

ARRAY E MAPPE*

Davide Rocchesso

This work is produced by The Connexions Project and licensed under the Creative Commons Attribution License †

Abstract

Utilizzazione di array mono- e bi-dimensionali nella realizzazione di mappe per il controllo dell'interazione.

1 Array

Un array è una lista numerata di valori, tutti dello stesso tipo (`int`, `float`, ecc.). La numerazione, in Processing (e Java), parte da zero. La dichiarazione e creazione di un array avviene in maniera uguale a quella di un oggetto istanza di classe. Ad esempio

```
float[] vettore = new float[100];
println(vettore);
```

crea un array di 100 elementi di tipo `float` e ne stampa il contenuto, che è inizialmente costituito da zeri. L'array così dichiarato è a tutti gli effetti un oggetto istanza di classe, e tale classe ha anche una variabile `length`, che ne contiene la lunghezza. Quindi, per inizializzare l'array come una rampa lineare che va da 0 a 1, si può fare

```
for (int i = 0; i < vettore.length; i++)
    vettore[i] = (float)i/vettore.length;
println(vettore);
```

NOTE: Si provi a togliere il typecasting¹ (`float`) e si spieghi perché la rampa non viene più generata.

Example 1: 1-to-1 mapping

Un MIDI controller² continuo, come uno slider, può inviare valori tra 0 e 127, che possono essere usati come indici per leggere i valori di una certa proprietà. Un array può servire a mappare un controllo in un valore di proprietà. La relazione tra controllo (indice) e proprietà è una mappa uno a uno. Nel codice che segue, la mappa è costruita muovendo sopra alla finestra il mouse con tasto premuto. Invece, se il mouse viene spostato senza essere premuto, la posizione orizzontale del puntatore (tra 0 e 127) viene interpretata come indice della mappa, che restituisce il valore della proprietà di brillantezza dello sfondo.

*Version 1.10: Feb 6, 2008 7:17 am -0600

†<http://creativecommons.org/licenses/by/2.0/>

¹http://en.wikipedia.org/wiki/Type_conversion#Explicit_type_conversion

²http://en.wikipedia.org/wiki/Midi_controller

```

int ris = 128;
int[] midimap = new int[128];

void setup() {
  size(ris,2*ris);
}

void draw() {
  background(255);
  if ((mouseX >= 0) && (mouseX < ris))
    if (!mousePressed)
      background((256 - midimap[mouseX]));
    else
      midimap[mouseX] = mouseY;

  for (int i=0; i<midimap.length; i++) point(i, midimap[i]);
}

```

Come si possono evitare i "buchi" nella mappa dovuti alla lettura non continua delle posizioni del mouse?

Estensione di risoluzione

Ho un controllore MIDI (a 128 valori) e voglio scandire una tabella di 256 valori. Si può iterare la lettura avanzando il punto di lettura con passo 2.

```

int sup_ris = 256;
int inf_ris = 128;
float[] midimap = new float[sup_ris];

void setup() {
  size(inf_ris,inf_ris);
  colorMode(RGB, 1.0);
  for (int i = 0; i < midimap.length; i++) midimap[i] = sin(PI*(float)i/midimap.length);
}

void draw() {
  if ((mouseX >= 0) && (mouseX < inf_ris) && (mousePressed))
  {
    point(mouseX, inf_ris - inf_ris*midimap[sup_ris/inf_ris*mouseX]);
  }
}

```

Riduzione di risoluzione

Ho un controllore a 10 bit (1024 valori) e devo mapparlo in una proprietà secondo una tabella di 128 valori che descrivono un semiperiodo di seno.

```

int inf_ris = 128;
int sup_ris = 1024;
float[] midimap = new float[inf_ris];

void setup() {
  size(sup_ris,32);
  colorMode(RGB, 1.0);
  for (int i = 0; i < midimap.length; i++) midimap[i] = sin(PI*(float)i/midimap.length);
}

void draw() {
  if ((mouseX >= 0) && (mouseX < sup_ris) && (mousePressed))
    background(midimap[int(float(mouseX)/sup_ris*inf_ris)]);
}

```

Come posso accedere alla tabella sfruttando per quanto possibile l'accuratezza dei 10 bit? Si può fare una **interpolazione lineare**. Il beneficio dell'interpolazione diventa molto evidente nel passaggio da 10 a 3 bit, cioè ponendo `inf_ris = 8`. Con l'interpolazione lineare, il metodo `draw()` va così riscritto:

```

void draw() {
  if ((mouseX >= 0) && (mouseX < sup_ris) && (mousePressed))
  {
    int flo = floor(float(mouseX)/sup_ris*inf_ris);
    float alpha = float(mouseX)/sup_ris*inf_ris - float(flo);
    background((1-alpha)*midimap[flo] + alpha*midimap[(flo+1)%midimap.length]);
  }
}

```

`alpha` è il resto del troncamento a $\log_2 \text{inf_res}$ bit dell'indice di $\log_2 \text{sup_res}$ bit, e viene usato per pesare due elementi contigui dell'array a `inf_res` elementi. Quale è la funzione dell'operazione di modulo `%midimap.length`?

Inversione di una mappa

Ha senso costruire una mappa inversa, ad esempio per ritrovare l'indice a partire da un valore di proprietà, solo se la mappa di partenza è monotona, crescente o decrescente, in modo da stabilire una corrispondenza biunivoca tra indice e valore. Ad esempio, pre-caricando un array con un quarto di seno campionato in 128 punti, come costruisco un array che contiene la mappa inversa, cioè l'arcoseno?

```

int ris = 128;
float[] midimap = new float[ris];
float[] invmap = new float[ris];

void setup() {
  size(ris,ris);
  colorMode(RGB, 1.0);
  /* inizializza */
  for (int i = 0; i < midimap.length; i++) midimap[i] = (ris - 1) * sin(PI/2*(float)i/midimap.length);
  /* inverti */
  for (int i = 0; i < midimap.length; i++) invmap[round(midimap[i])] = i;
}

```

```

/* disegna */
for (int i = 0; i < midimap.length; i++)
{
    point(i, ris - midimap[i]);
    point(i, ris - invmap[i]);
}
println(invmap);
}

```

Exercise 1

(Solution on p. 10.)

Si noti, dal printout prodotto dalla inversione di una mappa (Inversione di una mappa, p. 3), che la mappa inversa così costruita ha parecchi buchi (zeri). Come si possono eliminare tali buchi?

Mappe lineari

In molti casi i sensori si comportano linearmente e ciò che serve è una mappa lineare. In tal caso, non è necessario scomodare un array per realizzarla. Ad esempio, si supponga che un sensore di distanza restituisca un valore in metri che corrisponde alla quantizzazione della distanza massima di 10 metri in 1024 valori (10 bit). Se con tale sensore vogliamo mappare diverse posizioni da 1 a 3 metri in indici di pixel da 0 a 127, si può semplicemente notare che la distanza di 1 metro produce il valore 102, mentre la distanza di 3 metri produce il valore 307. Detto d il valore di distanza tra 0 e 1023 restituito dal sensore, il pixel da accendere è $\frac{128}{307-102} (d - 102)$

Istogrammi

Le operazioni che si basano sul conteggio di elementi vengono supportate bene da rappresentazioni mediante array. E' questo il caso della produzione dell'istogramma³. In elaborazione di immagini questa è una operazione che si fa assai di frequente in quanto supporta diverse elaborazioni efficaci dei colori.

Lettura

Si legga il capitolo 10 di How to Think Like a Computer Scientist⁴

2 Array nel visual programming

Gli array, come liste ordinate di numeri, si prestano naturalmente ad una rappresentazione visuale, e quindi trovano un impiego esteso nel visual programming⁵, quale è quello che si pratica con gli ambienti Max/MSP⁶ e Pure Data⁷ (Pd).

In Pd un array può essere allocato mediante la primitiva `table`, alle quale viene passato un nome e una lunghezza. Tutti gli array sono memorizzati come numeri floating point. Il contenuto degli array può essere visualizzato in una finestra facendo "open" sull'oggetto `table`. Ad esempio, se creo un oggetto con `table mappa 64`, l'interprete costruisce un subpatch contenente un array, il quale si può visualizzare cliccando sull'oggetto stesso. Si può accedere ad ogni singolo elemento dell'array con gli oggetti `tabread mappa` e `tabwrite mappa`, rispettivamente. In più, gli array in Pd hanno metodi che supportano varie operazioni. Ad esempio:

- una sequenza di valori crescente tra -3 a 3 che inizia alla locazione di posizione 4 si può impostare con il messaggio `; mappa 4 -3 -2 -1 0 1 2 3`;
- l'array di nome `mappa` si può rinominare `map` mediante il messaggio `; mappa rename map`;
- il riquadro che contiene la rappresentazione visuale dell'array si può ridimensionare con un messaggio del tipo `; map bounds 0 -2 10 2`

³<http://cnx.org/content/m12838/latest/#histogram>

⁴<http://www.greenteapress.com/thinkajava/>

⁵http://en.wikipedia.org/wiki/Visual_programming

⁶<http://en.wikipedia.org/wiki/Max/MSP>

⁷http://en.wikipedia.org/wiki/Pure_Data

- Sulla cornice del riquadro si possono porre delle marcature metriche con un messaggio del tipo `; map xticks 0 0.1 5` nonché dei valori di riferimento con `; map xlabel -1.1 0 1 2 3 4 5`

3 Matrici

Qualsiasi tipo si può usare come tipo base per un array (`int`, `float`, `string`, ecc.). In particolare, un array può essere tipo base per un array, in questo modo consentendo di fare array di array, o array bidimensionali. Un array bidimensionale si dichiara e si istanzia con

```
int[][] A = new int[ROWS][COLUMNS];
```

dove `ROWS` e `COLUMNS` conterranno i numeri di righe e colonne che si vogliono riservare per l'array. Ogni riga è essa stessa un array, al quale si può accedere con `A[i]`, mentre all'elemento di riga `i` e colonna `j` si accede con `A[i][j]`, che in questo caso è una variabile di tipo `int`. Questa volta `A.length` restituisce il numero di righe dell'array, mentre il numero di colonne si può conoscere con `A[0].length`.

Un array, mono- o bi-dimensionale, può essere inizializzato all'atto della creazione. Nel caso di un array a 3 righe e 4 colonne ciò si può fare con

```
int[][] A = { { 1, 0, 12, -1 },
              { 7, -3, 2, 5 },
              { -5, -2, 2, -9 }
            };
```

Così come gli array monodimensionali vengono scanditi agevolmente con un ciclo `for`, così gli array bidimensionali vengono scanditi con due cicli `for` annidati, uno che scandisce le righe e uno che scandisce le colonne:

```
for (int i = 0; i < ROWS; i++) {
    for (int j = 0; j < COLUMNS; j++) {
        A[i][j] = ...
    }
}
```

Ad esempio, nel codice seguente viene inizializzato un array bidimensionale di colori, e successivamente questa tavolozza di colori viene visualizzata.

```
int ROWS = 4;
int COLUMNS = 4;
color[][] grid = new color[ROWS][COLUMNS];

for (int row = 0; row < ROWS; row++) {
    for (int col = 0; col < COLUMNS; col++) {
        grid[row][col] = color(row*60, 0, col*60);
    }
}
```

```

for (int row = 0; row < ROWS; row++) {
    for (int col = 0; col < COLUMNS; col++) {
        fill(grid[row][col]);
        rect(row*25, col*25, 25, 25);
    }
}

```

Exercise 2*(Solution on p. 10.)*

Si visualizzi una tavolozza di 4x4 colori che ruota in senso orizzontale, come se fosse la superficie di un cilindro che ruota intorno all'asse verticale.

Gli array multidimensionali di tipo `float` o `double` sono utilizzati in moltissimi contesti, in quanto sono di supporto alla Aritmetica delle Matrici⁸. Questa aritmetica consente di rappresentare ed effettuare in maniera compatta operazioni su array di numeri. Ad esempio, per invertire l'ordine dei numeri di un array

di 4 elementi è sufficiente pre-moltiplicarlo per una matrice $\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$ oppure, per scambiare i numeri

in posizione pari con quelli in posizione dispari si può usare la matrice $\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Nella computer grafica interattiva, punti e vettori sono rappresentati in coordinate omogenee⁹ mediante array di quattro elementi, e tutte le trasformazioni geometriche¹⁰ (traslazioni, rotazioni, scalamenti, proiezioni) sono realizzate con prodotti matrice-vettore.

Prodotto Matrice-Vettore

L'inversione tra elementi di posto pari ed elementi di posto dispari mediante prodotto matrice-vettore è illustrata dal codice

```

int DIM = 4;
float[][] myMatrix = {{0, 1, 0, 0},
                      {1, 0, 0, 0},
                      {0, 0, 0, 1},
                      {0, 0, 1, 0}
                      };

float[] myVector = {0.1, 0.2, 0.3, 0.4};
float[] newVector = new float[4];

for (int i=0; i<DIM; i++)
    for (int j=0; j<DIM; j++)
        newVector[i] += myMatrix[i][j] * myVector[j];

```

⁸"Matrix Arithmetic" <<http://cnx.org/content/m10090/latest/>>

⁹"Rappresentazione di Media in Processing", Definition 1: "Coordinate Omogenee" <http://cnx.org/content/m12664/latest/#coordinate_omogenee>

¹⁰"Rappresentazione di Media in Processing": Section Traslazioni, Rotazioni, e Trasformazioni di Scala <http://cnx.org/content/m12664/latest/#rotazioni_traslazioni>

```
println(myVector); println();
println(newVector);
```

Exercise 3

(Solution on p. 11.)

Come è fatta la matrice che, pre-moltiplicata per un array, ne ruota circolarmente di una posizione gli elementi?

Example 2: Many-to-Many Mapping

In molte applicazioni di interaction design (si veda, ad esempio, il caso della interactive sonification¹¹), ci si trova a dover mappare una molteplicità di variabili, il cui valore può venire dalla lettura di sensori, in una molteplicità di parametri algoritmici. Spesso tale mapping è lineare ed esprimibile con un prodotto matrice-vettore. Ad esempio, **distanza, velocità, e temperatura** si possono mappare in **posizione, colore, e frequenza di oscillazione** di un oggetto grafico, in maniera che, ad esempio, il colore dipenda da una combinazione di velocità e temperatura. Il funzionamento delle Mixing Console¹² si può interpretare come mapping multivariato lineare degli ingressi nelle uscite. Purtroppo, molti mapping di interesse sono non-lineari, e non esprimibili mediante prodotto matrice-vettore. Ad esempio, la trasformazione delle coordinate di rappresentazione dei colori¹³ da RGB a HSB¹⁴ è non lineare.

Letture

Si legga la sezione Multi-Dimensional Arrays¹⁵ di Data Structures and Algorithms with Object-Oriented Design Patterns in Java¹⁶ o, ancor meglio, la sezione Multi-Dimensional Arrays¹⁷ di Introduction to Programming Using Java¹⁸.

4 Mappe

Gli array soffrono di due limitazioni principali:

- la loro occupazione di memoria va pre-impostata;
- gli indici sono obbligatoriamente numeri interi.

Quando si parla di mappe, in ambito delle strutture dati, ci si riferisce di solito a strutture più generali, la cui dimensione è dinamicamente allocabile, e nelle quali l'accesso agli elementi può avvenire attraverso oggetti di tipo arbitrario, chiamati **key** (chiave).

Letture

Si legga il capitolo 19 di How to Think Like a Computer Scientist¹⁹

5 Matrici nel Physical Computing

Nel progetto WAV²⁰, la classe Matrix²¹ per l'ambiente Wiring²² di physical computing²³ è stata usata per pilotare una matrice di LED a 8 righe e 16 colonne per mezzo di un paio di chip MAXIM MAX7219²⁴

¹¹http://www.icad.org/websiteV2.0/Conferences/ICAD2004/papers/hunt_hermann.pdf

¹²http://en.wikipedia.org/wiki/Mixing_console

¹³"Rappresentazione di Media in Processing" <http://cnx.org/content/m12664/latest/#color_type>

¹⁴http://en.wikipedia.org/wiki/HSV_color_space#From_RGB_to_HSV

¹⁵<http://www.brpreiss.com/books/opus5/html/page89.html#SECTION00520000000000000000>

¹⁶<http://www.brpreiss.com/books/opus5/>

¹⁷<http://math.hws.edu/javanotes/c7/s5.html>

¹⁸<http://math.hws.edu/javanotes/>

¹⁹<http://www.greenteapress.com/thinkajava/>

²⁰http://www.interaction-venice.com/courses/06-07Lab2/?page_id=135

²¹<http://wiring.org.co/reference/libraries/Matrix/index.html>

²²<http://wiring.org.co/index.html>

²³<http://itp.nyu.edu/physcomp/>

²⁴http://www.maxim-ic.com/quick_view2.cfm/qv_pk/1339

. Supponiamo per semplicità di dover gestire una matrice a 8 righe e 8 colonne per mezzo di un solo chip MAX7219 e di voler propagare una cresta d'onda da sinistra a destra. La matrice viene dichiarata e istanziata come

```
Matrix myMatrix = Matrix(0, 2, 1);
```

Si può predisporre un array bidimensionale di bit 8x8 mediante istanziazione della classe `Sprite`²⁵ :

```
Sprite wave = Sprite(
    8, 8,
    b10000000,
    b10000000,
    b01000010,
    b01000010,
    b00100100,
    b00100100,
    b00011000,
    b00011000,
    );
```

e scrivere una funzione da eseguirsi a clockrate

```
void loop()
{
    myMatrix.write(i, 1, wave);    // place sprite on screen (i is an int)
    delay(1000);                  // wait a little bit
    myMatrix.clear();             // clear the screen for next animation frame
    i = 1 + (i+1)%8;              // circular (between 1 and 8) increment of i
}
```

oppure, alternativamente, procedere direttamente alla scrittura ciclica

```
void loop()
{
    myMatrix.clear(); // clear display

    delay(1000);

    // turn pixels on (i is an int that is forced to stay between 0 and 7)
    // decreasing ramp
    myMatrix.write(1, 1+(i)%8, HIGH);
    myMatrix.write(2, 1+(i)%8, HIGH);
    myMatrix.write(3, 1+(i+1)%8, HIGH);
    myMatrix.write(4, 1+(i+1)%8, HIGH);
    myMatrix.write(5, 1+(i+2)%8, HIGH);
    myMatrix.write(6, 1+(i+2)%8, HIGH);
    myMatrix.write(7, 1+(i+3)%8, HIGH);
    myMatrix.write(8, 1+(i+3)%8, HIGH);
    // increasing ramp
```

²⁵<http://wiring.org.co/reference/libraries/Sprite/index.html>

```
        myMatrix.write(3, 1+(i+6)%8, HIGH);
        myMatrix.write(4, 1+(i+6)%8, HIGH);
        myMatrix.write(5, 1+(i+5)%8, HIGH);
        myMatrix.write(6, 1+(i+5)%8, HIGH);
        myMatrix.write(7, 1+(i+4)%8, HIGH);
        myMatrix.write(8, 1+(i+4)%8, HIGH);
    i++;
}
```

Solutions to Exercises in this Module

Solution to Exercise (p. 4)

Il codice seguente riempie i buchi per interpolazione, ma non funziona se i buchi sono fatti da più zeri contigui:

```
for (int i = 1; i < invmap.length-1; i++)
  if (invmap[i] < 0.0001) // se buco, fai la media degli adiacenti
    invmap[i] = (invmap[i-1] + invmap[i+1])/2; /* non funziona se i buchi sono fatti
    da più zeri contigui */
println(invmap);
```

Solution to Exercise (p. 6)

```
int ROWS = 4;
int COLUMNS = 4;
color[][] grid = new color[ROWS][COLUMNS];
int step;

void setup(){
  for (int row = 0; row < ROWS; row++) {
    for (int col = 0; col < COLUMNS; col++) {
      grid[row][col] = color(row*60, 0, col*60);
    }

    frameRate(5);
    noStroke();
  }

  for (int row = 0; row < ROWS; row++) {
    for (int col = 0; col < COLUMNS; col++) {
      fill(grid[row][col]);
      rect(row*25, col*25, 25, 25);
    }
  }
}

void draw(){
  for (int row = 0; row < ROWS; row++) {
    for (int col = 0; col < COLUMNS; col++) {
      fill(grid[(row+step)%COLUMNS][col]);
      rect(row*25, col*25, 25, 25);
    }
  }
  step = (step+1)%COLUMNS;
}
```

Solution to Exercise (p. 7)

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$
 Una matrice fatta così è un caso particolare di matrice circolante²⁶.

²⁶http://en.wikipedia.org/wiki/Circulant_matrix