

DOCUMENTAZIONE CODICE (Berzoini Borsetti Cestari)

```
Elettroscopio  Bachelite  Dito  Lamine  Vetro  ▼
1 //dichiarazioni
2 PImage img;
3
4 Bachelite bac;
5 Vetro vet;
6 Dito dito;
7 Lamine oro;
8
9 String testo1, testo2, testo3;
10 boolean touchB = false, touchV = false, touchD = false;
11 boolean nearB = false, nearV = false;
12 int c=0, k=0, j=0;
13 float pposX=0;
14 float pposY=0;
15
16 void setup () {
17   size(1100, 700);
18   start();
19   images();
20 }
```

SKETCH PRINCIPALE

Il codice è costituito da uno sketch principale, denominato “Elettroscopio” e 4 classi, ognuna per ciascuno degli oggetti con cui si può interagire nella schermata.

Partiamo dallo sketch “elettroscopio”: troviamo subito le dichiarazioni delle immagini, variabili e classi varie.

Nel **setup** viene impostata la dimensione predefinita dello schermo e vengono chiamate due funzioni:

- **start()** per aggiungere gli elementi delle classi alla schermata
- **images()** per aggiungere le immagini alla schermata

```
void start(){
  bac = new Bachelite();
  vet = new Vetro();
  dito = new Dito();
  oro = new Lamine();
}
```

```
void images(){
  img = loadImage("elettroscopio.png");
}
```


```
22 void draw () {
23   background(255, 254, 217);
24   resize();
25
26   title();
27
28   //tavolino
29   stroke(0);
30   strokeWeight(1);
31   fill(161, 161, 161);
32   rect(100, 660, 500, 50);
33
34   display();
```

Nel **draw** viene

impostato lo sfondo e le dimensioni delle immagini vengono modificate tramite **resize()**.

La funzione **title** imposta posizione e dimensioni del testo “elettroscopio”, posto al centro della schermata in alto quando viene eseguito il programma.

Viene disegnato poi un rettangolo che rappresenta il tavolo di lavoro su cui poggia l’elettroscopio e successivamente viene chiamata la funzione **display()** per ogni singolo oggetto, cui vengono assegnate delle coordinate iniziali, che durante l’esecuzione del programma cambieranno in continuazione.



```
void display(){
  bac.display();
  vet.display();
  oro.display();
  dito.display();
}
```

```
//bachelite
nearB = bac.xB>=395 && bac.xB<=510 && bac.yB>=28 && bac.yB<=328;
touchB = bac.xB>330 && bac.xB<395 && bac.yB>120 && bac.yB<225;

//vetro
nearV = vet.xV>=395 && vet.xV<=510 && vet.yV>=28 && vet.yV<=328;
touchV = vet.xV>330 && vet.xV<395 && vet.yV>120 && vet.yV<225;

//dito
touchD = dito.xD>330 && dito.xD<395 && dito.yD>120 && dito.yD<225;
```

Alle variabili booleane **nearB**, **touchB** ecc vengono assegnati particolari intervalli di coordinate sullo schermo che serviranno rispettivamente per l’induzione e la conduzione.

Seguono poi una serie di condizioni che permettono di distinguere induzione e conduzione grazie al contatore c.

```
//CONTATTO E INDUZIONE
//contatto
if (touchB || touchV) {
    oro.open(); //se le bacchette stanno toccando le lamine si aprono
    c++;        //serve nell'induzione quando allontanano la bacchetta -> le lamine non si chiudono
}
```

- Se le bacchette si trovano in prossimità del pomello (nearB per la bachelite e nearV per il vetro) ma non lo stanno toccando, le lamine si allargano progressivamente all'aumentare della vicinanza della bacchetta al pomello.
- Se le bacchette si trovano in prossimità del pomello e il mouse le sta trascinando da sinistra verso destra (bac.xB>pmouseX per la bachelite, stessa cosa per il vetro) vuol dire che sto tornando dall'induzione (c==0), quindi le lamine si riavvicinano progressivamente

```
//induzione con allargamento delle lamine progressivo
if ((nearB && !touchB && mousePressed && mouseX==bac.xB && mouseY==bac.yB && bac.xB<pmouseX) || (nearV && !touchV && mousePressed && mouseX==bac.xB && mouseY==bac.yB && bac.xB<pmouseX)) {
    oro.aumenta();
}
if (c==0 && nearB && mousePressed && mouseX==bac.xB && mouseY==bac.yB && bac.xB>pmouseX || (c==0 && nearV && mousePressed && mouseX==bac.xB && mouseY==bac.yB && bac.xB>pmouseX)) {
    oro.diminuisce();
}
if (((c==0 && mousePressed && mouseX==bac.xB && mouseY==bac.yB && bac.xB>pmouseX) && mouseX>715) || ((c==0 && nearV && mousePressed && mouseX==bac.xB && mouseY==bac.yB && bac.xB>pmouseX) && mouseX>715)) {
    oro.closed();
}
```

Se poi tocco il pomello con il dito (touchD) la carica delle lamine si scarica a terra (in ambito fisico) per cui le lamine si chiudono.

- Se poi le lamine erano già aperte, in seguito al contatto con una delle bacchette, il valore del contatore torna a 0.
- Se, invece, oltre a toccare con il dito il pomello, una delle due bacchette sta inducendo, il valore del contatore k viene incrementato

```
if (touchD){
    oro.closed(); //Se tocco con il dito le lamine si chiudono.
    if (oro.carico){ //Se prima erano aperte per contatto,
        c=0;        //il contatore c torna a zero ->vedi diminuzione progressiva lamine.
    }
    else if (nearB || nearV){ //Se oltre al dito una delle bacchette è in induzione,
        k++;                //il contatore k aumenta.
    }
}
```

Se sto ancora inducendo con le bacchette in vicinanza del pomello e tocco il pomello con il dito

- se allontanano prima il dito, le lamine si riaprono poiché la bacchetta sta ancora inducendo ed è ancora carica, quindi le lamine tornano ad essere cariche. Se poi, dopo aver tolto il dito tolgo anche la bacchetta le lamine si richiudono progressivamente
- se allontanano prima la bacchetta le lamine restano chiuse perchè vengono scaricate del tutto, ma se poi tolgo il dito le lamine si riaprono

```
//se tolgo il dito dopo aver fatto induzione (k>1), le lamine si aprono
if (k>=1 && c==0 && touchD==false && (nearB==false || nearV==false)) {
  oro.open();
  k=0;
}
```

L'ultima funzione che troviamo nello sketch principale è **mouseDragged()** con la quale è possibile spostare gli oggetti trascinandoli con il mouse: infatti se viene premuto il mouse quando esso si trova sopra uno dei tre strumenti (bachelite, vetro o dito), questi ultimi prendono le coordinate del mouse.

```
void mouseDragged(){
  if(mouseButton == LEFT){//bachelite e vetro
    if(mouseX >= bac.xB - 40 && mouseX <= bac.xB+240 && mouseY >= bac.yB - 40 && mouseY <= bac.yB+250){
      bac.xB = mouseX;
      bac.yB = mouseY;
    }else if(mouseX >= vet.xV - 40 && mouseX <= vet.xV+240 && mouseY >= vet.yV - 40 && mouseY <= vet.yV+250){
      vet.xV = mouseX - 40;
      vet.yV = mouseY - 20;
    }
  }else if (mouseButton == RIGHT){ //dito
    dito.xD = mouseX - 40;
    dito.yD = mouseY - 20;
  }
}
```

CLASSE BACHELITE, DITO E VETRO

La classe “bachelite” è una classe pubblica come le variabili dichiarate al suo interno in

```
1 public class Bachelite{
2
3     //dichiarazione
4     public int xB, yB;
5     public PImage bachelite;
6
7     //dati
8     Bachelite(){
9         xB = width - width/4;
10        yB = height/8;
11        bachelite = loadImage("bachelite.png");
12    }
```

modo che si possa accedere ad esse anche dallo sketch principale.

Viene caricata poi l'immagine dello strumento e con la funzione **display()**, che viene chiamata nello sketch “elettroscopio”, vengono assegnate particolari coordinate allo strumento.

```
//disposizione
void display(){
    bachelite.resize(240, 250);
    image(bachelite, xB, yB);
}
```

Le classi “dito” e “vetro” sono analoghe alla classe bachelite, l'unica differenza sono le coordinate che vengono assegnate agli oggetti nel costruttore poiché variano per ogni strumento.

CLASSE LAMINE

Infine, nel nostro programma troviamo la classe “lamine”.

```
1 public class Lamine{
2     //dichiarazione
3     public float x1, y1, x2D, x2S, y2D, y2S;
4     public float n;
5     public boolean carico = false;
6     //x1 e y1 delle due linee in comune, la seconda è diversa per dx e sx
7
8     //dati
9     Lamine(){ //inizialmente sono chiuse
10        n = 3;
11
12        x1 = 362;
13        y1 = 571;
14        x2D = x1 + n;
15        x2S = x1 - n;
16        y2D = y2S = 650;
17    }
```

E' presente la dichiarazione delle variabili che riguardano le lamine d'oro, rappresentate come due linee con una coppia di coordinate in comune, mentre la seconda coppia è diversa per lamina destra e sinistra.

- **display()**

```
//disposizione
void display(){
    //lamina dx
    stroke(255, 226, 38);
    strokeWeight(7);
    line(x1, y1, x2D, y2D); //x1, y1, x2, y2

    //lamina sx
    stroke(255, 226, 38);
    strokeWeight(7);
    line(x1, y1, x2S, y2S);
}
```

La funzione viene chiamata nello sketch principale e permette di disegnare e visualizzare sullo schermo le due linee, le quali rappresentano i foglietti d'oro che fanno parte dell'elettroscopio.

- **open() e closed()**

```
void open(){
    x2D = x1 + 36;
    x2S = x1 - 36;
    y2D = y2S = 641;

    carico = true; //le lamine sono aperte e cariche
}

void closed(){ //posizione chiusa e iniziale
    y2D = y2S = 650;
    x2D = x1 + 3;
    x2S = x1 - 3;

    carico = false;
}
```

La funzione **open()** viene chiamata nel caso in cui si stia facendo conduzione con lo strumento (bacchetta di vetro o bachelite) e quindi permette di mantenere le lamine aperte assegnando particolari coordinate alle due linee. La variabile booleana carico è quindi vera.

La funzione **closed()** invece viene chiamata una volta finita l'induzione poiché se la bacchetta è troppo lontana le lamine non sono soggette a nessuna carica per cui carico è false.

Quest'ultima funzione viene anche chiamata nel caso in cui si voglia scaricare l'elettroscopio dopo aver fatto conduzione con la bacchetta.

- **aumenta() e diminuisce()**

```
void aumenta(){ //aumenta l'apertura delle lamine
    y2D = y2D - 0.02;
    y2S = y2S - 0.02;

    n = n + 0.5;
    x2D = x1 + n;
    x2S = x1 - n;
}

void diminuisce(){//diminuisce l'apertura delle lamine
    y2D = y2D + 0.02;
    y2S = y2S + 0.02;

    n = n - 0.7;
    x2D = x1 + n;
    x2S = x1 - n;
}
```

Infine troviamo le due funzioni **aumenta()** e **diminuisce()** le quali, quando vengono chiamate, aumentano e diminuiscono progressivamente l'apertura delle lamine, modificando ripetutamente la seconda coppia di coordinate di ogni linea (lamina).