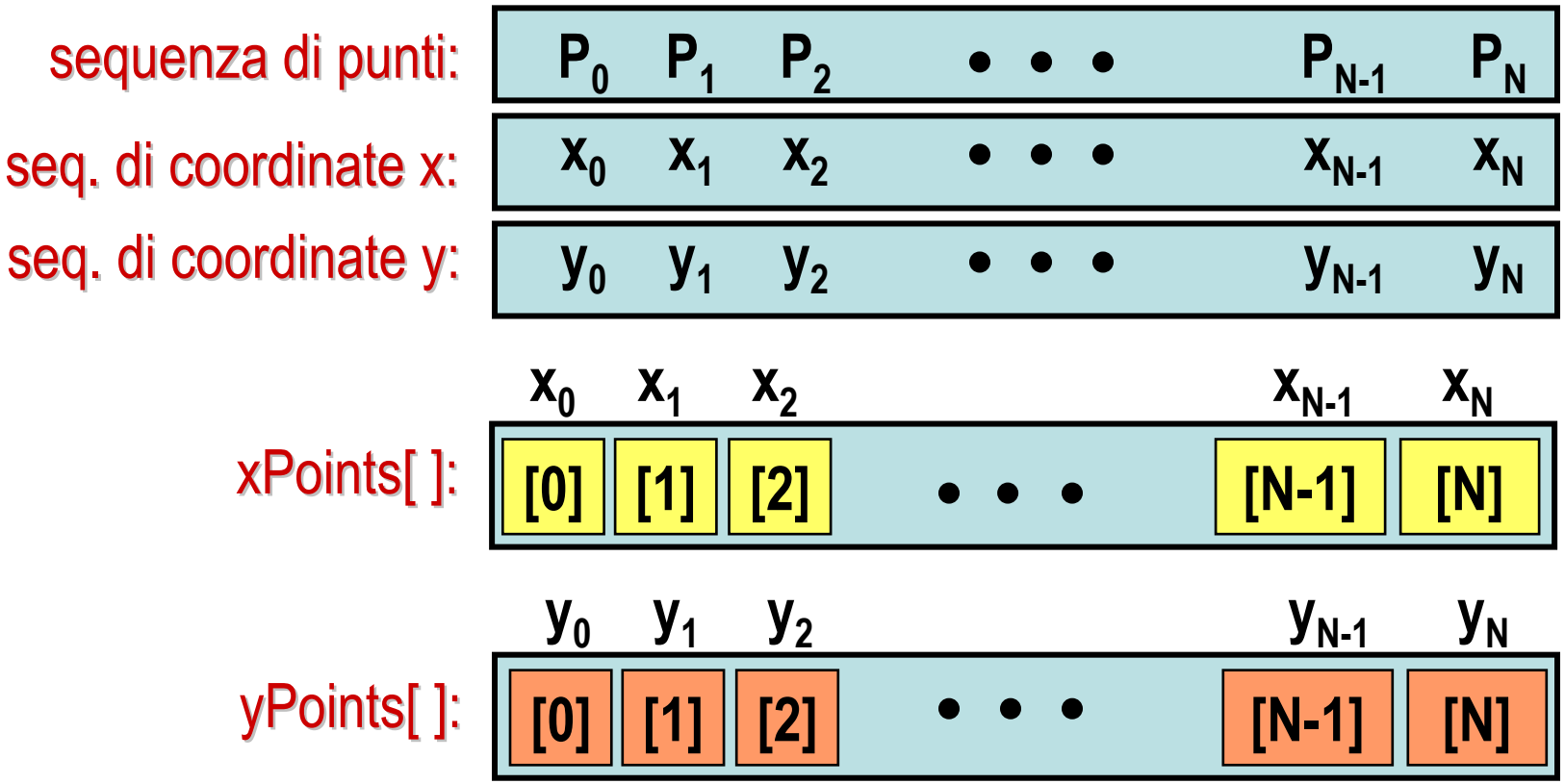
Disegno di spezzate (o polyline)

- Una polyline di N linee è descrivibile elencando la sequenza dei suoi N+1 vertici distinti, rispettando l'ordine con il quale si incontrano percorrendo la polilyne da uno dei suoi vertici esterni all'altro:
  $P_0 \rightarrow P_1 \rightarrow \ldots \rightarrow P_{N-1} \rightarrow P_N$ o $P_N \rightarrow P_{N-1} \rightarrow \ldots \rightarrow P_1 \rightarrow P_0$.
- Una polilyne è una figura geometrica aperta se i suoi vertici esterni non sono coincidenti, altrimenti è chiusa.

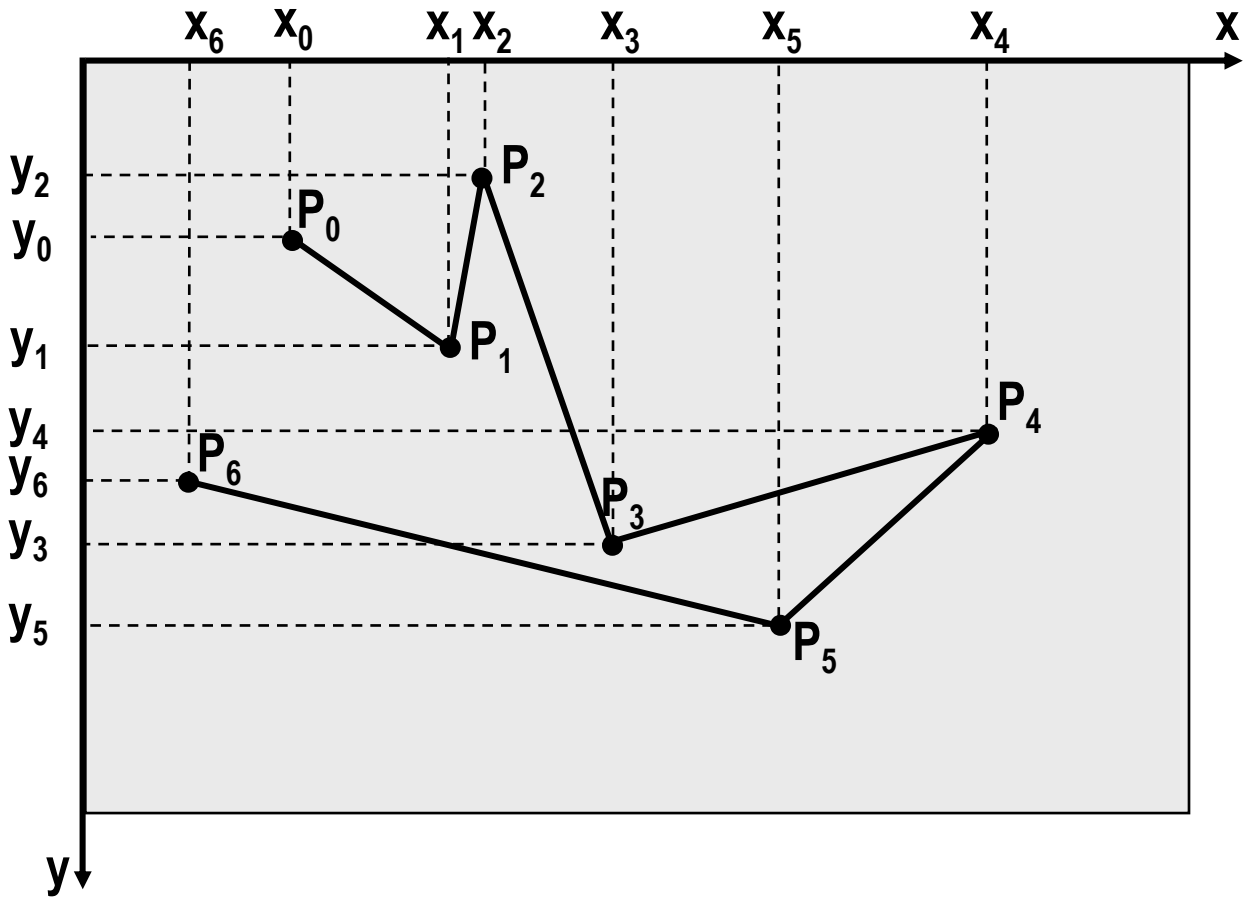


Disegno di spezzate (o polyline)

- Una polyline di N linee è pertanto descrivibile specificando due array di N+1 valori numerici contenenti rispettivamente le sequenze delle coordinate x e y della sequenza di punti $\{P_i \equiv (x_i, y_i)\}_{i=0,..,N}$:
  $P_0 \rightarrow P_1 \rightarrow \ldots \rightarrow P_{N-1} \rightarrow P_N$.



Disegno di spezzate (o polyline)



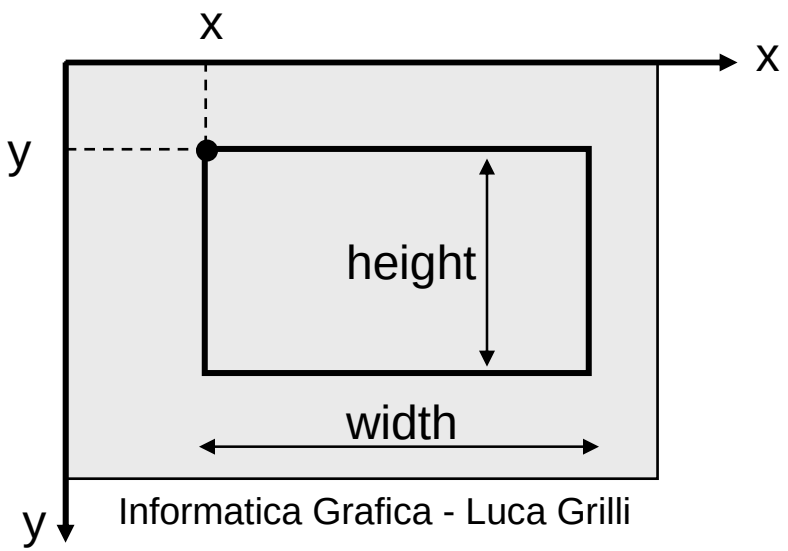
Disegno di polyline: drawPolyLine( )

```
/*
Disegna, con il colore corrente, una sequenza connessa di linee
(polyline o spezzata) specificate dagli array contenenti le
sequenze delle coordinate x e y.
Ogni coppia di coordinate (x, y) definisce un punto (vertice)
della polyline. La figura non è chiusa se il primo punto
differisce dall'ultimo.
*/
public void drawPolyline(int[] xPoints,
                         int[] yPoints,
                         int nPoints)


/*
Parametri:
xPoints – array delle coordinate x dei punti della polyline
yPoints – array delle coordinate y dei punti della polyline
nPoints – numero totale di punti della polyline
*/
```

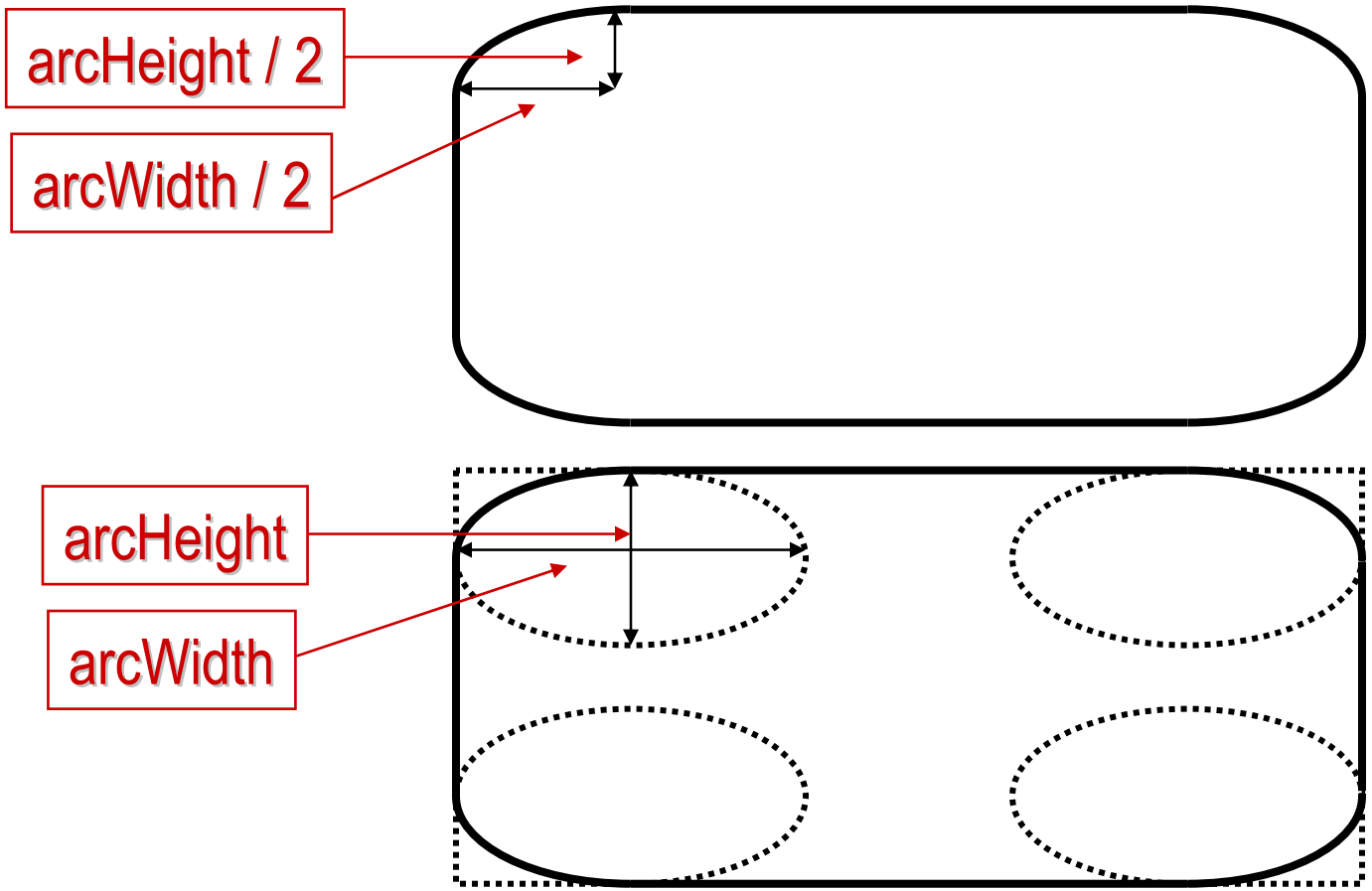
Disegno di rettangoli (contorno): drawRect( )

```
/* Disegna, utilizzando il colore corrente, il contorno di un
rettangolo. Le ascisse dei lati sinistro e destro del rettangolo
sono rispettivamente x e x+width. Le ordinate dei lati in alto e
in basso del rettangolo sono rispettivamente y e y+height. */
public void drawRect(int x,
                     int y,
                     int width,
                     int height)
```



Disegno di rettangoli arrotondati (roundRect)

- Un rettangolo arrotondato (smussato) è un rettangolo avente gli spigoli smussati: gli spigoli anziché essere squadrati vengono raccordati da un arco di ellisse (circonferenza).

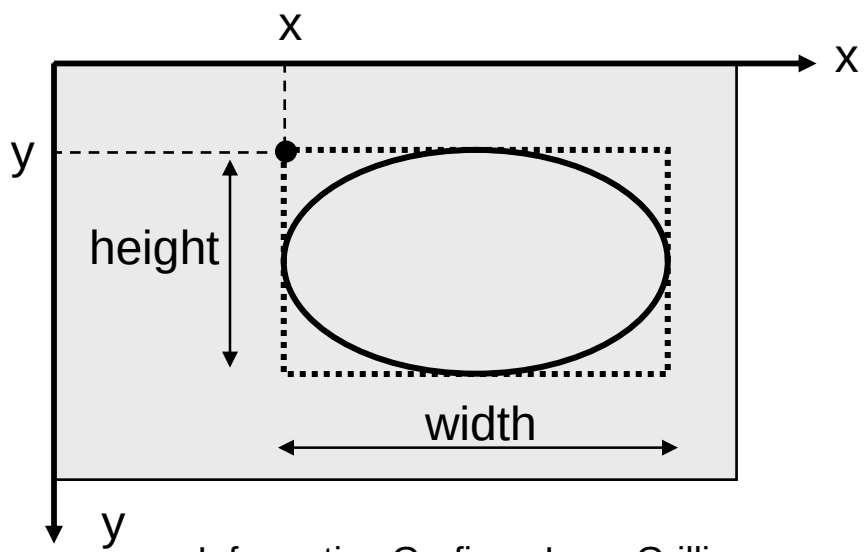


Disegno di rettangoli (contorno) arrotondati

```
/* Disegna, utilizzando il colore corrente, il contorno di un
rettangolo con gli spigoli arrotondati. Le ascisse dei lati
sinistro e destro del rettangolo sono rispettivamente x e
x+width. Le ordinate dei lati in alto e in basso del rettangolo
sono rispettivamente y e y+height. */
public void drawRoundRect(int x,
                          int y,
                          int width,
                          int height,
                          int arcWidth,
                          int arcHeight)
/*
Parametri:
x - coordinata x del rettangolo da disegnare
y - coordinata y del rettangolo da disegnare
width - larghezza del rettangolo da disegnare
height - altezza del rettangolo da disegnare
arcWidth - diametro orizzontale dell'arco ai quattro corner
arcHeight - diametro verticale dell'arco ai quattro corner
*/
```

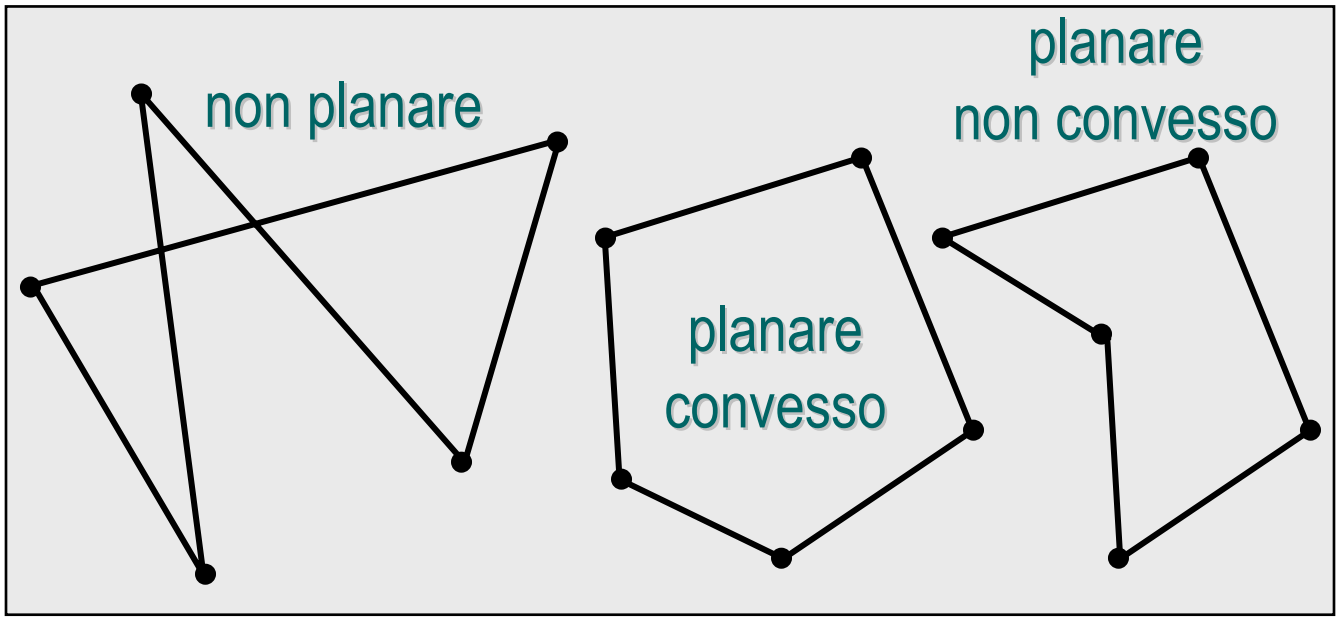
Disegno di ovali (contorno): drawOval( )

```
/* Disegna, utilizzando il colore corrente, il contorno di un
ovale. Il risultato è un ellisse o un cerchio che viene inscritto
all'interno della regione rettangolare specificata dagli
argomenti x, y, width e height. */
public void drawOval(int x,
                     int y,
                     int width,
                     int height)
```



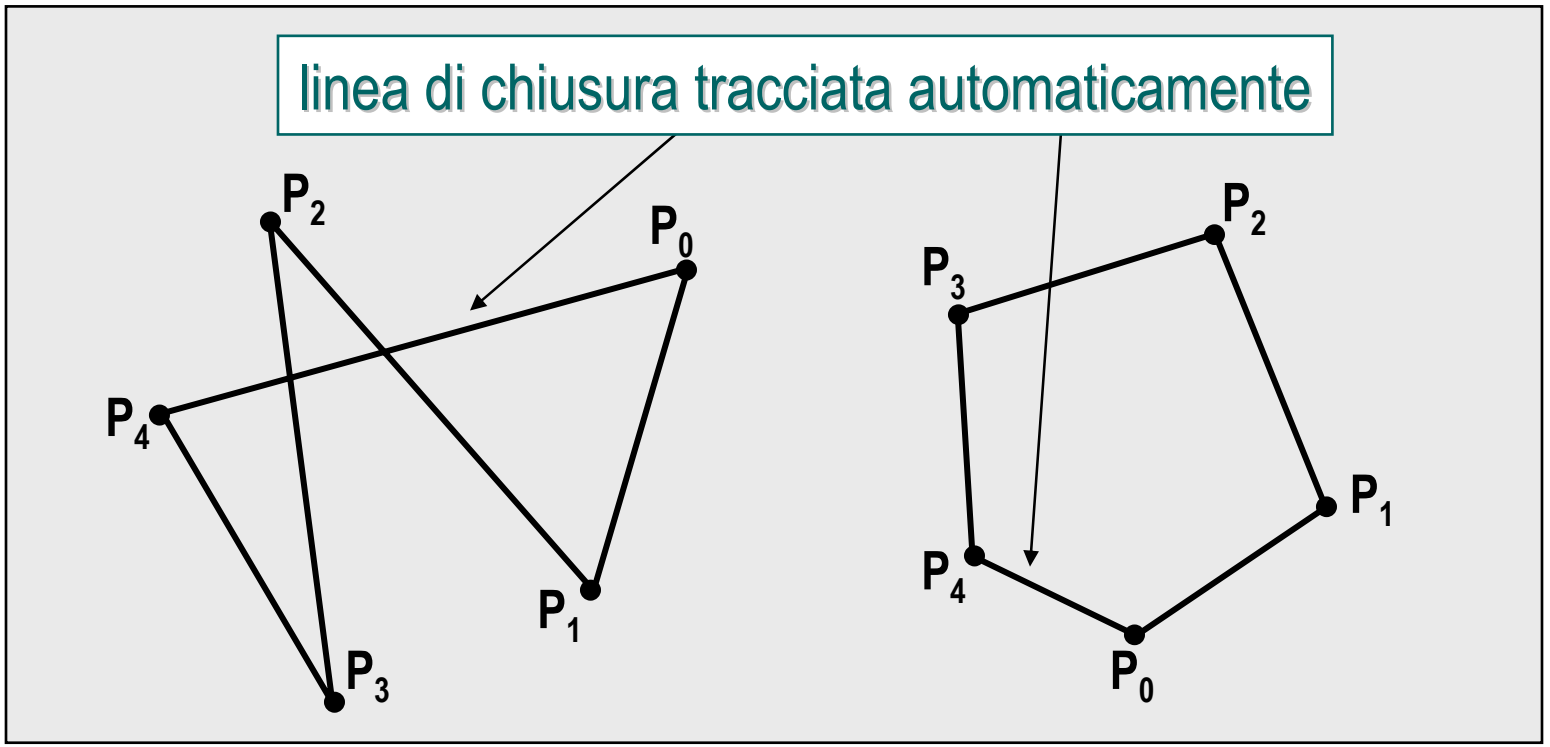
Disegno di poligoni (o polygon)

- Per poligono o polygon, intendiamo una polyline chiusa ovvero una figura geometrica ottenuta connettendo un insieme di linee (segmenti) in modo tale che l'ultima linea termini nell'origine della prima (si tratta di una generalizzazione dei poligoni tradizionali).
- In un poligono ogni vertice (punto) è connesso a due linee $\Rightarrow$ non esistono vertici esterni.
- Un poligono può essere planare, non planare, concavo, convesso, etc..



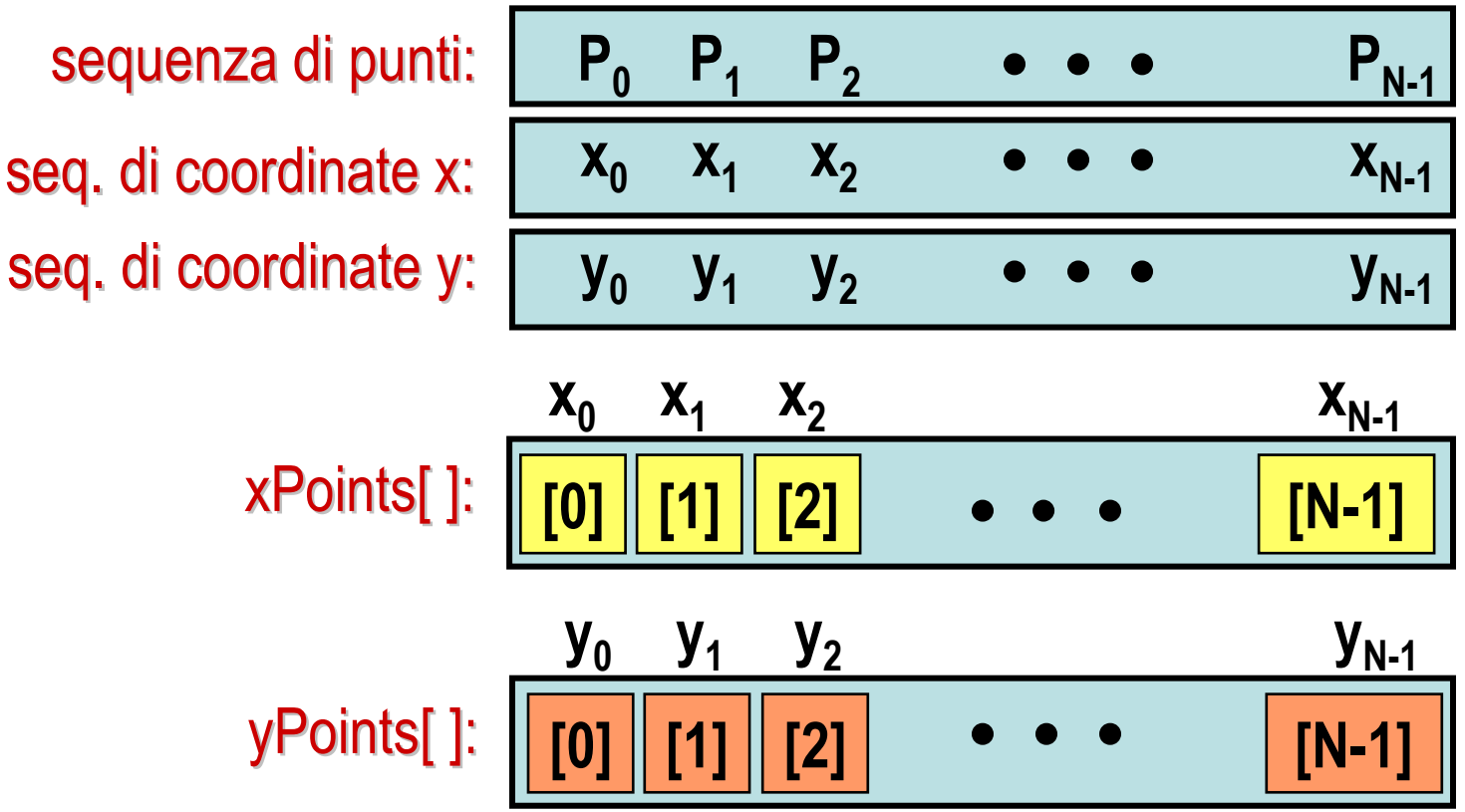
Disegno di un poligoni (o polygon)

- Un poligono di N linee, similmente ad una polyline, è descrivibile elencando la sequenza dei suoi N vertici distinti che si incontrano percorrendo il poligono a partire da un vertice scelto in modo arbitrario: $P_0 \rightarrow P_1 \rightarrow \ldots \rightarrow P_{N-1} \rightarrow P_N$.
- A differenza della polyline viene sempre tracciata la linea che unisce l'ultimo punto P_N con il primo P_0 (tranne quando coincidono).



Disegno di poligoni (o polyline)

- Una poligono di N linee (lati) è pertanto descrivibile specificando due array di N valori numerici contenenti rispettivamente le sequenze delle coordinate x e y della sequenza di punti $\{P_i \equiv (x_i, y_i)\}_{i=0,..,N-1}$: $P_0 \rightarrow P_1 \rightarrow \ldots \rightarrow P_{N-1}$.



Disegno di poligoni: drawPolygon( )

```
/*
Disegna, con il colore corrente, un poligono chiuso (poligonale
chiusa) definito dagli array contenenti le coordinate x e y di
una sequenza dei suoi punti.
Ogni coppia di coordinate (x, y) definisce un punto (vertice) del
poligono. Tale metodo disegna il poligono definito da nPoint
segmenti di linea, dove i primi nPoint-1 segmenti si ottengono
connettendo i punti (xPoints[i-1], yPoints[i-1]) ai punti
(xPoints[i], yPoints[i]), per 1≤i≤nPoints.
La figura è automaticamente chiusa disegnando una linea
congiungente il punto finale con il punto iniziale, se tali punti
sono diversi.*/
public void drawPolygon(int[] xPoints,
                        int[] yPoints,
                        int nPoints)
/*
Parametri:
xPoints – array delle coordinate x dei punti del poligono
yPoints – array delle coordinate y dei punti del poligono
nPoints – numero totale di punti del poligono
*/
```

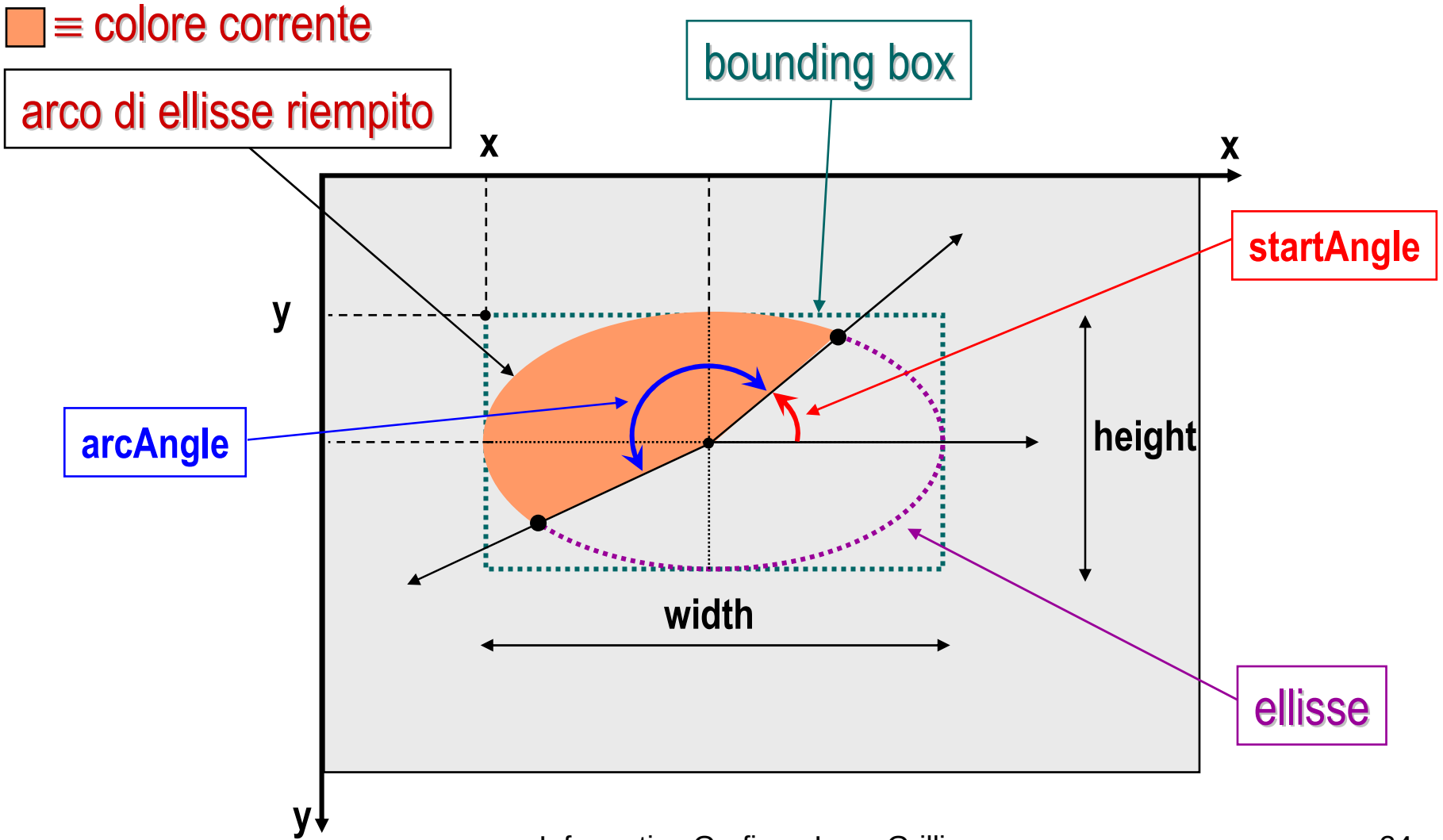
Riempimento di forme chiuse

- Per ciascuna primitiva geometrica che disegna il contorno di una forma chiusa (includendo anche gli archi) è prevista una primitiva geometrica per il riempimento della stessa forma con il colore corrente del pennello grafico.
- Regola mnemonica:
  - I metodi che iniziano per "draw…" disegnano il contorno della forma chiusa.
  - I metodi che iniziano con "fill…" riempiono una forma chiusa con il colore corrente del pennello grafico.
- Ad eccezione del nome il prototipo è lo stesso!

Riempimento di archi: fillArc( )

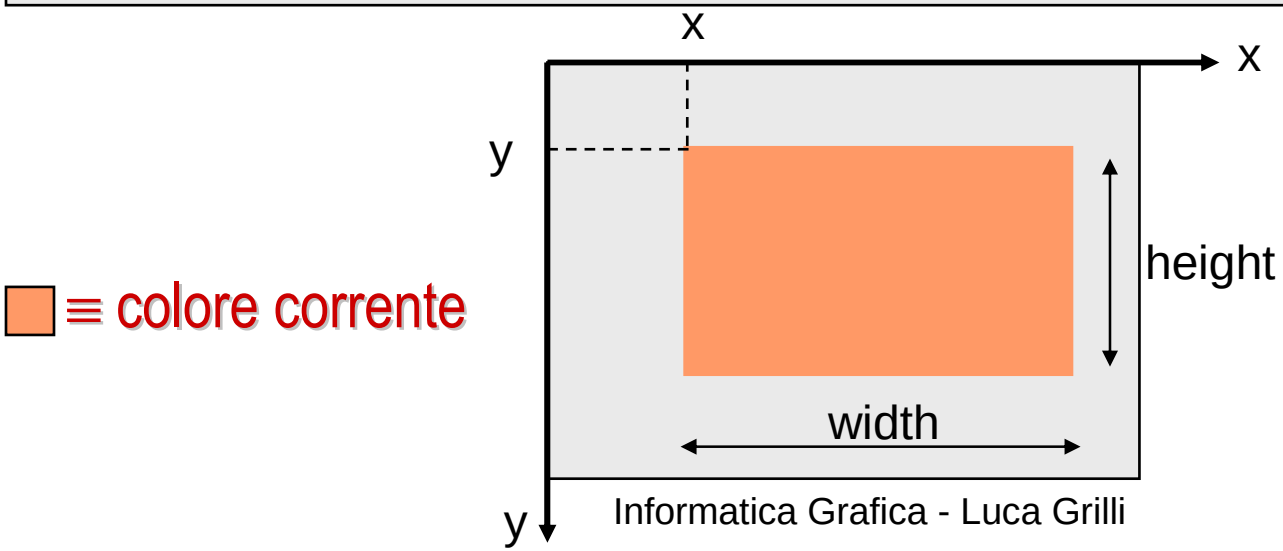
```
/* Riempie, utilizzando il colore corrente, un arco di ellisse
(circonferenza).
L'ellisse viene definita specificando la regione rettangolare
(bounding box: x, y, width, height) in cui è inscritta.
L'arco viene definito specificando la sua estensione angolare
(arcAngle) e l'offset angolare (startAngle), espressi in gradi.
Il riferimento angolare è la semiretta orizzontale, orientata da
sinistra verso destra e che ha origine nel centro dell'ellisse
(bounding box).
Gli angoli positivi (negativi) corrispondono a delle rotazioni
angolari in senso anti-orario (orario) rispetto tale semiretta.
*/
public void fillArc(int x,
                    int y,
                    int width,
                    int height,
                    int startAngle,
                    int arcAngle)
```

Riempimento di archi: fillArc( )



Riempimento di rettangoli fillRect( )

```
/* Riempie, utilizzando il colore corrente, il rettangolo
specificato. Le ascisse dei lati sinistro e destro del rettangolo
sono rispettivamente x e x+width-1. Le ordinate dei lati in alto
e in basso del rettangolo sono rispettivamente y e y+height-1.
Il rettangolo risultante copre un'area avente una larghezza di
width pixel e un'altezza di height pixel.*/
public void fillRect(int x,
                     int y,
                     int width,
                     int height)
```



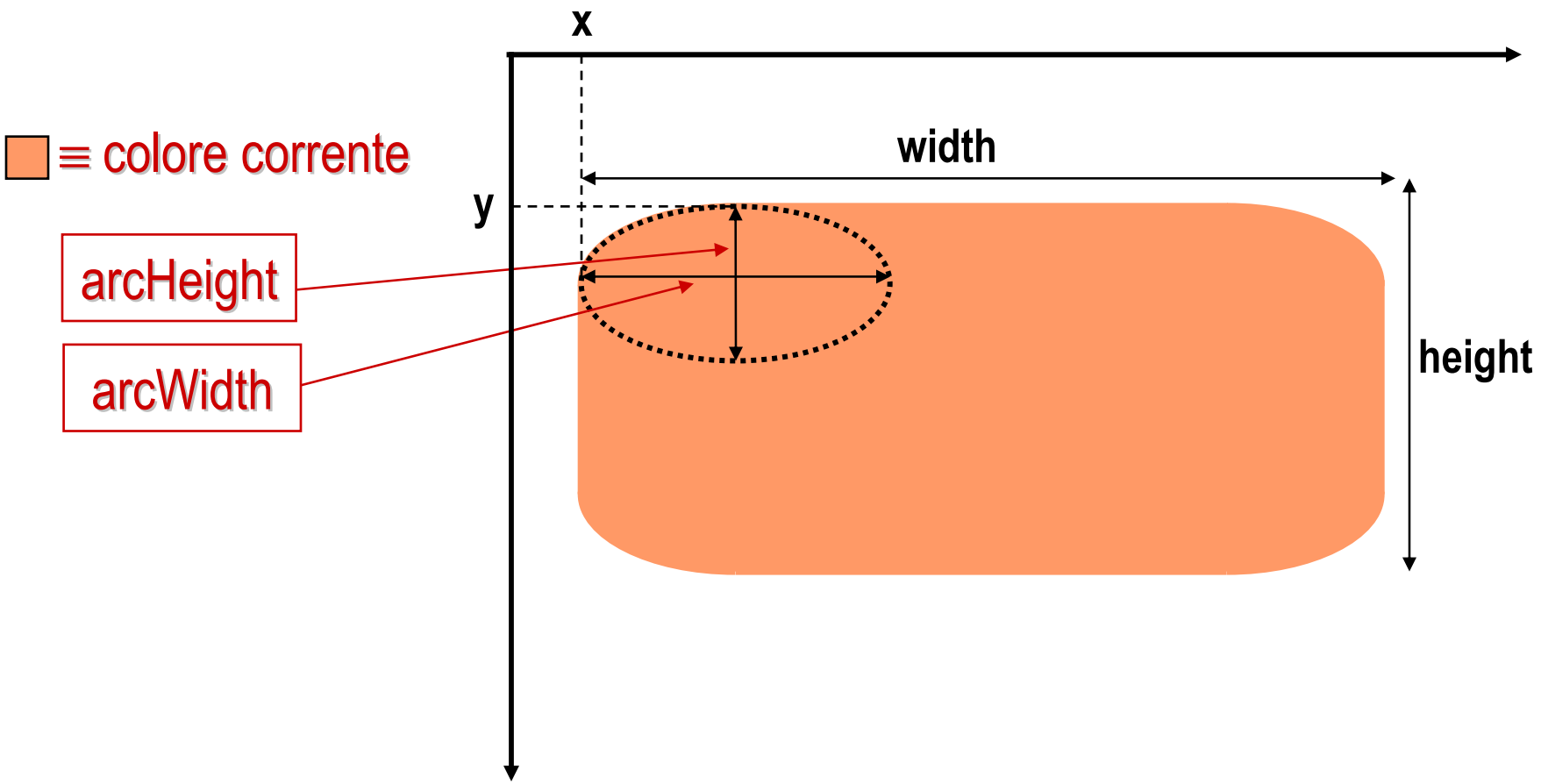
≡ colore corrente

Riempimento di rettangoli arrotondati

```
/* Riempie, utilizzando il colore corrente, il rettangolo
arrotondato specificato. Le ascisse dei lati sinistro e destro
del rettangolo sono rispettivamente x e x+width-1. Le ordinate
dei lati in alto e in basso del rettangolo sono rispettivamente y
e y+height-1. */
public void fillRoundRect(int x,
                          int y,
                          int width,
                          int height,
                          int arcWidth,
                          int arcHeight)
/*
Parametri:
x - coordinata x del rettangolo da riempire
y - coordinata y del rettangolo da riempire
width - larghezza del rettangolo da riempire
height - altezza del rettangolo da riempire
arcWidth - diametro orizzontale dell'arco ai quattro corner
arcHeight - diametro verticale dell'arco ai quattro corner
*/
```

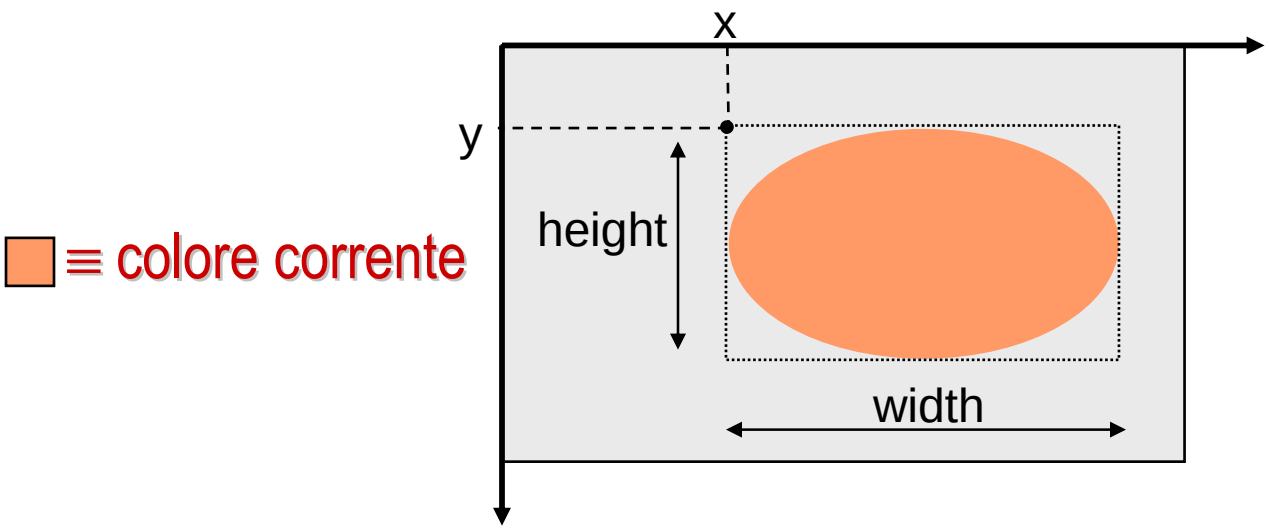
Riempimento di rettangoli arrotondati

- Esempio di utilizzo del metodo fillRoundRect( ).



Riempimento di ovali: fillOval( )

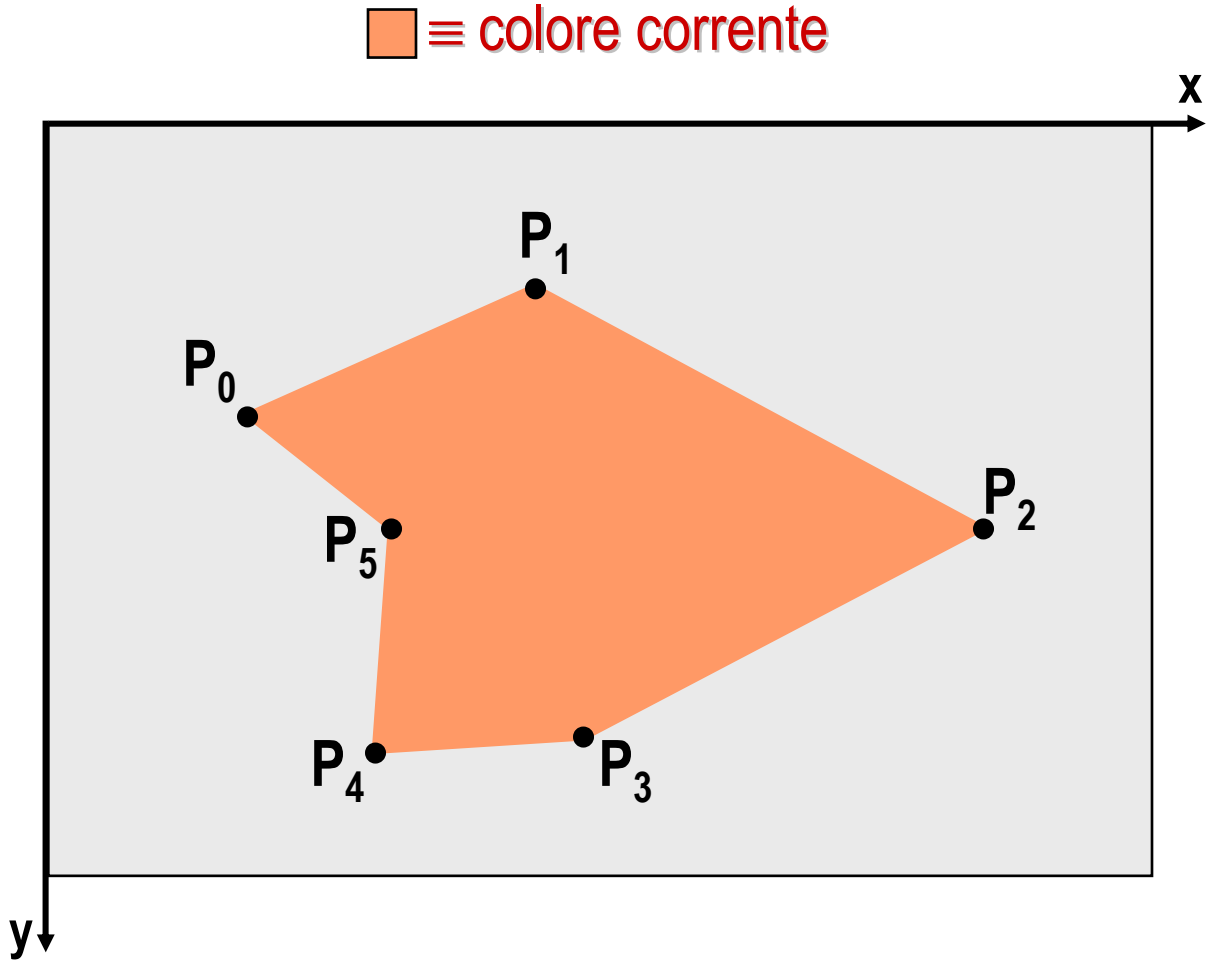
```
/* Riempie, utilizzando il colore corrente, l'ovale inscritto nel
rettangolo specificato.*/
public void fillOval(int x,
                     int y,
                     int width,
                     int height)
```



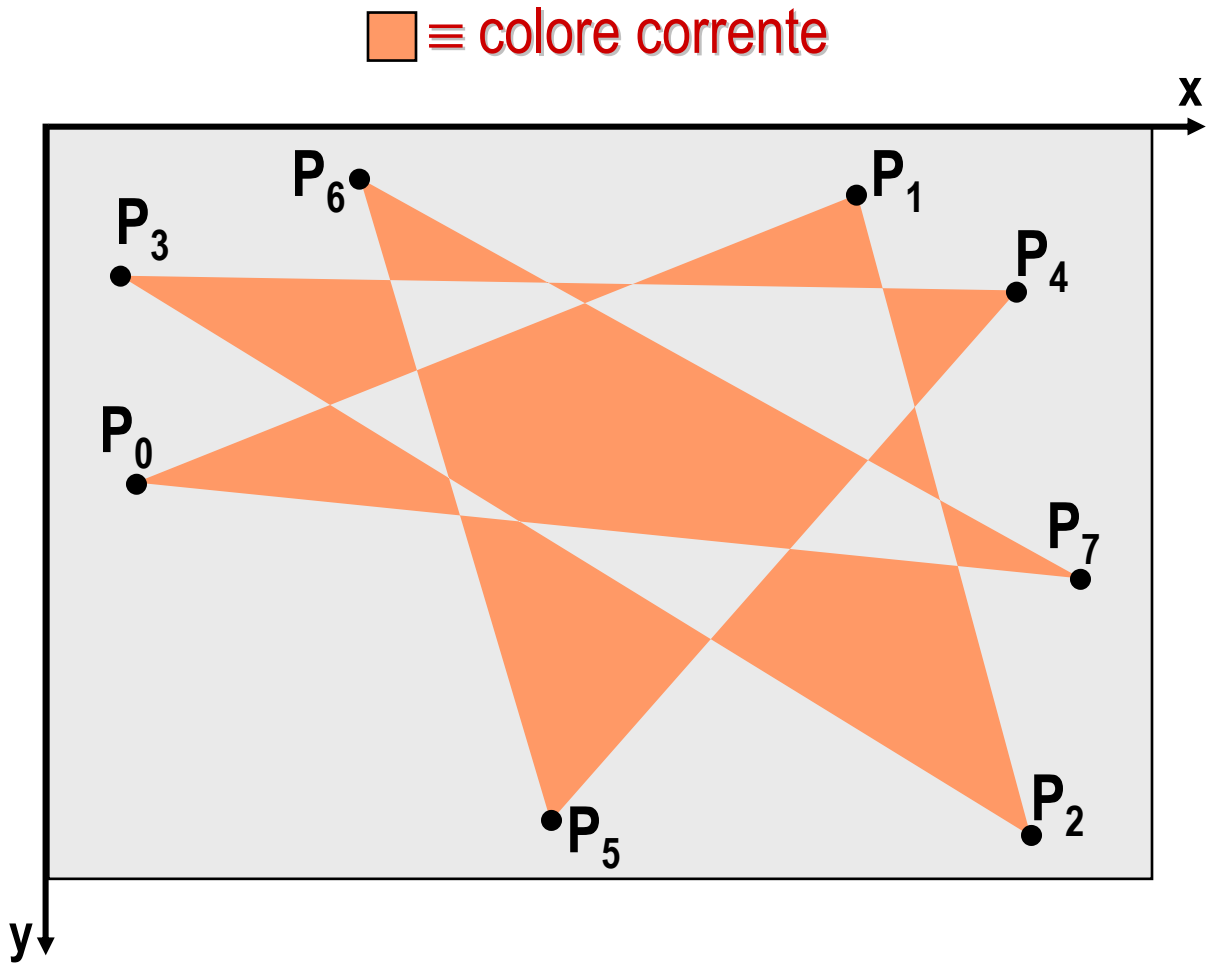
Riempimento di poligoni: fillPolygon( )

```
/* Riempie, con il colore corrente, un poligono chiuso
(poligonale chiusa) definito dagli array contenenti le coordinate
x e y di una sequenza dei suoi punti.
Tale metodo disegna il poligono definito da nPoint segmenti di
linea, dove i primi nPoint-1 segmenti si ottengono connettendo i
punti (xPoints[i-1], yPoints[i-1]) ai punti (xPoints[i],
yPoints[i]), per 1≤i≤nPoints.
La figura è automaticamente chiusa disegnando una linea
congiungente il punto finale con il punto iniziale, se tali punti
sono diversi.
L'area all'interno è definita usando la regola di riempimento
pari-dispari anche nota come regola dell'alternanza.*/
public void fillPolygon(int[] xPoints,
                        int[] yPoints,
                        int nPoints)
/*
Parametri:
xPoints - array delle coordinate x dei punti del poligono
yPoints - array delle coordinate y dei punti del poligono
nPoints - numero totale di punti del poligono
*/
```

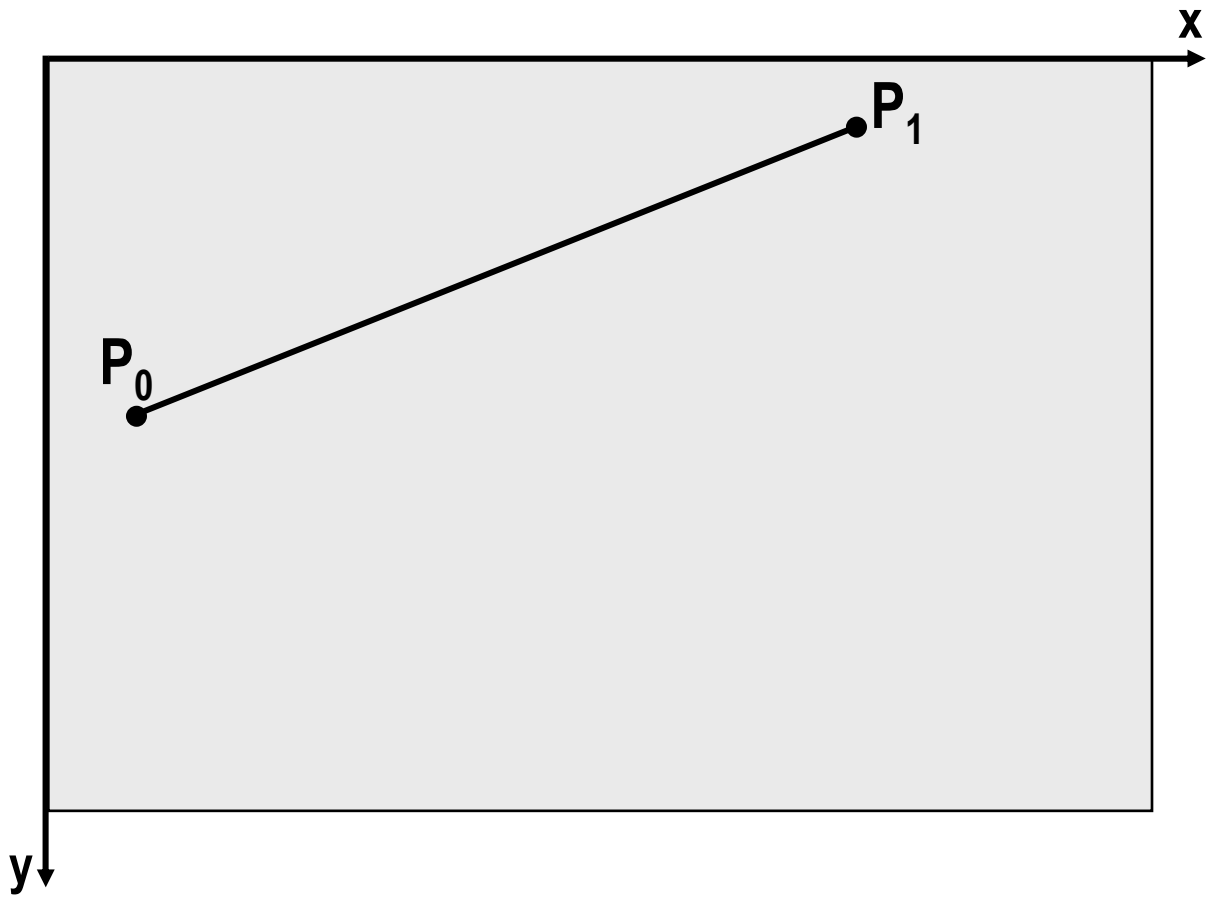
fillPolygon( ): esempio per poligono planare



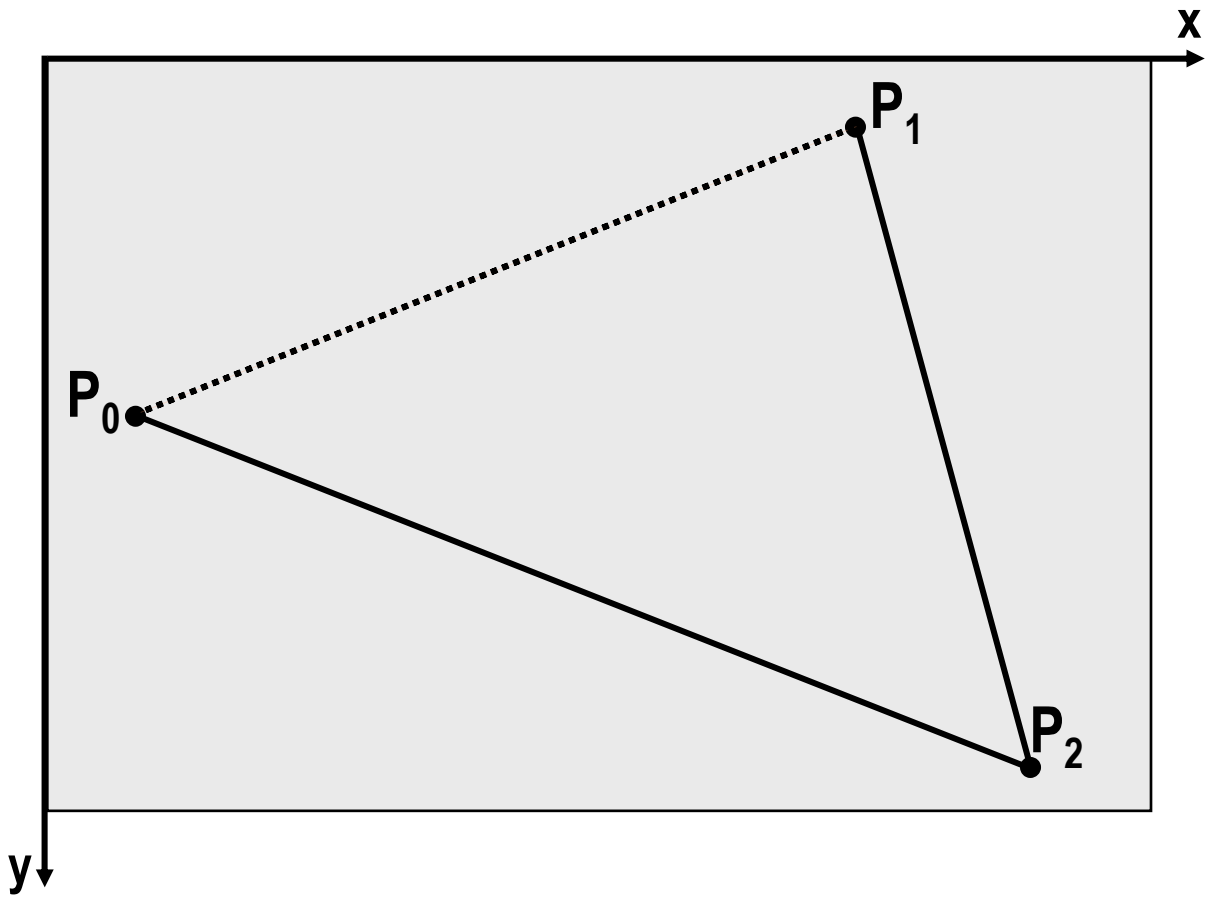
Regola di riempimento pari-dispari



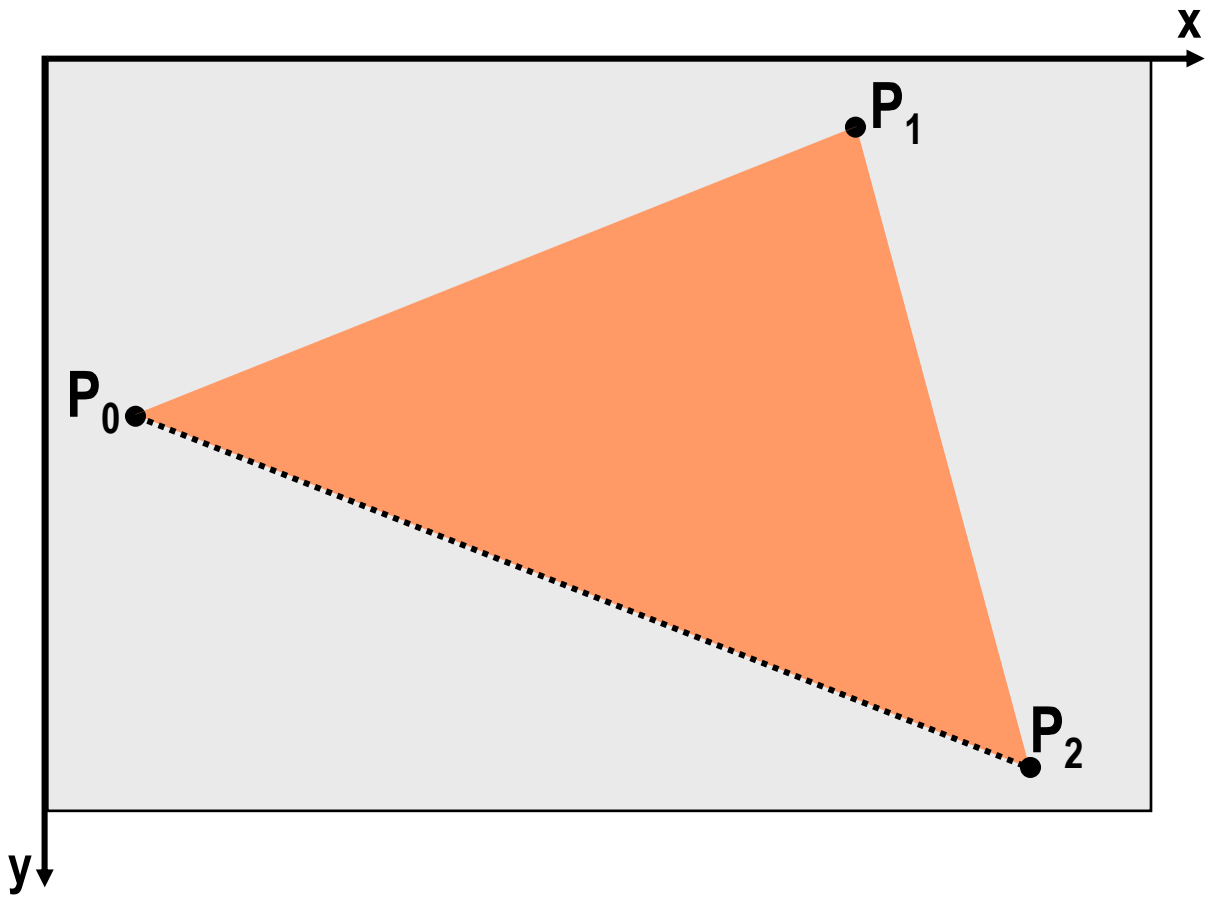
Regola di riempimento pari-dispari



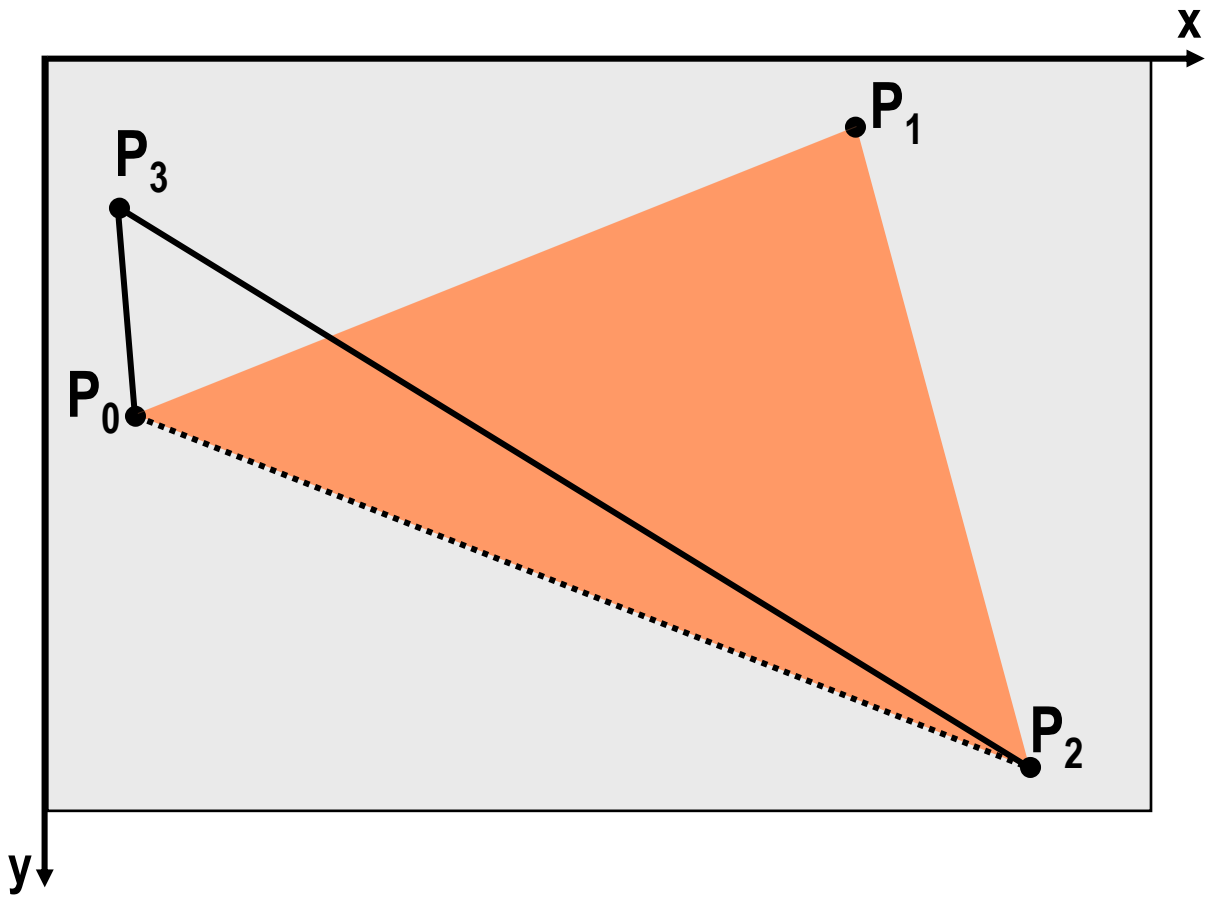
Regola di riempimento pari-dispari



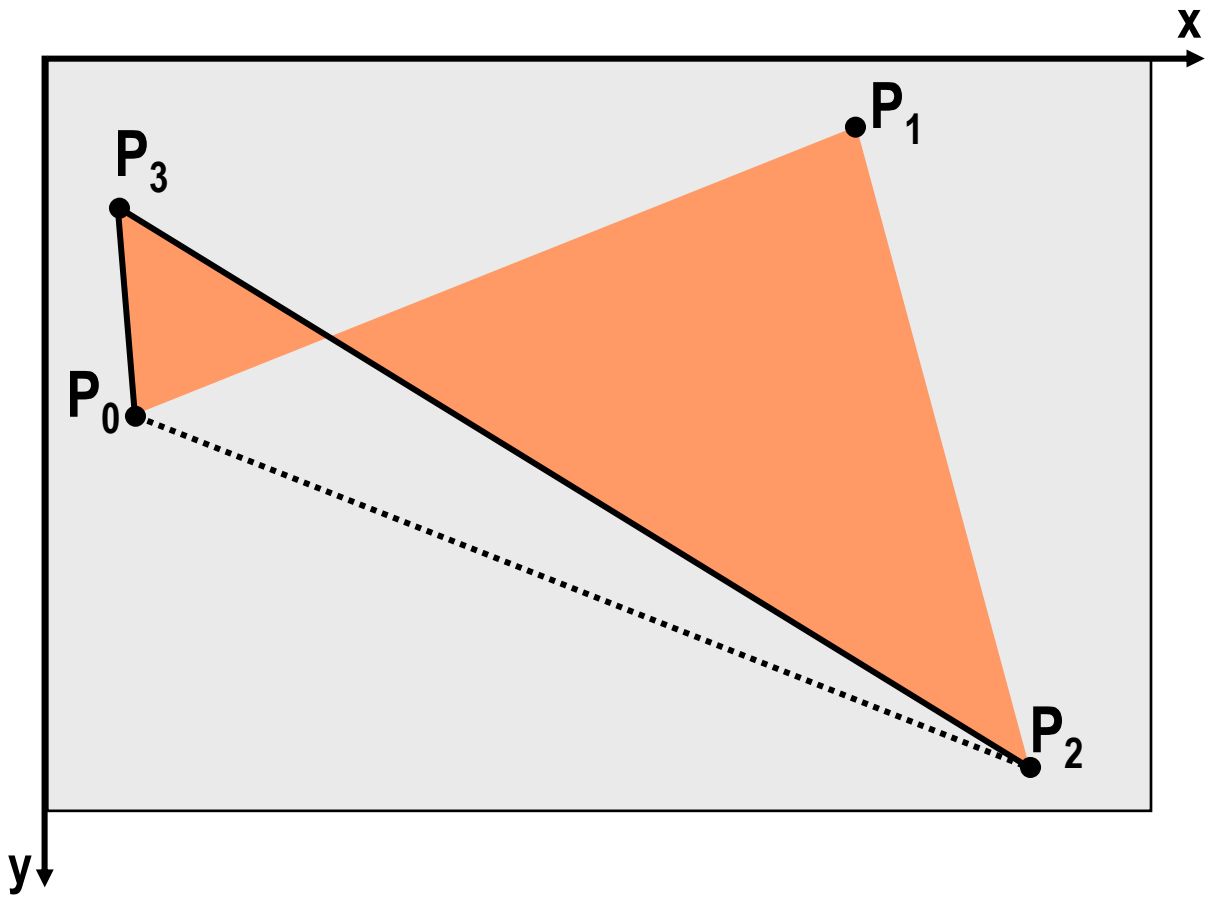
Regola di riempimento pari-dispari



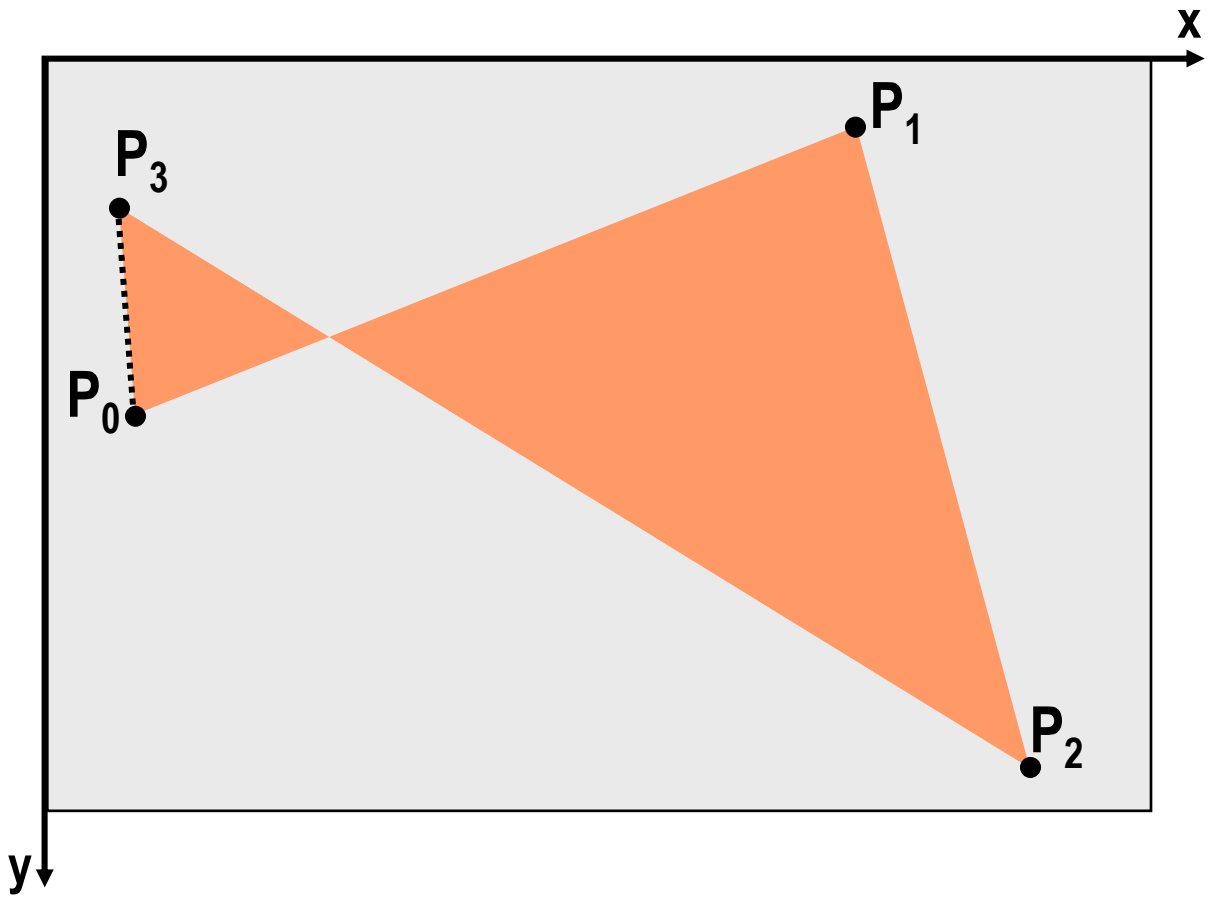
Regola di riempimento pari-dispari



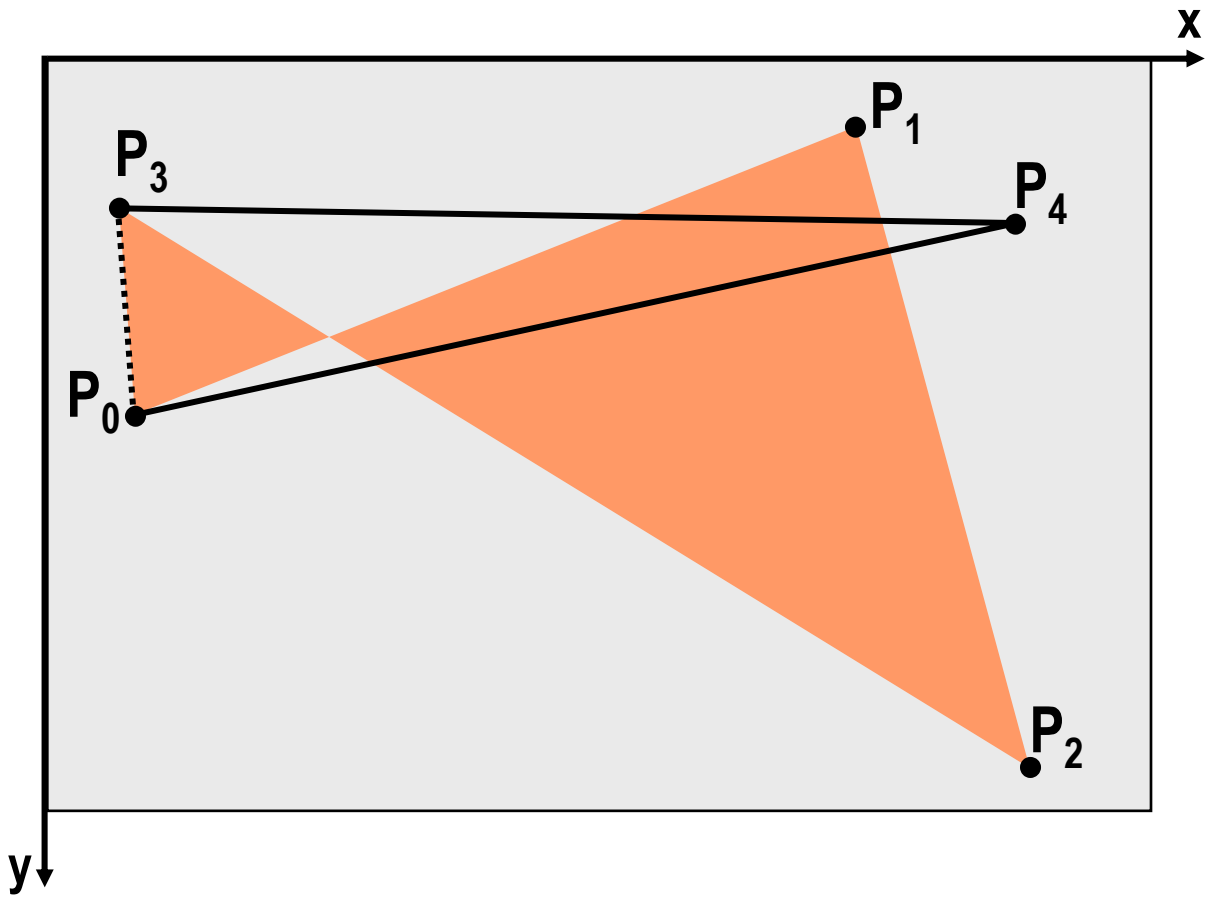
Regola di riempimento pari-dispari



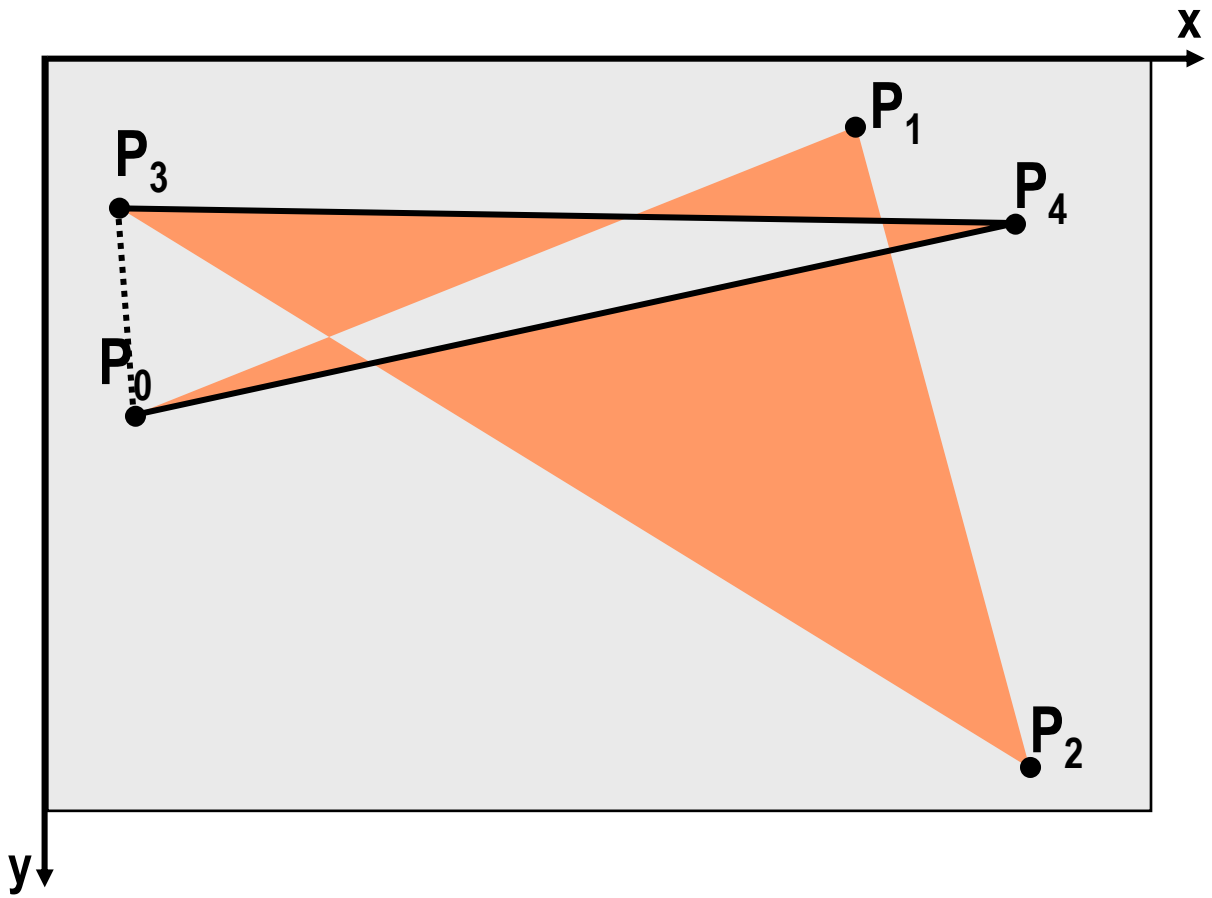
Regola di riempimento pari-dispari



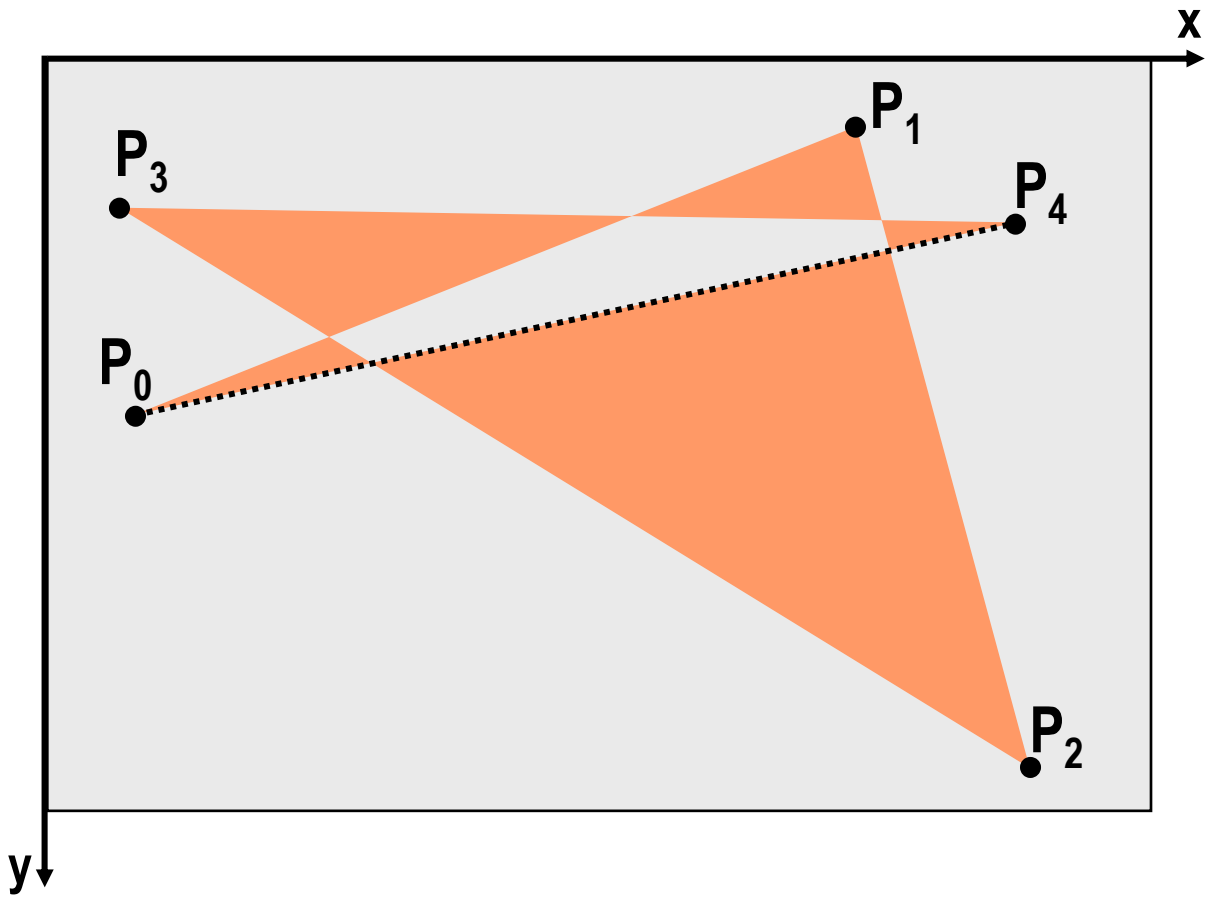
Regola di riempimento pari-dispari



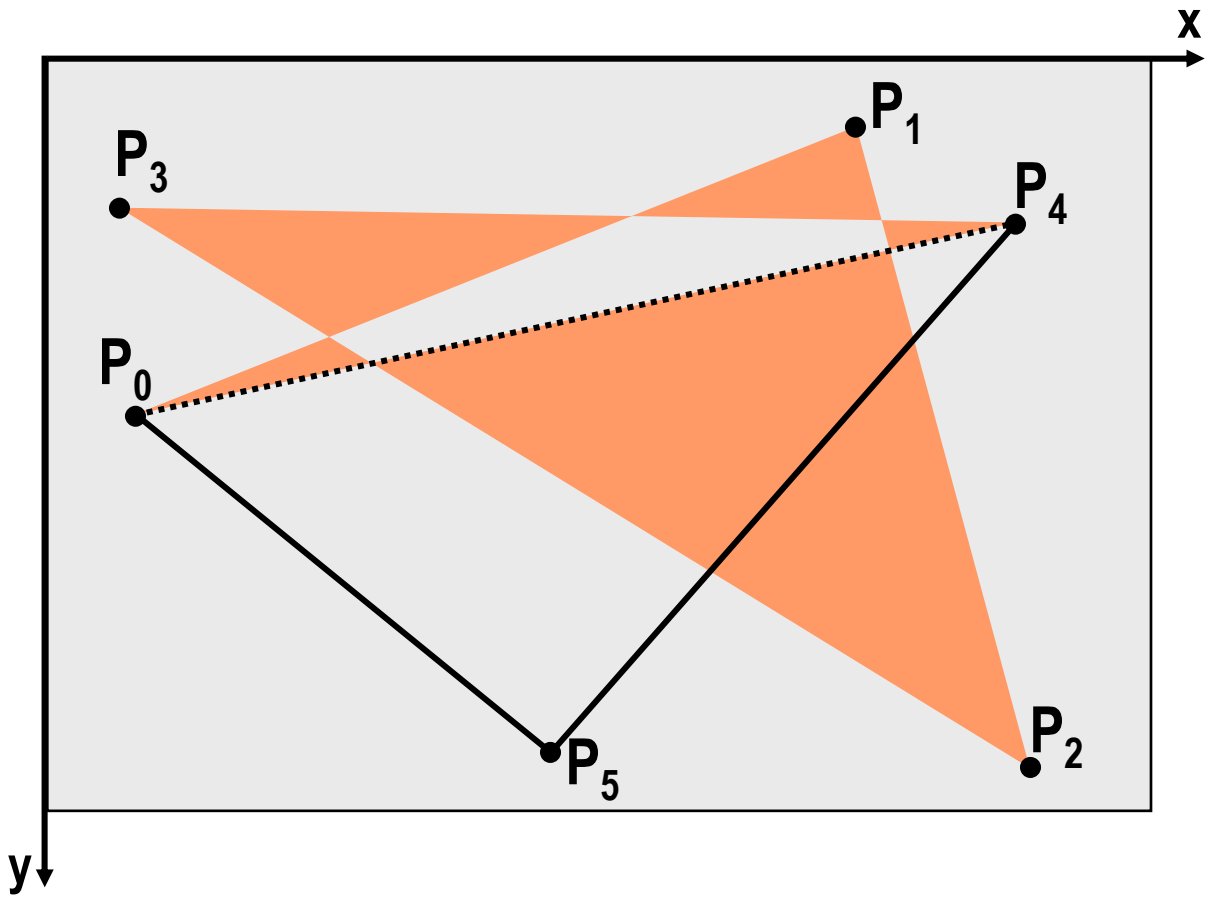
Regola di riempimento pari-dispari



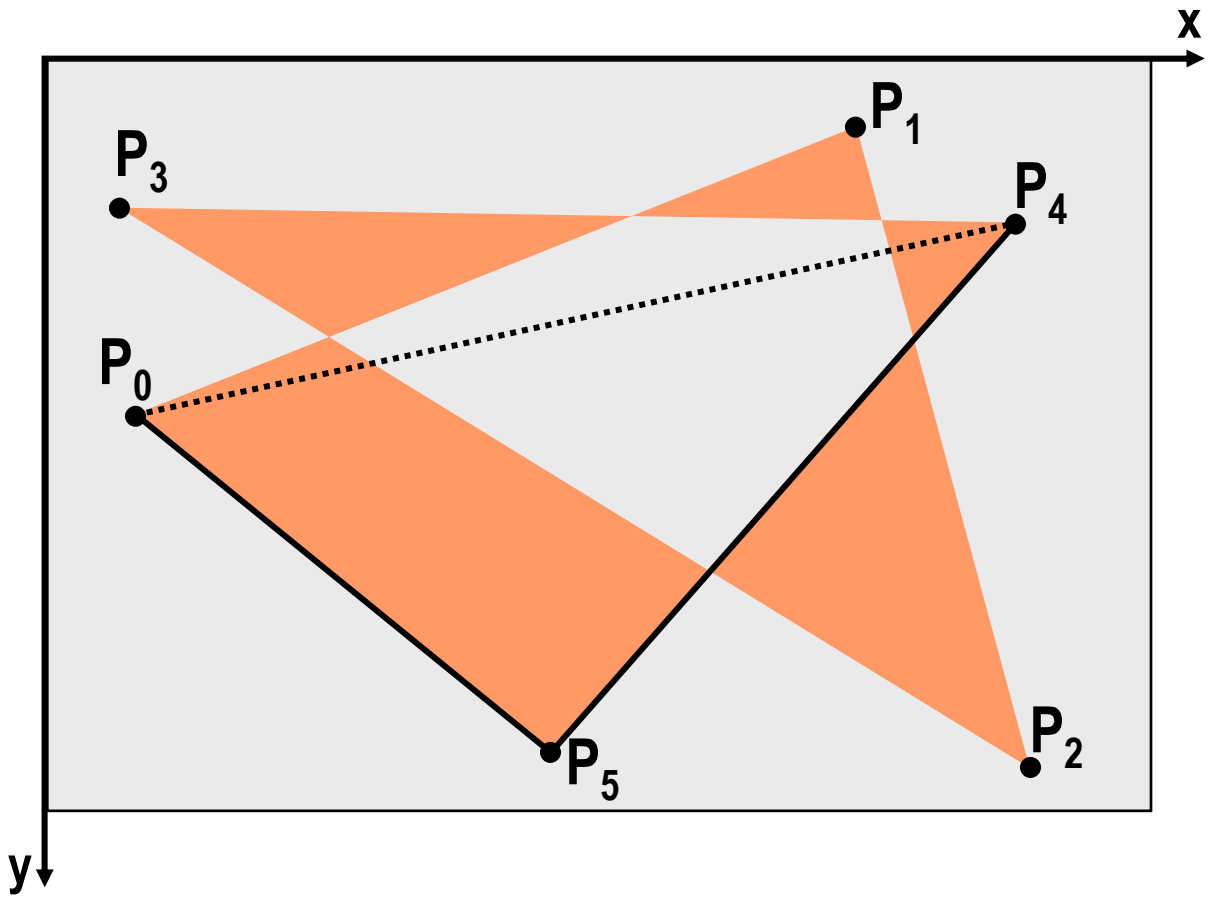
Regola di riempimento pari-dispari



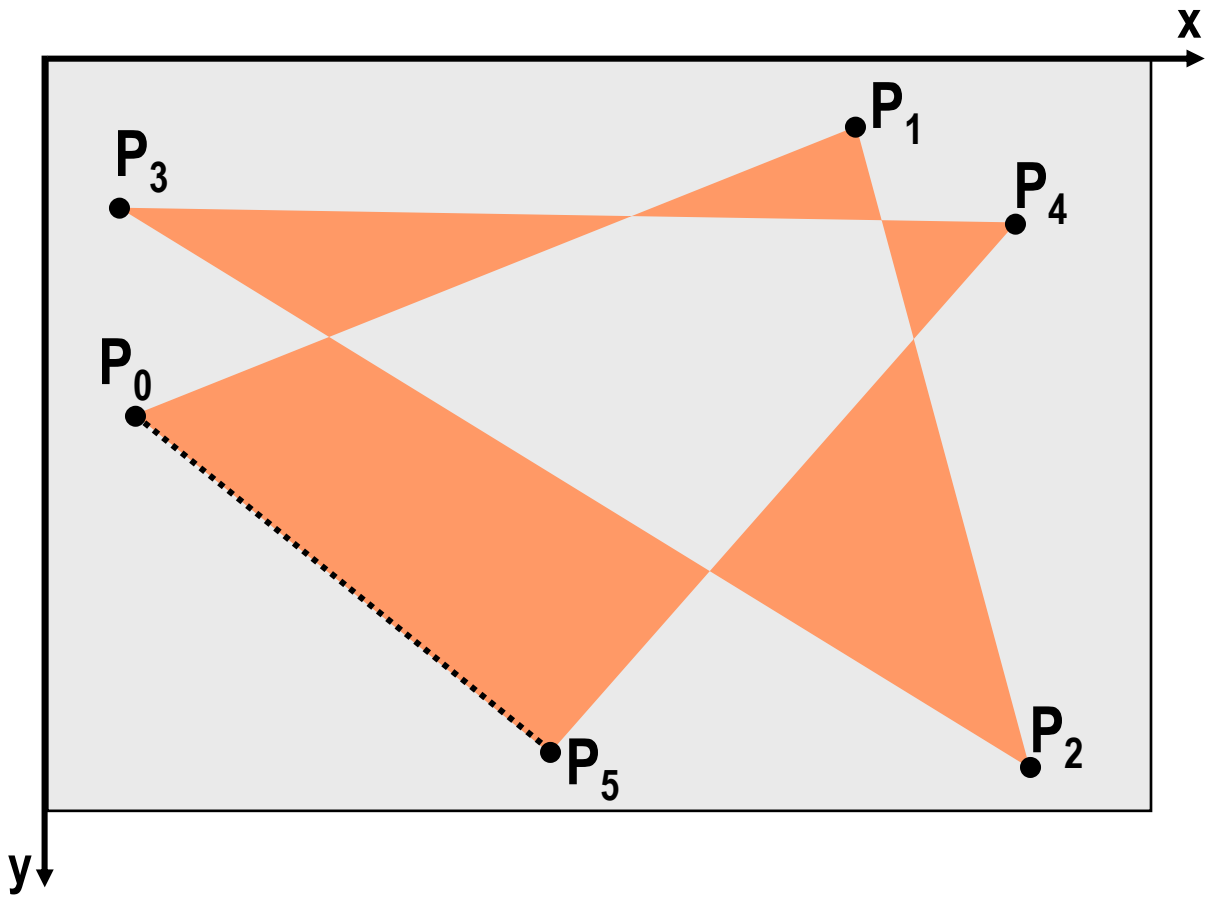
Regola di riempimento pari-dispari



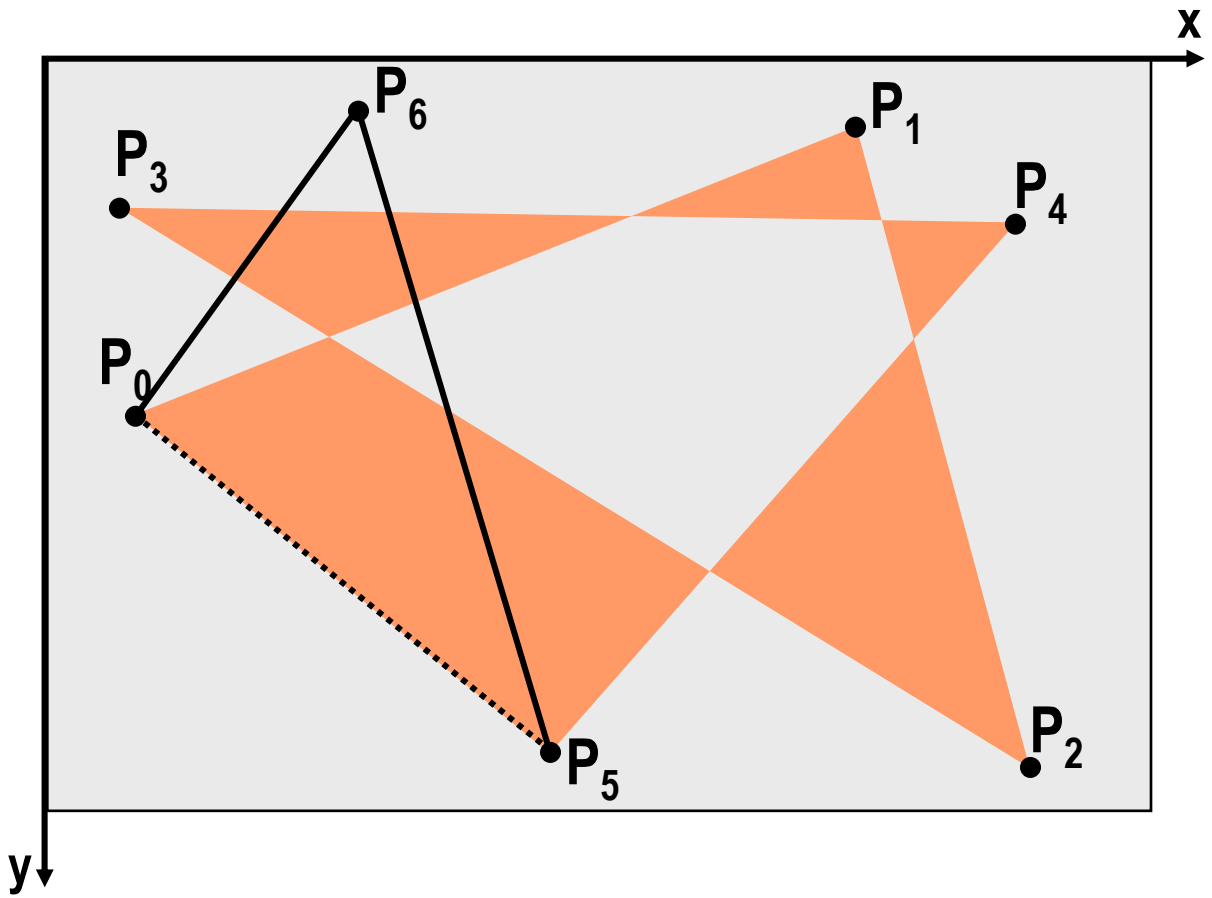
Regola di riempimento pari-dispari



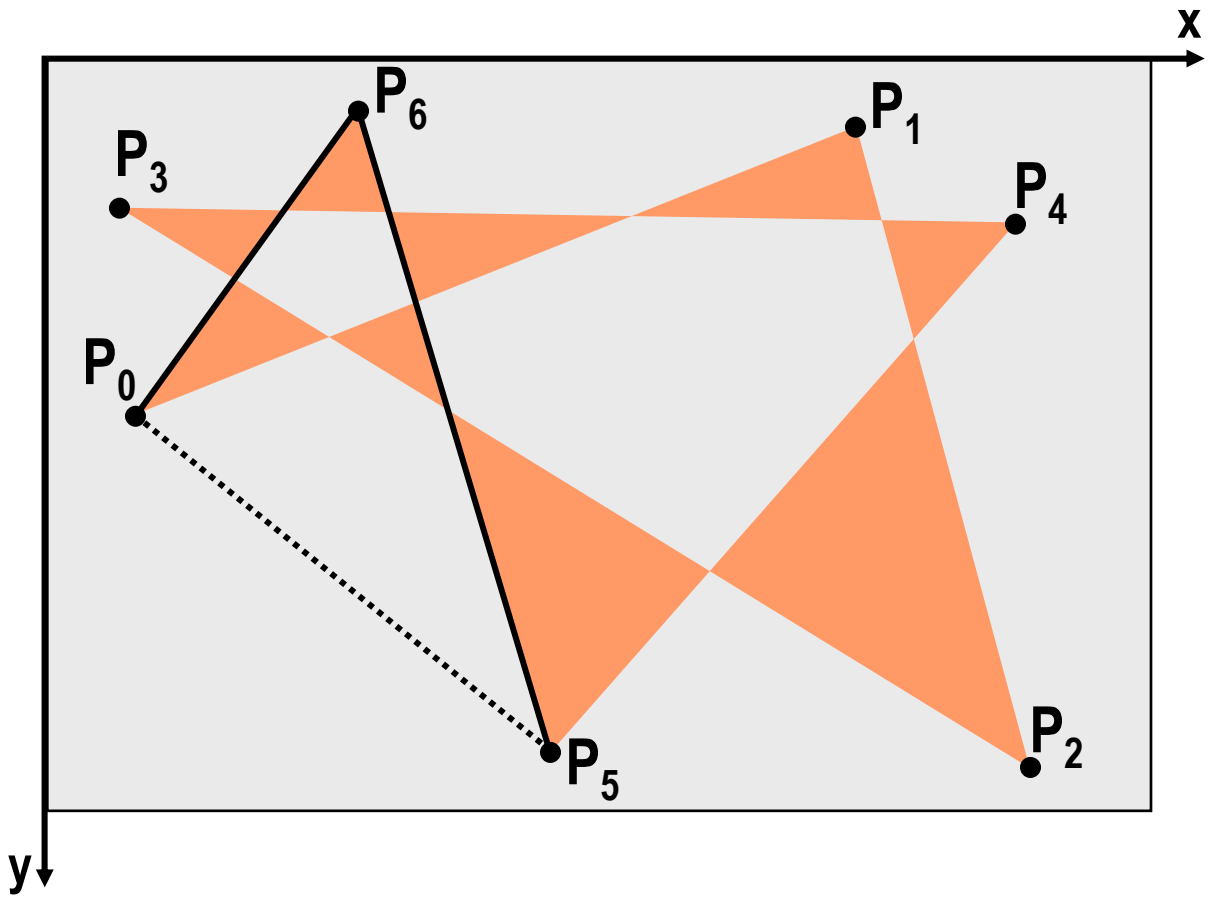
Regola di riempimento pari-dispari



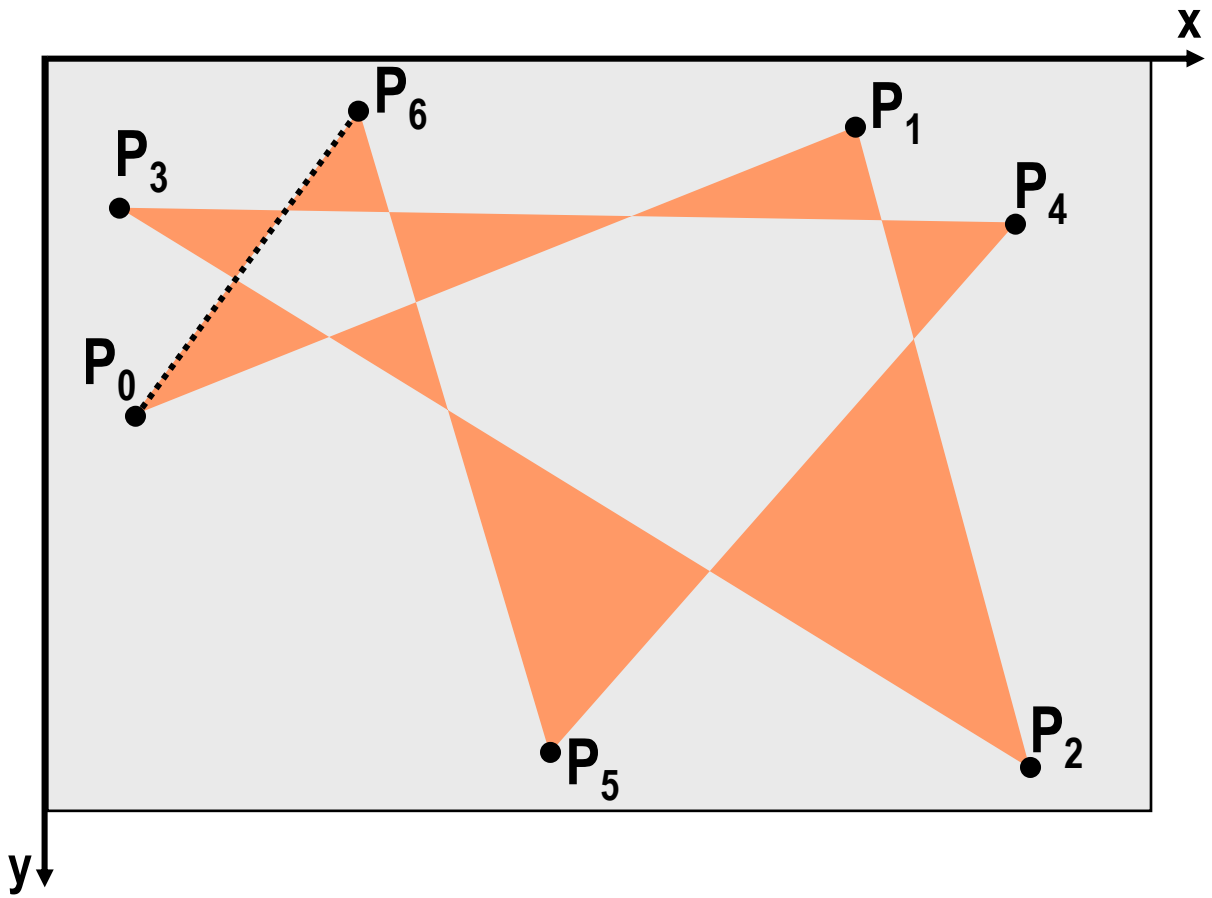
Regola di riempimento pari-dispari



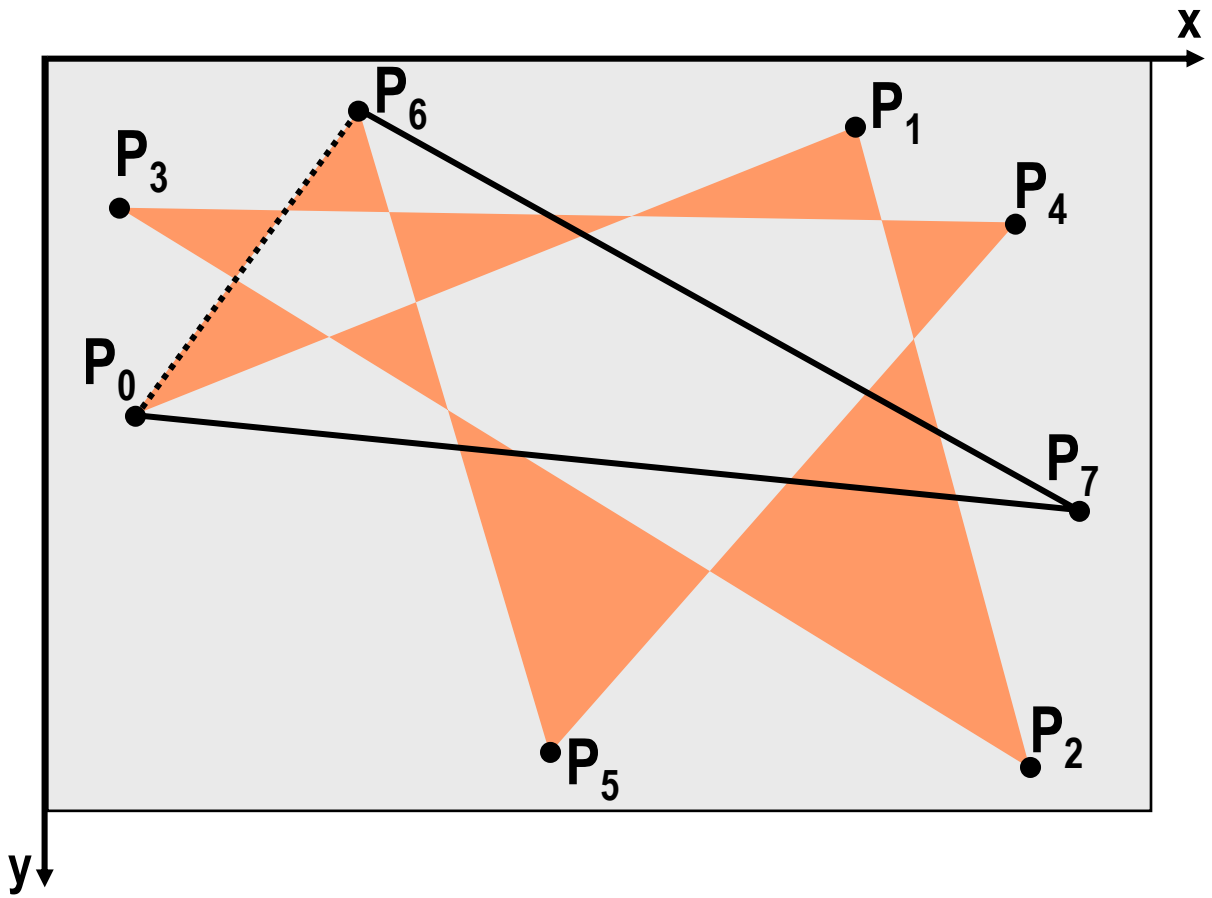
Regola di riempimento pari-dispari



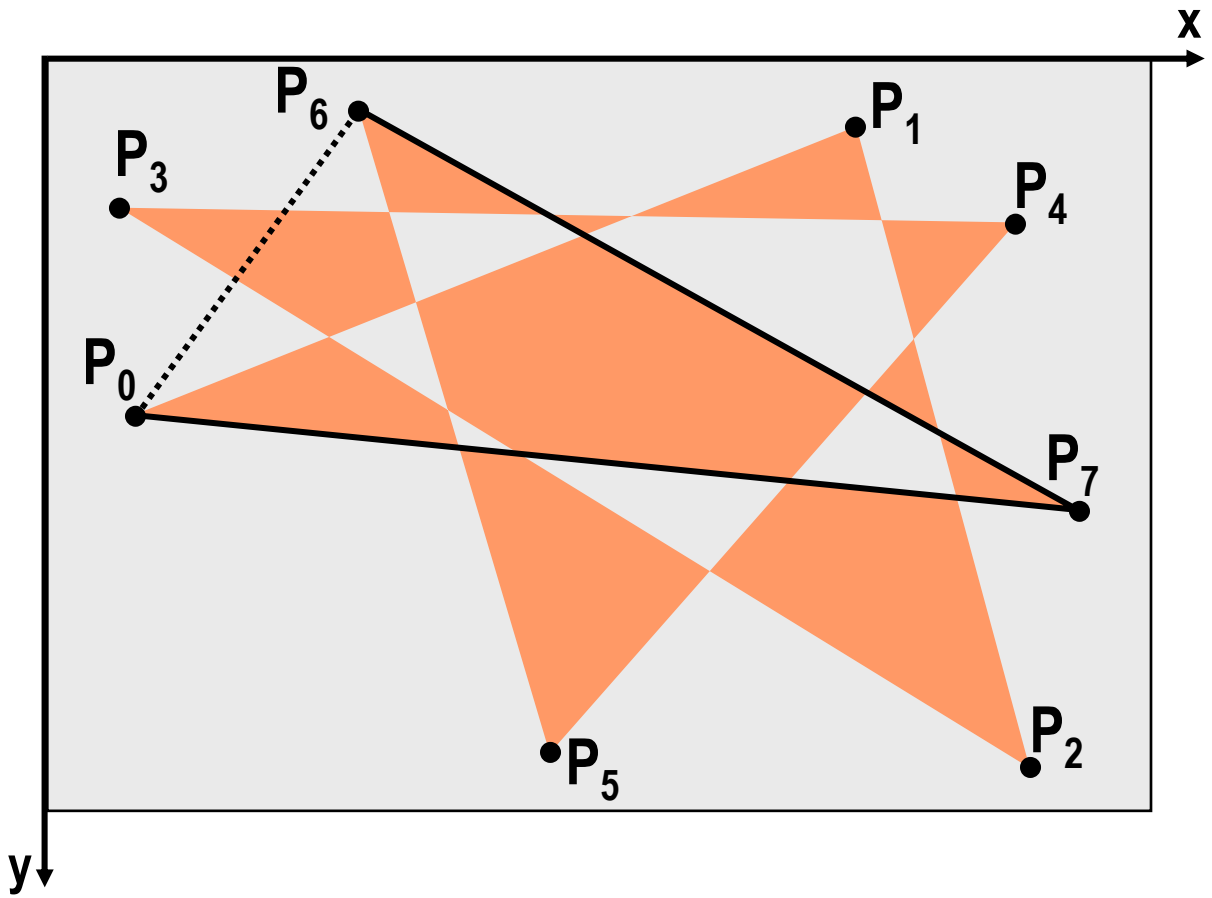
Regola di riempimento pari-dispari



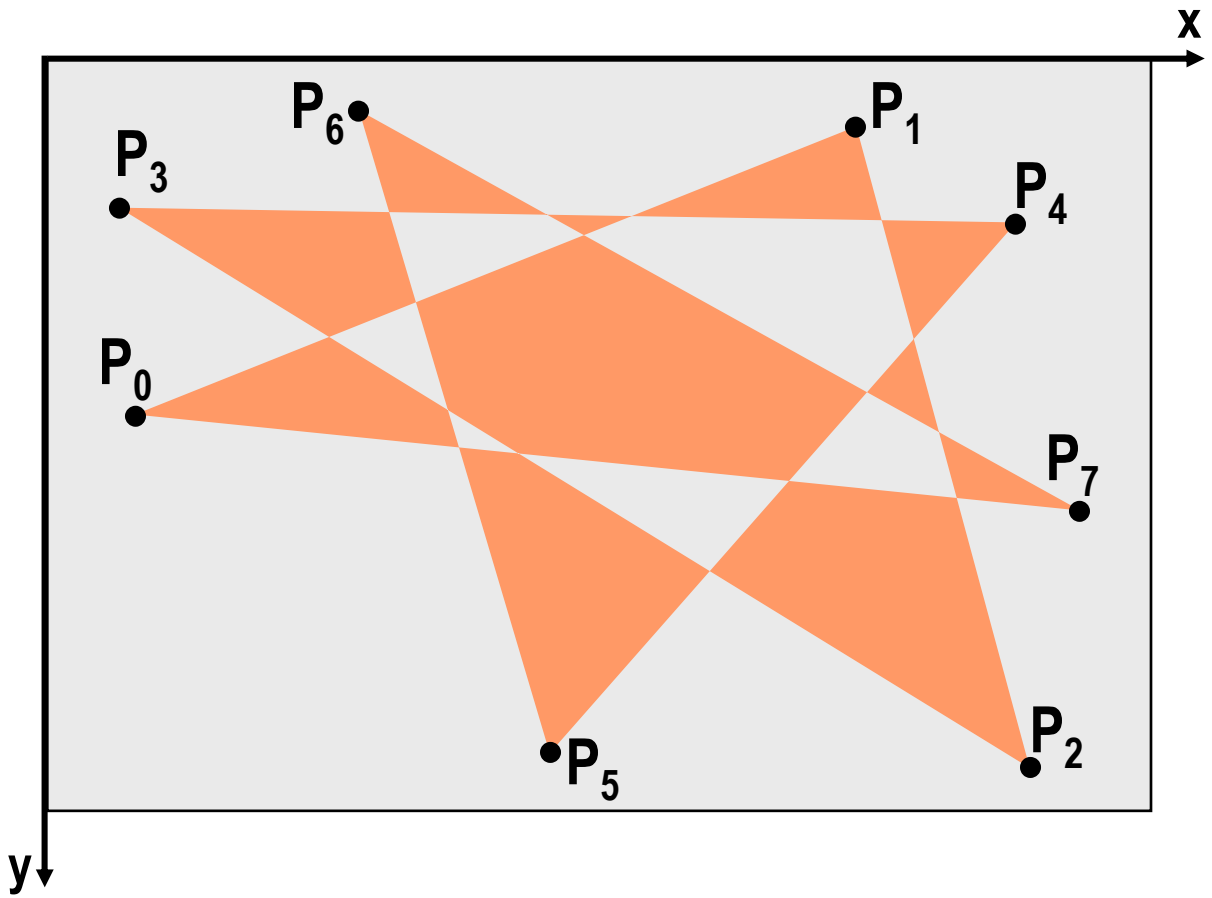
Regola di riempimento pari-dispari



Regola di riempimento pari-dispari



Regola di riempimento pari-dispari



Cancellazione di una regione rettangolare

- L'oggetto Graphics prevede anche un metodo clearRect( ) per cancellare il contenuto di una regione rettangolare ripristinando il colore di background del componente grafico sul quale si sta disegnando (JPanel, JFrame).

```
/* Cancella la regione rettangolare specificata riempiendola con
il colore di sfondo del superficie di disegno corrente.*/
public void clearRect(int x,
                      int y,
                      int width,
                      int height)
```

Visualizzazione di stringhe

La classe java.awt.Font

- Nelle prossime slide studieremo i metodi e le costanti per il controllo dei tipi di caratteri messi a disposizione dalla classe java.awt.Font.
- Il tipico costruttore della classe Font prende tre argomenti: il nome, lo stile e la dimensione del tipo di carattere.
- Il nome è definito da una stringa rappresentante un qualsiasi tipo di carattere.
  - Per nome si intende sia uno specifico font che una particolare famiglia.
  - Java distingue tra font fisici (o reali) e font logici.
  - I font fisici sono quelli effettivamente installati nel sistema in uso; ad esempio: Times, Helvetica, Courier, etc.
  - Per font logici si intendono le cinque famiglie di font definite dall'ambiente Java e che devono essere supportate da ogni Java Runtime Environment: Serif, SansSerif, Monospaced, Dialog e DialogInput.

```
/* Cea un nuovo font avente il nome, lo stile e la dimensione
specificati. */
public Font(String nome, int style, int size)
```

La classe java.awt.Font

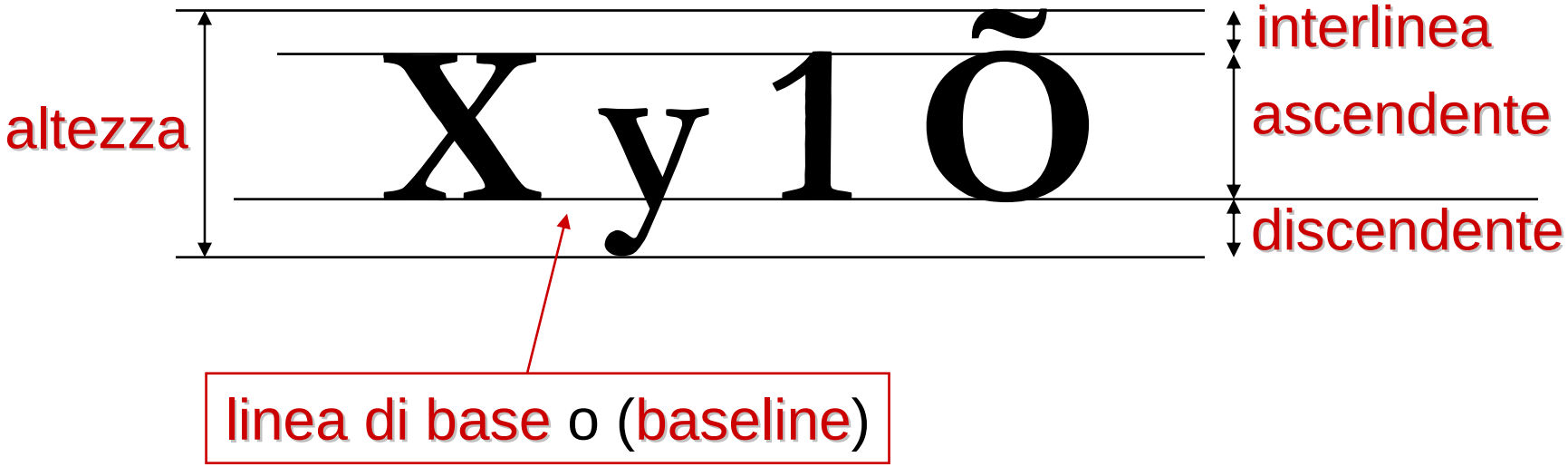
- Lo stile (style) del carattere, codificato con una costante statica della classe Font, può essere:
  - PLAIN (NORMALE) ⇒ Font.PLAIN
  - BOLD (GRASSETTO) ⇒ Font.BOLD
  - ITALIC (CORSIVO) ⇒ Font.ITALIC
  - BOLD + ITALIC (GRASSETTO + CORSIVO) ⇒ Font. BOLD + Font.ITALIC.
- La dimensione del carattere è misurata in punti, dove un punto corrisponde a 1/72 di pollice.

```
/* Cea un nuovo font avente il nome, lo stile e la dimensione
specificati. */
public Font(String nome, int style, int size)
```

Informazioni metriche di un carattere

- Spesso è necessario conoscere delle informazioni precise riguardo la metrica di un carattere. Generalmente si fa riferimento alle seguenti metriche.
  - Altezza: estensione verticale del carattere.
  - Line di base (baseline): linea orizzontale di riferimento ove poggia la "parte principale" del carattere.
  - Discendente: parte di carattere che si estende al di sotto della linea base del carattere stesso.
  - Ascendente: parte di carattere che si estende al di sopra della linea base del carattere stesso.
  - Interlinea: differenza tra l'altezza e l'ascendente.

Informazioni metriche di un carattere



Visualizzazione di una stringa

- Un oggetto Graphics prevede il metodo drawString( ) per la visualizzazione di una stringa.
- Tale metodo visualizza la stringa con il colore e font corrente del pennello grafico.
- Tale metodo riceve in input le coordinate (x, y) della baseline del carattere più a sinistra.

```
/* Disegna il testo specificato dalla stringa, usando il font e
il colore corrente del contesto grafico. La baseline del
carattere più a sinistra viene posizionata nel punto di
coordinate (x, y) del sistema di coordinate del contesto grafico.
*/
public void drawString(String str,
                       int x,
                       int y)
```

Visualizzazione di una stringa

- Un oggetto Graphics prevede il metodo drawString( ) per la visualizzazione di una stringa.
- Tale metodo visualizza la stringa con il colore e font corrente del pennello grafico.
- Tale metodo riceve in input le coordinate (x, y) della baseline del carattere più a sinistra.

```
/* Disegna il testo specificato dalla stringa, usando il font e
il colore corrente del contesto grafico. La baseline del
carattere più a sinistra viene posizionata nel punto di
coordinate (x, y) del sistema di coordinate del contesto grafico.
*/
public void drawString(String str,
                       int x,
                       int y)
```

Modifica del font

- Un oggetto Graphics prevede i metodi:
  - setFont( ) per la modifica del font corrente del contesto grafico.
  - setColor( ) per la modifica del colore corrente del contesto grafico.

```
/* Imposta il font specificato come font per il contesto grafico.
Tutte le successive operazioni sui testi che utilizzano questo
contesto grafico utilizzeranno il nuovo font impostato.
*/
public void setFont(Font font)
```

```
/* Imposta il colore specificato come colore per il contesto
grafico. Tutte le successive operazioni grafiche che utilizzano
questo contesto grafico utilizzeranno il nuovo colore impostato.
*/
public void setColor(Color color)
```