

Programmazione Funzionale e Parallela (A.A. 2016-2017)

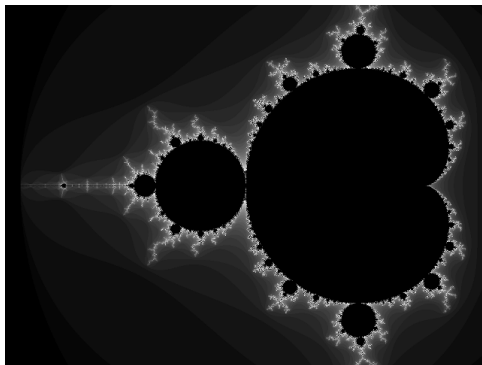
Corsi di Laurea in Ingegneria Informatica e Automatica, Ingegneria dei Sistemi Informatici, e Laurea Magistrale in Ingegneria Informatica

Sapienza Università di Roma

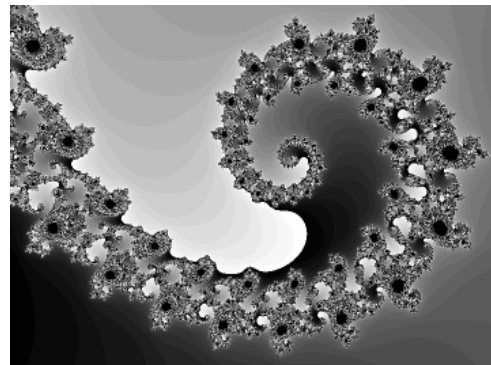
Esercitazione di laboratorio del 21 dicembre 2016

Visualizzazione dell'insieme di Mandelbrot

L'insieme di Mandelbrot è un'affascinante struttura matematica frattale definita come l'insieme dei punti $p = (x, y)$ nel piano complesso per cui iterando la formula $z \leftarrow z^2 + p$ a partire da $z = (0,0)$, il punto z rimane entro distanza 2 dall'origine.



(a) Insieme di Mandelbrot con centro di coordinate $(-0.75, 0)$ e ingrandimento 1x



(b) Porzione con centro di coordinate $(-0.7453, 0.1127)$ e ingrandimento 1500x

Per sua natura, l'insieme di Mandelbrot può essere visualizzato come un'immagine a due dimensioni in cui ogni pixel viene fatto corrispondere a un punto nel piano complesso, colorato in base al seguente semplice algoritmo (*escape time algorithm*):

1. se dopo un certo numero massimo di iterazioni z rimane entro distanza 2 dall'origine, il pixel corrisponde probabilmente a un punto nell'insieme di Mandelbrot e viene colorato di nero;
2. altrimenti il pixel certamente **non** corrisponde a un punto nell'insieme di Mandelbrot¹ e il suo colore dipende dal numero di iterazioni che sono servite portare z a distanza maggiore di 2.

In questo esercizio, viene fornita una funzione C che crea un'immagine a toni di grigio della porzione dell'insieme di Mandelbrot contenuta in un rettangolo, opportunamente scalato, centrato in un determinato punto c del piano complesso. La funzione, che implementa l'*escape time algorithm*, ha il seguente prototipo:

¹ Paradossalmente, i punti colorati che rendono maggiormente suggestive le visualizzazioni del frattale di Mandelbrot sono proprio quelli che **non** appartengono all'insieme.

```
void mandelbrot_host(float rc, float ic, float scale,
                    int w, int h, int max_iter,
                    unsigned char* img, double* t)
```

Dove:

1. (rc, ic) sono le coordinate nel piano complesso del punto c in cui è centrata l'immagine;
2. $scale$ è la distanza orizzontale e verticale fra i punti del piano complesso che corrispondono a pixel adiacenti;
3. w e h sono rispettivamente la dimensione orizzontale e quella verticale dell'immagine, in pixel;
4. max_iter è il massimo numero di iterazioni della formula $z \leftarrow z^2 + p$;
5. img è un puntatore a una matrice di dimensione $w \times h$, disposta in memoria in formato row-major, che rappresenta l'immagine creata dalla funzione;
6. t è un puntatore a un oggetto di tipo `double` in cui viene scritto il tempo richiesto dalla funzione, in secondi.

Si noti che la funzione `mandelbrot_host()`²:

1. realizza l'operazione $z \leftarrow z^2 + p$ come segue:

```
float new_iz = ip + 2.0*rz*iz;
float new_rz = rp + rz*rz - iz*iz;
iz = new_iz;
rz = new_rz;
```

dove:

- rz e iz rappresentano la parte reale e immaginaria di $z = (rz, iz)$;
 - rp e ip rappresentano la parte reale e immaginaria di $p = (rp, ip)$.
2. assegna il colore nero ai punti per cui il numero di iterazioni k ha raggiunto il massimo ($k == max_iter$) e assegna il colore $8 * 255.0 * k / max_iter$ ai punti per cui z è arrivato oltre distanza 2 dall'origine ($rz*rz + iz*iz >= 4.0$).

Obiettivo.

L'obiettivo di questo esercizio è fornire una versione data-parallel dell'escape time algorithm implementato dalla funzione `mandelbrot_host()`, usando OpenCL.

Si vada nella directory di lavoro `mandelbrot` e:

1) si definisca nel file `mandelbrot.cl` un kernel OpenCL C che calcoli il tono di grigio di ogni pixel dell'immagine da produrre, usando l'algoritmo *escape time* fornito dalla funzione `mandelbrot_host`.

2) si definisca nel file `mandelbrot_device.c` la funzione:

```
void mandelbrot_device(float rc, float ic, float scale,
                      int w, int h, int max_iter,
                      unsigned char* img, clut_device* dev,
                      double* t)
```

² Si veda il file: `mandelbrot/mandelbrot_host.c`

con gli stessi parametri di `mandelbrot_host()` e l'ulteriore parametro `dev` di tipo `clut_device*` (si veda `clut.h`) che contiene l'ambiente di esecuzione OpenCL corrente. La funzione deve:

- istanziare ed eseguire il kernel definito in `mandelbrot.cl` per calcolare il valore di ogni pixel dell'immagine `img` di dimensione $w \times h$;
- usare un `NDRange` bidimensionale con `local size` maggiore di 1 (es. 8 o 16) in modo da avere un work item per ogni pixel dell'immagine da generare;
- scrivere in `*t` il tempo richiesto dall'esecuzione del kernel usando la funzione `clut_get_duration()` (si veda `clut.h`);
- gestire ogni errore stampando un messaggio e terminando il programma (si usi ad esempio: `clut_check_err(err, "messaggio errore")` dichiarata in `clut.h`);
- deallocare correttamente tutti gli oggetti OpenCL allocati dalla funzione.

Compilazione e test.

- *Directory di lavoro:* `mandelbrot/`
- *Compilazione programma:* dare il comando `make`, che genera il file eseguibile `mandelbrot`;
- *Compilazione ed esecuzione:* dare il comando `make test`.

Nelle directory `img_device` e `img_host` verranno generate varie immagini che mostrano vari "luoghi suggestivi" dell'insieme di Mandelbrot:

- `img_host`: immagini generate dal codice host (fornito);
- `img_device`: immagini generate dal codice device da scrivere come esercizio (a un'ispezione visuale devono essere uguali a quelle in `img_host`)

Le immagini, salvate in formato PGM (Portable Graymap Format), possono essere visualizzate con il programma Gimp.