
Il contesto grafico Graphics 2D

Luca Grilli

Da Graphics a Graphics2D

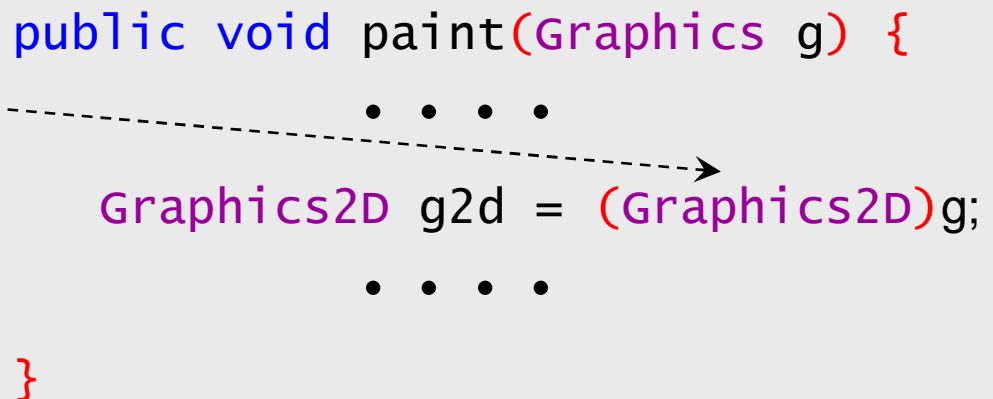
- Nelle precedenti lezioni abbiamo utilizzato un oggetto di tipo `Graphics` per effettuare il **rendering** di figure 2D.
- Abbiamo visto che le primitive geometriche di `Graphics` permettono di specificare le diverse forme geometriche utilizzando delle coordinate intere.
- `Graphics` prevedeva inoltre un ristretto numero di primitive grafiche.
- Come evoluzione di `Graphics` nelle API di Java è stata inserita anche la classe astratta `java.awt.Graphics2D` che **estende** la classe astratta `java.awt.Graphics` introducendo molte funzionalità grafiche di grande utilità.

Da Graphics a Graphics2D

- Esattamente come avveniva per gli oggetti di tipo **Graphics**, anche un oggetto di tipo **Graphics2D** non può essere ottenuto mediante una istanziazione diretta con **new**.
- Un oggetto di tipo **Graphics2D** è ottenibile da un oggetto di tipo **Graphics** effettuando un semplice **casting esplicito** (conversione di tipo esplicita).

casting esplicito

```
public void paint(Graphics g) {  
    . . .  
    Graphics2D g2d = (Graphics2D)g;  
    . . .  
}
```



Primitive geometriche di Graphics2D

- Le primitive grafiche di **Graphics2D** per tracciare il **contorno** e per il **riempimento** di **forme geometriche 2D** sono riconducibili a due metodi pubblici aventi il seguente prototipo:

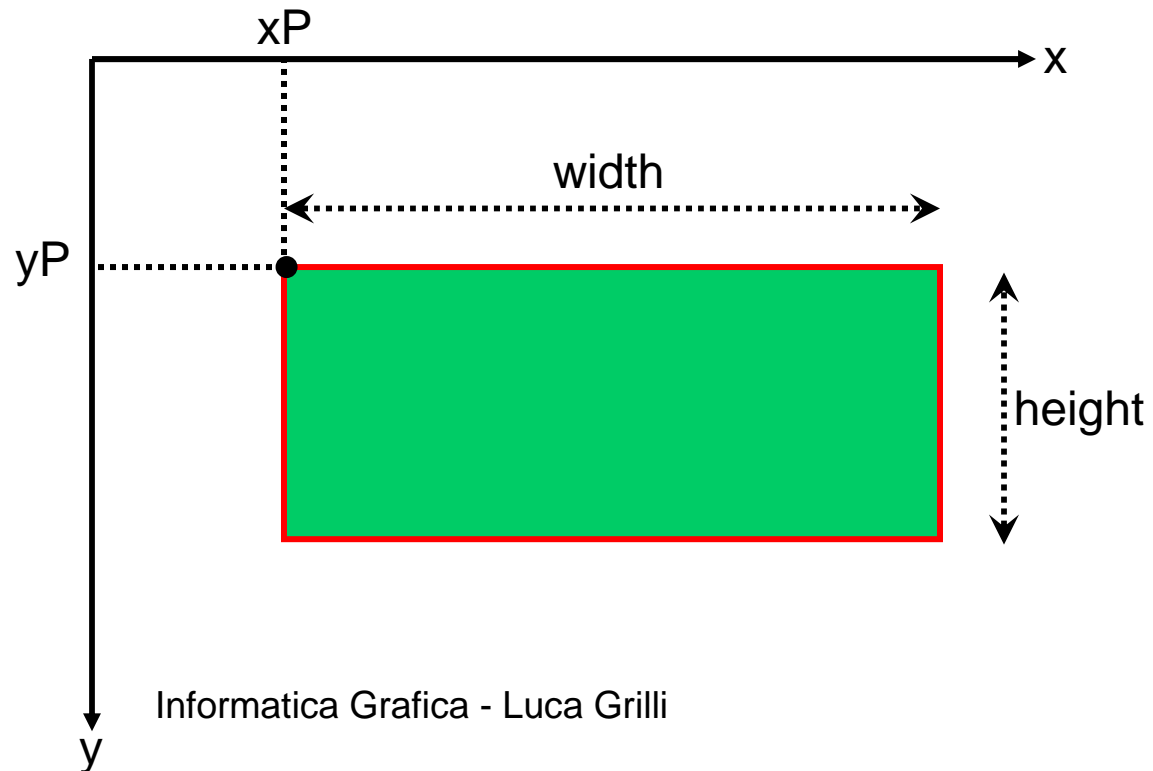
```
public abstract void draw(Shape s);  
public abstract void fill(Shape s);
```

- Dove **s** rappresenta il riferimento ad un qualsiasi oggetto di una classe che implementa l'interfaccia **java.awt.Shape**.
- In altre parole, invocando su un oggetto di tipo **Graphics2D** il metodo **draw()** e passandogli come argomento un oggetto **obj** di una classe che implementa l'interfaccia **Shape** \Rightarrow viene tracciato il **contorno** di dell'oggetto **obj**.
- Il metodo **fill()** si comporta in modo analogo, ma anziché tracciare il contorno effettua il **riempimento** della forma.

Esempio: disegno di un rettangolo

- Supponiamo di voler visualizzare il rettangolo riportato in figura, espresso in coordinate di tipo `double`, invocando i metodi `draw()` e `fill()` su un oggetto di tipo `Graphics2D`.

`xP = 59.376`
`yP = 123.553`
`width = 190.824`
`height = 137.432`



Esempio: disegno di un rettangolo

- I metodi `fill()` e `draw()` di `Graphics2D` richiedono in input un oggetto di una classe che implementa l'interfaccia `Shape`.
- In linea di principio è necessario definire una nostra classe `Rettangolo` che implementa l'interfaccia `Shape`, creare un oggetto di tale classe e passarlo ai metodi `fill()` e `draw()`.
- Comunque nelle API di Java, nel pacchetto `java.awt.geom`, sono presenti molte classi di frequente utilizzo che rappresentano delle forme geometriche 2D e che implementano l'interfaccia `Shape`.
- Ad esempio per la visualizzazione di un rettangolo sono state inserite le seguenti classi concrete (non astratte):
 - `Rectangle`: utilizza coordinate intere (tipo `int`);
 - `Rectangle2D.Float`: utilizza coordinate in virgola mobile (tipo `float`);
 - `Rectangle2D.Double`: utilizza coordinate in virgola mobile con doppia precisione (tipo `double`).

Esempio: disegno di un rettangolo

- Pertanto possiamo utilizzare la classe `Rectangle2D.Double` per creare un oggetto rettangolo di tipo `Shape`.
- Sotto viene mostrato come sovrascrivere il metodo `paint()` di `JFrame`.
- Chiaramente si mantengono valide tutte le osservazioni fatte nella precedente lezione sulla separazione della visualizzazione dalla descrizione della geometria del disegno.

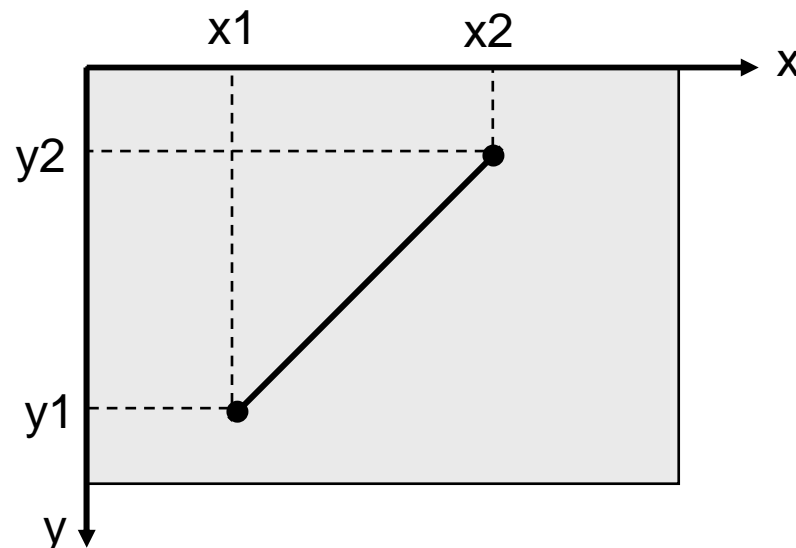
```
public void paint(Graphics g) {  
    super.paint(g);  
    Graphics2D g2d = (Graphics2D)g;  
    Rectangle2D.Double rect;  
    rect = new Rectangle2D.Double(57.376, 123.553, 190.824, 137.432);  
    g2d.setColor(Color.green);  
    g2d.fill(rect);  
    g2d.setColor(Color.red);  
    g2d.draw(rect);  
}
```

Classi concrete di tipo Shape

- Nelle prossime slide analizzeremo le seguenti **classi concrete** del pacchetto **java.awt.geom** che implementano l'interfaccia **Shape**:
 - Line2D.Float Line2D.Double
 - Path2D.Float Path2D.Double
 - Rectangle Rectangle2D.Float Rectangle2D.Double
 - Ellipse2D.Float Ellipse2D.Double
 - Per le classi
 - RoundRectangle2D.Float RoundRectangle2D.Double
 - Arc2D.Float Arc2D.Double
 - Polygon
- si rimanda alla documentazione Java.

Line2D.Double e Line2D.Float

- Tali classi estendono la classe astratta **Line2D** e implementano l'interfaccia **Shape**.
- Rappresentano un **segmento di linea** specificato rispettivamente con coordinate **double** e **float**.
- Nelle prossime slide illustreremo i campi e i metodi di **Line2D.Double**, i campi e i metodi di **Line2D.Float** sono definiti in modo simile.



La classe Line2D.Double

Campi della classe

double	<u>x1</u> // Coordinata X del primo punto del segmento.
double	<u>x2</u> // Coordinata X del secondo punto del segmento.
double	<u>y1</u> // Coordinata Y del primo punto del segmento.
double	<u>y2</u> // Coordinata Y del secondo punto del segmento.

Metodi costruttori

<u>Line2D.Double</u> () // Costruisce un segmento degenero con gli estremi di coordinate $P1 = P2 = (0,0)$.
<u>Line2D.Double</u> (double x1, double y1, double x2, double y2) // Costruisce un segmento con gli estremi di coordinate $P1 = (x1, y1)$ e $P2 = (x2, y2)$.
<u>Line2D.Double</u> (<u>Point2D</u> p1, <u>Point2D</u> p2) // Costruisce un segmento con gli estremi rappresentati da due oggetti della classe Point2D.

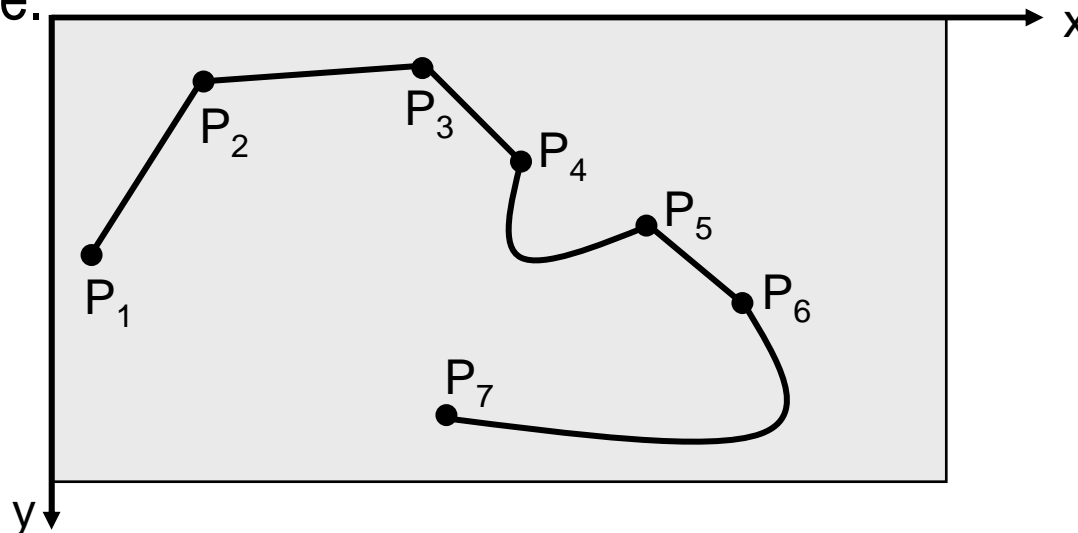
La classe Line2D.Double

Metodi pubblici

<u>Rectangle2D</u>	<u>getBounds2D</u> () <i>// Restituisce il bounding box di tipo Rectangle2D che contiene il segmento di linea.</i>
<u>Point2D</u>	<u>getP1</u> () <i>// Restituisce il primo Point2D del segmento.</i>
<u>Point2D</u>	<u>getP2</u> () <i>// Restituisce il secondo Point2D del segmento.</i>
double	<u>getX1</u> () <i>// Restituisce la coordinata X del primo punto del segmento.</i>
double	<u>getX2</u> () <i>// Restituisce la coordinata X del secondo punto del segmento.</i>
double	<u>getY1</u> () <i>// Restituisce la coordinata Y del primo punto del segmento.</i>
double	<u>getY2</u> () <i>// Restituisce la coordinata Y del secondo punto del segmento.</i>
void	<u>setLine</u> (double x1, double y1, double x2, double y2) <i>// Imposta i punti P1 = (x1, y1) e P2 = (x2, y2) come end points del segmento.</i>

Path2D.Double e Path2D.Float

- Tali classi estendono la classe astratta **Path2D** e implementano l'interfaccia **Shape**.
- Rappresentano un spezzata mista cioè costituita da segmenti rettilinei e curvilinei; l'**ultimo punto** del path è detto anche il **punto corrente**.
- Tale classe permette di attaccare al punto corrente del path un segmento rettilineo o curvilineo con dei metodi appositi.
- Nelle prossime slide illustreremo i campi e i metodi di **Path2D.Double**, i campi e i metodi di **Path2D.Float** sono definiti in modo simile.



La classe Path2D.Double

Metodi costruttori

Path2D.Double() // Costruisce un nuovo path vuoto con coordinate di tipo double.

Path2D.Double(Shape s) // Costruisce un nuovo path, espresso in coordinate double, dall'oggetto di tipo Shape.

Metodi pubblici

void moveTo(**double** x, **double** y) // Aggiunge un punto al path espresso in coordinate double. Tale punto diventa l'ultimo punto (o il punto corrente) del path.

void lineTo(**double** x, **double** y) // Aggiunge un punto al path disegnando un segmento rettilineo dal punto corrente del path al nuovo punto specificato in coordinate double.

void quadTo(**double** x1, **double** y1, **double** x2, **double** y2)
// Aggiunge un segmento curvilineo, definito da due nuovi punti, al path disegnando una curva quadratica che interseca il punto corrente del path e il punto di coordinate (x2, y2), usando il punto (x1, y1) come un punto di controllo quadratico (Bézier).

La classe Path2D.Double

Atri metodi pubblici

void	<u>curveTo</u> (double x1, double y1, double x2, double y2, double x3, double y3) <i>// Aggiunge un segmento curvilineo, definito da tre nuovi punti, al path disegnando una curva di Bézier che interseca il punto corrente del path e il punto di coordinate (x3, y3), usando i punti (x1, y1) e (x2, y2) come punti di controllo di Bézier.</i>
void	<u>closePath</u> () <i>// Chiude il “subpath” disegnando un segmento rettilineo che congiunge il punto corrente con l’ultimo punto inserito con il metodo moveTo(). Se il path è già stato chiuso non produce alcun effetto.</i>

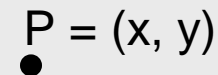
Il metodo moveTo()

- Il metodo **moveTo()** definisce il **punto iniziale** di un path (cammino) se il path è vuoto. Altrimenti, inizia un nuovo **subpath** (sotto-cammino).

PATH VUOTO

moveTo(x, y)

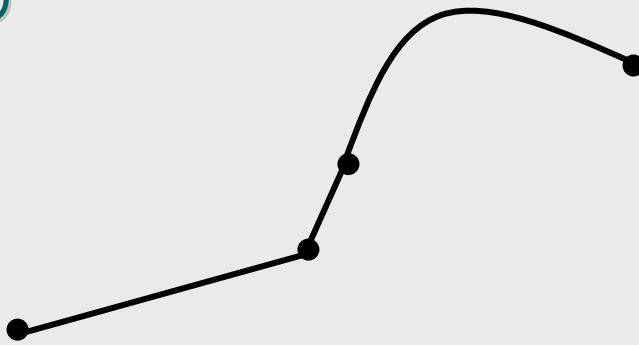
$P = (x, y)$

A single black dot representing the starting point P = (x, y) of an empty path.

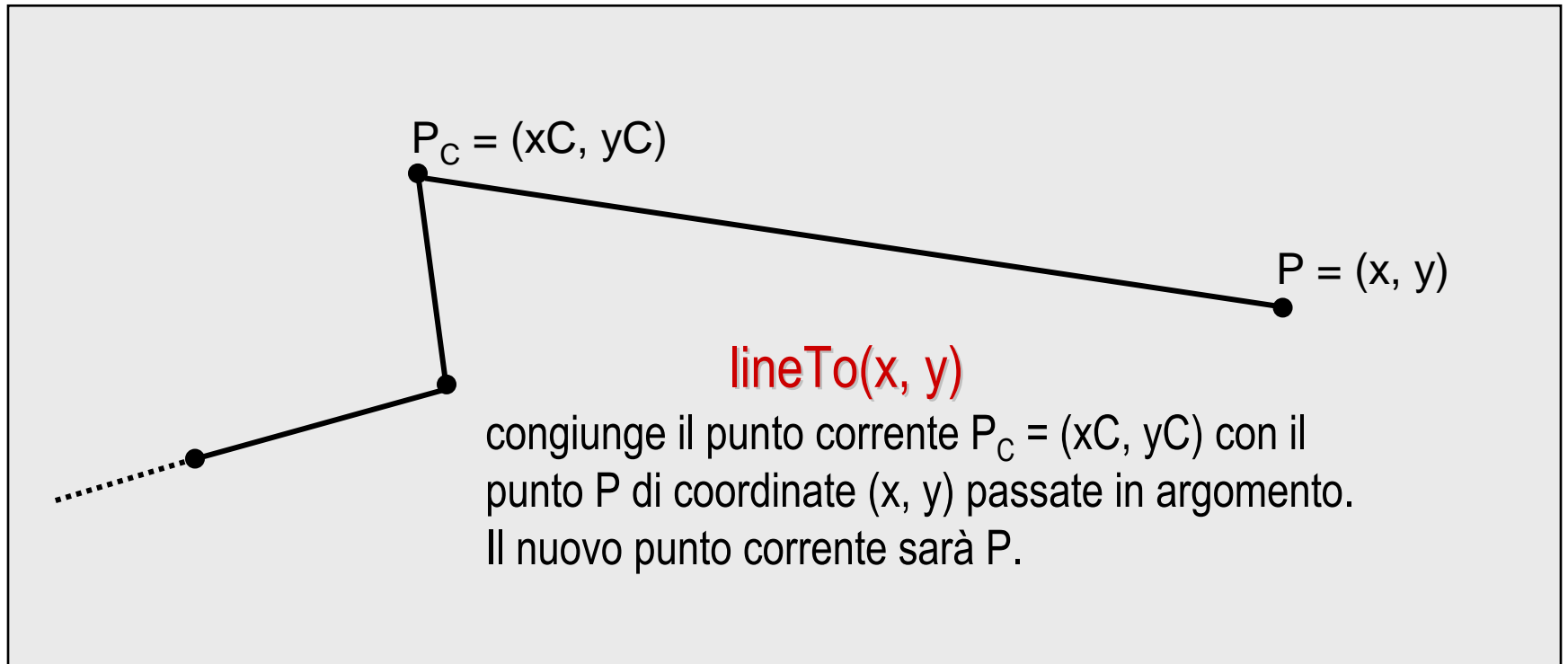
PATH NON VUOTO

moveTo(x, y)

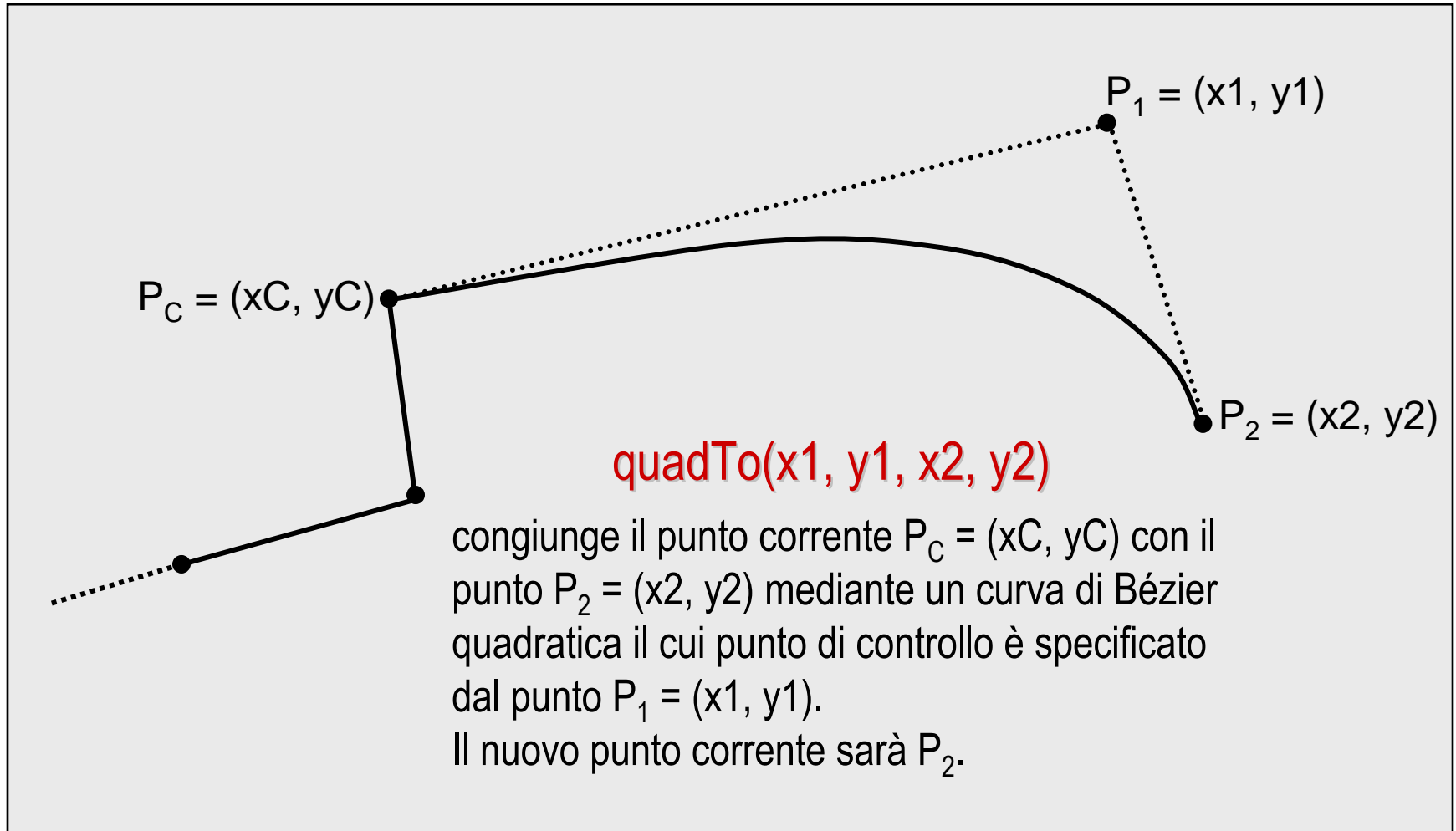
$P = (x, y)$

A path consisting of three connected segments: a straight line, a curved line, and another straight line. A new point P = (x, y) is shown to the right, representing the start of a new subpath.

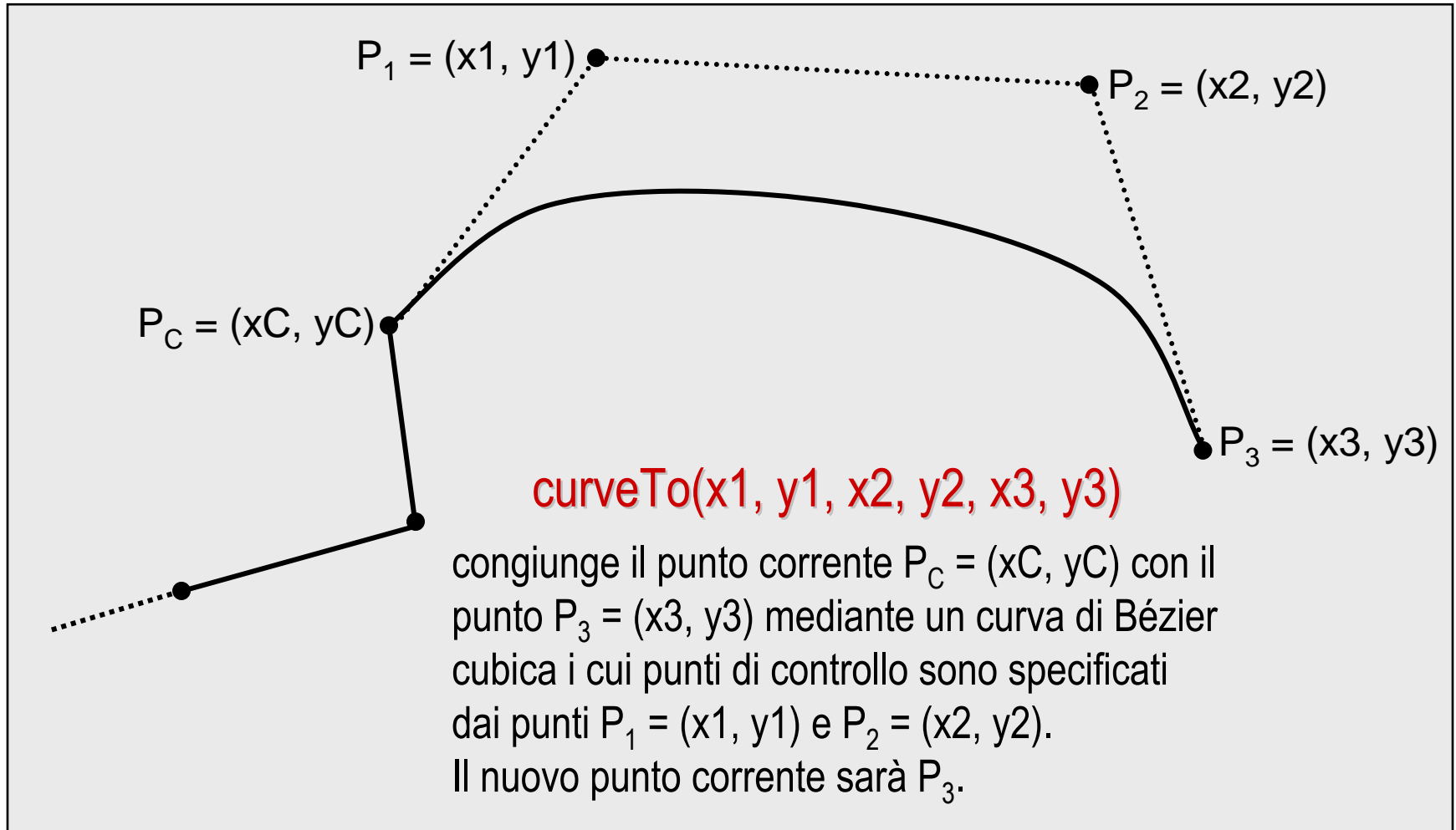
Il metodo lineTo()



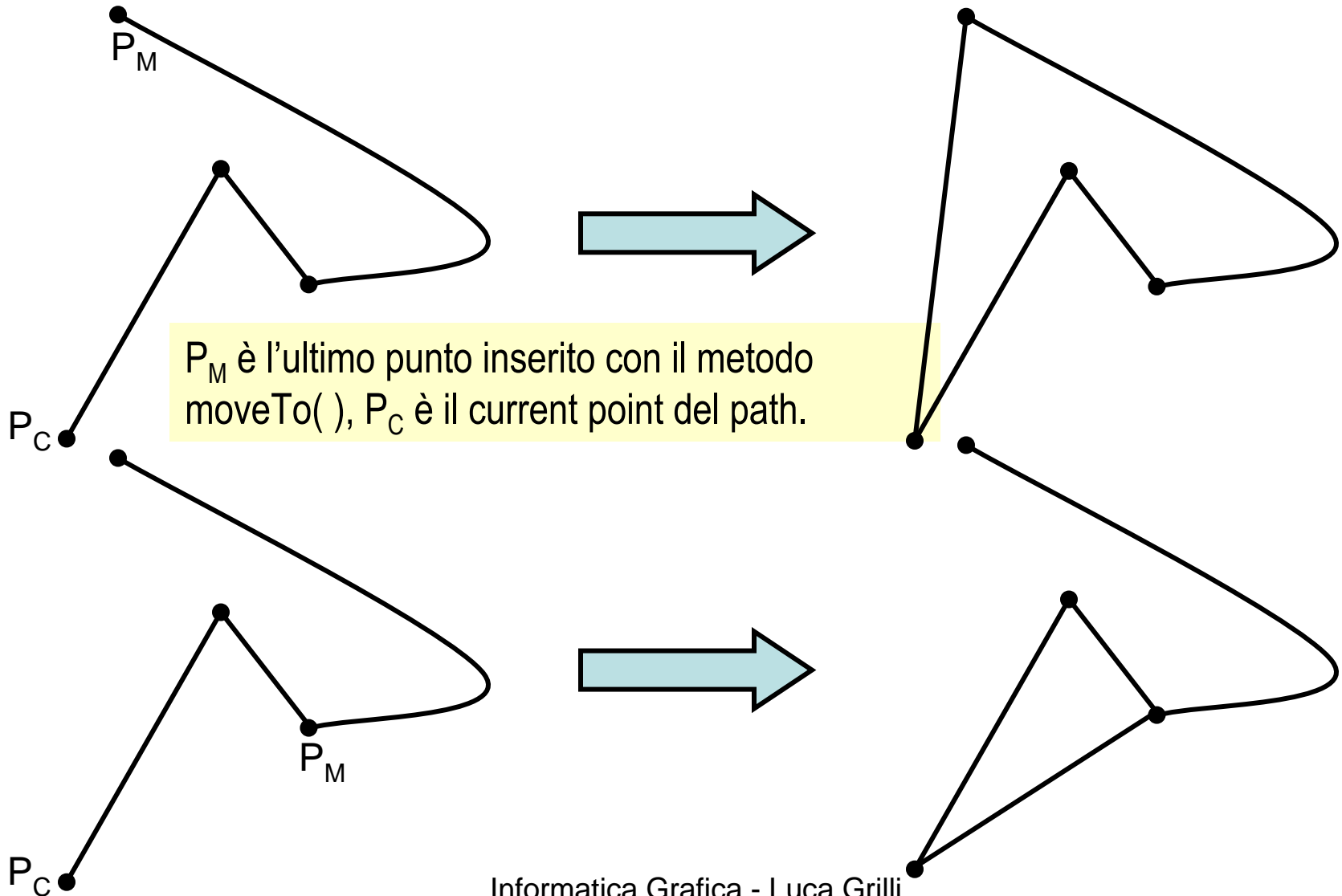
Il metodo quadTo()



Il metodo curveTo()



Il metodo closePath()



Classi di tipo Rettangolo

- Le classi **Rectangle**, **Rectangle2D.Float** e **Rectangle2D.Double**, del pacchetto **java.awt.geom**, estendono la classe astratta **Rectangle2D** e implementano l'interfaccia **Shape**.
- Permettono di rappresentare dei rettangoli espressi rispettivamente in **coordinate intere** (**int**), in **virgola mobile** (**float**) e in **doppia precisione** (**double**).
- Nelle prossime slide illustreremo i campi e i metodi della classe **Rectangle2D.Double**. I campi e i metodi di **Rectangle** e **Rectangle2D.Float** sono definiti in modo simile.

La classe Rectangle2D.Double

Campi della classe

double	<u>height</u> // Altezza del rettangolo.
double	<u>width</u> // Larghezza del rettangolo.
double	<u>x</u> // Coordinata X del vertice in alto a sinistra.
double	<u>y</u> // Coordinata Y del vertice in alto a sinistra.

Metodi costruttori

	<u>Rectangle2D.Double</u> () // Costruisce un nuovo rettangolo con il vertice in alto a sinistra in posizione (0,0) e avente altezza e larghezza nulle.
	<u>Rectangle2D.Double</u> (double x, double y, double w, double h) // Costruisce ed inizializza il rettangolo utilizzando le coordinate double specificate.

La classe Rectangle2D.Double

Metodi pubblici di uso comune

double	<u>getHeight()</u> <i>// Ritorna l'altezza del rettangolo.</i>
double	<u>getWidth()</u> <i>// Ritorna la larghezza del rettangolo.</i>
double	<u>getX()</u> <i>// Ritorna l'ascissa del vertice in alto a sinistra.</i>
double	<u>getY()</u> <i>// Ritorna l'ordinata del vertice in alto a sinistra.</i>
void	<u>setRect(double x, double y, double w, double h)</u> <i>// Imposta la posizione e le dimensione del rettangolo utilizzando i valori specificati.</i>
void	<u>setRect(Rectangle2D r)</u> <i>// Imposta la posizione e le dimensione del rettangolo in modo da ottenere un rettangolo analogo a quello passato in argomento.</i>

Classi di tipo Ellisse

- Le classi `Ellipse2D.Float` e `Ellipse2D.Double`, del pacchetto `java.awt.geom`, estendono la classe astratta `Ellipse2D` e implementano l'interfaccia `Shape`.
- Permettono di rappresentare delle ellissi espresse rispettivamente in **virgola mobile** (`float`) e in **doppia precisione** (`double`).
- Nelle prossime slide illustreremo i campi e i metodi della classe `Ellipse2D.Double`. I campi e i metodi di `Ellipse2D.Float` sono definiti in modo simile.

La classe `Ellipse2D.Double`

Campi della classe

<code>double</code>	<u>height</u> // Altezza totale dell'ellisse, cioè della bounding box che la contiene.
<code>double</code>	<u>width</u> // Larghezza totale dell'ellisse.
<code>double</code>	<u>x</u> // Coordinata X del vertice in alto a sinistra della bounding box dell'ellisse.
<code>double</code>	<u>y</u> // Coordinata Y del vertice in alto a sinistra della bounding box dell'ellisse.

Metodi costruttori

	<u>Ellipse2D.Double</u> () // Costruisce una nuova ellisse con il vertice in alto a sinistra della bounding box in posizione (0,0) e avente altezza e larghezza nulle.
	<u>Ellipse2D.Double</u> (<code>double x</code> , <code>double y</code> , <code>double w</code> , <code>double h</code>) // Costruisce ed inizializza l'ellisse utilizzando le coordinate double specificate.

La classe Ellipse2D.Double

Metodi pubblici di uso comune

double	<u>getHeight</u> () // Ritorna l'altezza della bounding box dell'ellisse.
double	<u>getWidth</u> () // Ritorna la larghezza della bounding box dell'ellisse.
double	<u>getX</u> () // Ritorna l'ascissa del vertice in alto a sinistra della bounding box dell'ellisse.
double	<u>getY</u> () // Ritorna l'ordinata del vertice in alto a sinistra della bounding box dell'ellisse.
void	<u>setFrame</u> (double x, double y, double w, double h) // Imposta la posizione e le dimensione della bounding box dell'ellisse utilizzando i valori specificati.