Marco Barioni

Giorgio Bolognesi

Fabio Ganzaroli

Scheda Tecnica Programma Rifrazione

Per una maggior chiarezza il programma è stato suddiviso in quattro sketch, una di queste rappresenta una Classe vera e propria.

- ***RIFRAZIONE***

È il primo sketch, nonché quello principale.

All'inizio possiamo trovare la dichiarazione di tutte le variabili che verranno poi usate

```
float m;//coefficiente angolare
float x, y;
float tx, ty;//coordinate linea perpendicolare centrale
float n1, n2;// indici di rifrazione
float AB, BC, AC, Rangle, Dangle, Nangle, Langle;//variabili usate per calcolo angolo e nuovo rag
boolean check = true; //Variabile utilizzata per gestione Menù
Pulsante V1, W1, B1, O1, D1, A1, V2, W2, B2, O2, D2, A2; //Variabili per I Pulsanti


PFont font;// variabile contenente font scritte

int screen=0; //Variabile che controlla le varie schermate

```

Successivamente nel setup troviamo le classiche impostazioni per quanto riguarda la finestra e l'inizializzazione dei Pulsanti creati tramite la loro apposita classe e poi usati per cambiare materiale all'interno del programma.

```
void setup() {
  size(1280, 720);
  background(255);
  y=400;
  x=0;
  tx=width/2;
  frameRate(60);

  //Caricamento del font e della dimensione del testo
  font = loadFont("RussoOne.vlw");
  textFont(font, 20);
  //Creazione pulsanti
  V1 = new Pulsante(50,80,50,50,loadImage("vetro.png"));
  W1 = new Pulsante(110,80,50,50,loadImage("acqua.png"));
  B1 = new Pulsante(170,80,50,50,loadImage("bromo.png"));
  O1 = new Pulsante(230,80,50,50,loadImage("olio.png"));
  D1 = new Pulsante(290,80,50,50,loadImage("diamante.png"));
  A1 = new Pulsante(350,80,50,50,loadImage("aria.png"));
  V2 = new Pulsante(50,480, 50, 50,loadImage("vetro.png"));
  W2 = new Pulsante(110,480, 50, 50,loadImage("acqua.png"));
  B2 = new Pulsante(170,480,50,50,loadImage("bromo.png"));
  O2 = new Pulsante(230,480,50,50,loadImage("olio.png"));
  D2 = new Pulsante(290,480,50,50,loadImage("diamante.png"));
  A2 = new Pulsante(350,480,50,50,loadImage("aria.png"));
```

Ogni pulsante ha le sue coordinate ed è collegato a un determinato materiale, 6 pulsanti saranno per il primo mezzo e 6 per il secondo.

```
void mousePressed() {
  if (V1.clicked()) {  //controllo stato pressione del tasto
    V1.cliccato = !V1.cliccato;//se gia cliccato, ricliccandolo torna false, quind
  }
```

La funzione ***mousePressed()*** serve per capire se , quando si preme un pulsante, esso è già premuto oppure no, nel caso in cui sia premuto cliccandolo nuovamente lo fa tornare false, quindi non premuto.

```
if (V1.cliccato) { /
  V1.cliccato=true;
  W1.cliccato=false;
  B1.cliccato=false;
  O1.cliccato=false;
  D1.cliccato=false;
  A1.cliccato=false;
  n1 = 1.52;
}
```

All’interno della stesa funzione troviamo anche un controllo che impedisce a due

tasti di essere premuti contemporaneamente.

- ***PULSANTE***

```
class Pulsante{ //classe per l'uso dei pulsanti per cambiare materiale
  float x,y;
  float w,h;//width e heigth
  float cr;//angolo del bordo
  boolean cliccato;//variabile che controlla lo stato di pressione del Pulsante
  PImage img;

  public Pulsante(float x, float y, float w, float h, PImage img){ //costruttore
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h; //assegno variabile alla classe
    this.cr = 8.0;
    this.img = img;
  }

  void disegna(){  //funzione per il disegno dei pulsanti
    if(cliccato) fill(120);// se è cliccato lo sfondo avrà un colore piu scuro per far capire che il tasto è premuto
    else fill(200);//altrimenti il colore sarà più chiaro
    rectMode(CENTER);//impostazione modalità disegno rettangolo, center indica che le coordinate sono quelle del cent
    rect(x,y,w,h,cr);
    rectMode(CORNER);//corner indica invece le coordinate del vertice in alto a sinistra del rettangolo
    imageMode(CENTER);//impostazione modalità disegno immagine
    image(img,x,y,w*0.9,h*0.9);
    imageMode(CORNER);

  }

  boolean clicked(){ //controlla stato pressione del Pulsante
    return hovered() && mousePressed;
  }

  boolean hovered(){ //controlla sovrapposizione retta con Pulsante
    float mx=mouseX;
    float my=mouseY;
    return (mx>x-w/2 && mx<x+w/2) && (my>y-h/2 && my<y+h/2) ;
  }
}
```

Il secondo sketch è adibito alla classe Pulsante, qui è contenuta la porzione di codice necessaria per creare e far funzionare correttamente i Pulsanti.

All'interno della classe sono presenti i parametri float delle coordinate x e y, dell'altezza e larghezza del pulsante, dell'angolo del bordo del pulsante e infine una variabile booleana per controllarne lo stato.

Dopo aver inizializzato il costruttore, troviamo la funzione ***disegna()*** che ha il compito di disegnare il nostro pulsante.

RectMode e ***ImageMode*** impostano rispettivamente le modalità di disegno del rettangolo e dell' immagine. Le coordinate dell' immagine, altezza e larghezza, sono moltiplicate x0.9 per evitare di avere l'immagine attaccata ai bordi del Pulsante.

Le due funzioni ***clicked()*** e ***hovered()*** controllano rispettivamente la pressione del tasto e la sovrapposizione della retta col tasto, in modo da evitare pressioni dei tasti indesiderate.

- ## *BACKGROUND*

Nella scheda Background troviamo tutte le funzioni necessarie alla stampa dell'ambiente grafico. In base alla scelta fatta in ogni menù(grazie alle funzioni ***menu1()*** e ***menu2()***) si andrà poi ad assegnare un determinato valore all' indice di rifrazione del mezzo e di conseguenza grazie a ciò la funzione ***PrintSup()*** stamperà il vero e proprio ambiente di lavoro.
All' interno di questa scheda due variabili molto importanti sono ***Screen*** e soprattutto ***check***.
Screen ha funzione di contatore e permette di cambiare la schermata una volta selezionata una voce di un menù.
Check invece ci ha permesso di risolvere un problema che si presentava una volta passati da un menù all' altro.

A causa della funzione ***keyPressed*** infatti, riscontravamo problemi passando da un menù all' altro in quanto la funzione, pur premendo il tasto una sola volta, andava a riceve l'input di quel tasto più volte impedendoci la selezione del secondo materiale, che risultava cosi uguale al primo.
In un primo momento per evitare ciò avevamo cambiato i tasti per la selezione tra menù 1 e menù 2 poi abbiamo introdotto ***check***, che diventa falso dopo la prima selezione e vero solamente dopo esser passato in un *if* che lo rende vero, facendo ciò la selezione del secondo materiale è gestita da un *if* nel quale si entra solo se, oltre a premere il tasto desiderato, ***check*** è vero.
In questo modo abbiamo mantenuto gli stessi tasti nei due menù risolvendo il problema che ci si era presentato.

```
if (key == '3') {
  screen=1;
  n1=1.661;
  check = false;
}
```

```
if (keyPressed && check) {
  if (key == '1') {
    n2=1.52;
    screen=2;
  }
  if (key == '2') {
    n2=1.33;
    screen=2;
  }
  if (key == '3') {
    screen=2;
    n2=1.661;
  }
  if (key == '4') {
    screen=2;
    n2=1.46;
  }
  if (key == '5') {
    screen=2;
    n2=2.41;
  }
  if (key == '6'){
    screen=2;
    n2=1.0003;
  }
}
else if(keyPressed){ //se che
}
else{  //subito dopo check d
  check = true;
}
```

- ***CALCOLI***

Questo sketch potrebbe essere considerato il cuore del programma.

Qui sono contenute le funzioni per il calcolo e disegno dell'angolo incidente e riflesso e delle rette.

```
void seglength() {    //Funzione che determina la lunghezza dei vari segmenti necessari a determina
  AB=dist(640, mouseY, 640, 400);
  BC=dist(640, 400, mouseX, mouseY);
  AC=dist(mouseX, mouseY, 640, mouseY);

  println("ab=", AB);
  println("bc=", BC);
  println("ac=", AC);
}


void anglecalc() { //Funzione per determinare l' angolo e successivamente diseganre il suo arco

  Rangle= asin(AC/BC);
  Dangle=degrees(Rangle);
  println("angolo1=", Dangle);
  textFont(font, 30);
  text(Dangle, 340, 30);

  if (mouseX < 640 && mouseY < 400) {
    m = abs(tan(radians(270) + Rangle)); //coefficiente angolare retta riflessa
    fill(255, 255, 0);
    arc(640, 400, 100, 100, radians(270), radians(270) + Rangle, PIE); //arco angolo riflesso
    line(640, 400, 1280, 400-m*680); //retta che genera il raggio riflesso
    fill(255, 0, 0);
    arc(640, 400, 100, 100, radians(270) - Rangle, radians(270), PIE); //arco angolo incidente
    noFill();
  } else if (mouseX > 640 && mouseY < 400) {
    m = abs(tan(radians(270) - Rangle)); //coefficiente angolare retta riflessa
    fill(255, 255, 0);
    arc(640, 400, 100, 100, radians(270) - Rangle, radians(270), PIE); //arco angolo riflesso
    line(640, 400, 0, 400-m*680); //retta che genera il raggio riflesso
    fill(255, 0, 0);
    arc(640, 400, 100, 100, radians(270), radians(270) + Rangle, PIE); //arco angolo incidente
    noFill();
  }
}


void newangle() {  //Funzione per determinare l'angolo Rifratto
  Nangle =(asin((n1*sin(Rangle))/n2));
  println("angolo2=", degrees(Nangle));
  textFont(font, 30);
  fill(0);
  text(degrees(Nangle), 330, 430);
  noFill();
}
```

Seglength() calcola la lunghezza dei vari segmenti che vanno a costituire il triangolo rettangolo che si forma tra retta incidente, normale e piano.

Anglecalc() ci permette di determinare l' angolo sfruttando le formule trigonometriche e si occupa anche di stampare a schermo l'arco dell' angolo della retta incidente. Inoltre permette di disegnare il raggio riflesso e il suo arco a partire dal coefficiente angolare dell'angolo (l'arco sarà riempito con un colore diverso proprio per far capire che si tratta di un altro fenomeno ottico, quello della riflessione). La funzione *PIE* invece, disegna l'arco come se fosse una sezione di un diagramma a torta (è stata una scelta esclusivamente estetica,

poiché senza questa funzione, sarebbe stata oscurata la normale dal *FILL* dell'angolo).

Newangle() calcola il nuovo angolo rifratto usando la **legge di Snell** attraverso la formula ***AngoloRifratto=arcsin(n1*sin(AngoloIncidente)/n2)*** , in questo caso abbiamo aggiunto anche la conversione in gradi durante la successiva stampa a schermo poiché processing lavora con gli angoli in radianti.

```
void newretta() { //Funzione per disegnare il raggio rifratto
  m = tan(abs(Nangle-HALF_PI)); //usa coeff angolare(Tan angolo)

  if (mouseY < 400 & mouseX < 640) {
    fill(255, 0, 0);
    arc(640, 400, 100, 100, abs(Nangle-HALF_PI), radians(90), PIE);
    noFill();
    line(640, 400, 1280, 400+(m*640));
  } else if (mouseY < 400 & mouseX > 640 ) {
    fill(255, 0, 0);
    arc(640, 400, 100, 100, radians(90), abs(Nangle+HALF_PI), PIE);
    noFill();
    line(640, 400, 0, 400+(m*640));
  }
  noFill();
}

void angleLimit() { //Funzione per cercare l'esistenza dell'angolo limite
  Langle = asin(n2/n1);
  if (Rangle > Langle) {
    m= tan(HALF_PI-Rangle);
    println("angolo rifratto inesistente");
    textFont(font, 30);
    fill(255, 0, 0);
    text(" INESISTENTE", 330, 430);
    noFill();

    if (mouseY < 400 && mouseX < 640) {
      fill(255, 0, 0);
      arc(640, 400, 100, 100, radians(270), abs(radians(270)+Rangle), PIE);
      noFill();
      line(640, 400, 1280, 400-m*680);
    } else if (mouseY < 400 && mouseX > 640) {
      fill(255, 0, 0);
      arc(640, 400, 100, 100, abs(radians(270)-Rangle), radians(270), PIE);
      noFill();
      line(640, 400, 0, 400-m*680);
    }
  }
}
```

Newretta() permette la creazione della nuova retta rifratta e del nuovo arco dell'angolo rifratto. Poiché Processing considera gli angoli a partire da destra conoscendo le coordinate di un punto, sarà necessario riconvertire il coefficiente angolare, in modo tale che l'angolo rifratto sarà compreso tra la normale e la retta rifratta e non tra la linea di demarcazione delle superfici e la retta rifratta. Le nuove rette sono state quindi disegnate secondo la funzione matematica

della retta: y= mx +q.

In entrambi i casi (sia nel disegno dell'arco della retta incidente che nel disegno dell'arco e della retta rifratta) si è tenuto caso di ogni possibilità, modificando come l'arco deve essere disegnato in base al quadrante in cui si trova. Ciò è espresso negli if che distinguono i vari casi.

L'ultima funzione è ***angleLimit()***, che calcola l'angolo limite mediante la formula (***arcsin(n2/n1)***. Viene poi scritto in output un messaggio al posto dell'angolo in gradi che indica che l'angolo rifratto è inesistente e quindi vi è una riflessione totale. Analogamente alla funzione ***Newretta(),*** anche qui è stata effettuata la distinzione su come disegnare la retta e l'arco a seconda del quadrante nel quale si trova la retta incidente.