

# Introduzione alla Generazione di Immagini Fotorealistiche

*Corso di Dottorato in Matematica e Informatica  
Università degli Studi della Basilicata*

Dott. Ugo Erra

*4° Lezione – Antialiasing*

# Sommario

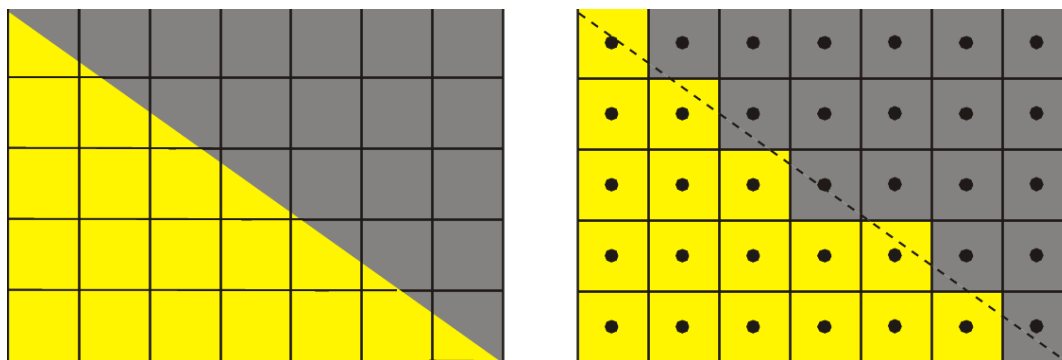
- Aliasing
  - Campionamento
-

# Aliasing

- L'*aliasing* (dal latino alias, altrove) è il fenomeno per il quale due segnali analogici diversi possono diventare indistinguibili una volta campionati
  - Il ray tracing è un processo discreto di campionamento in cui si genera un'immagine attraverso un numero finito di pixel
    - Ogni raggio è generato a partire dal centro del pixel attraverso un processo di point-sampling
-

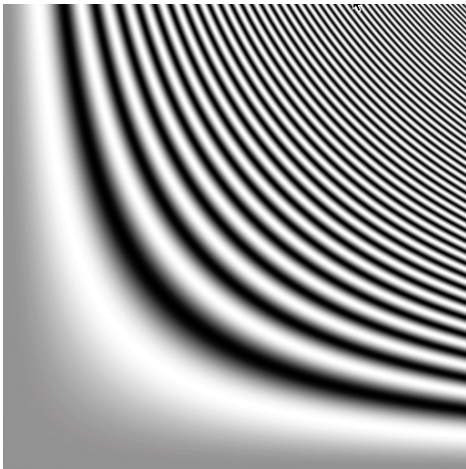
# Jaggies

- Il *jaggies* è una forma di aliasing
- Durante il campionamento l'errore introdotto nel caso di linee continue porta a visualizzarle successivamente in forma scalettata
- Il jaggies noto anche come staircases (scale) si nota particolarmente sui bordi e sulle linee diagonali

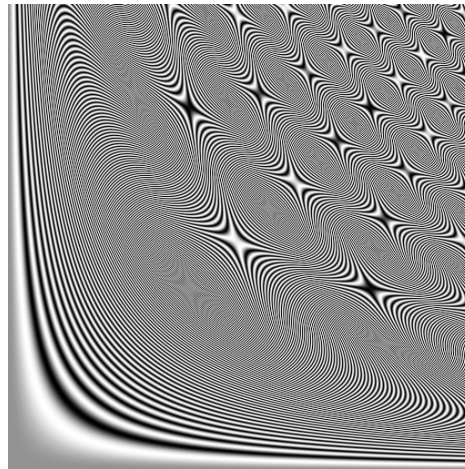


# Moirè pattern

- L'*effetto moirè* è una forma di aliasing causata da due pattern simili non completamente sovrapposti (pattern di interferenza)
- Ad esempio due griglie parallele con maglie distanziate in modo leggermente diverso



$(x,y) \in [0, 3.79]^2$



$(x,y) \in [0, 10.83]^2$

$$f(x,y) = 1/2(1 + \sin(x^2y^2))$$

# Antialiasing

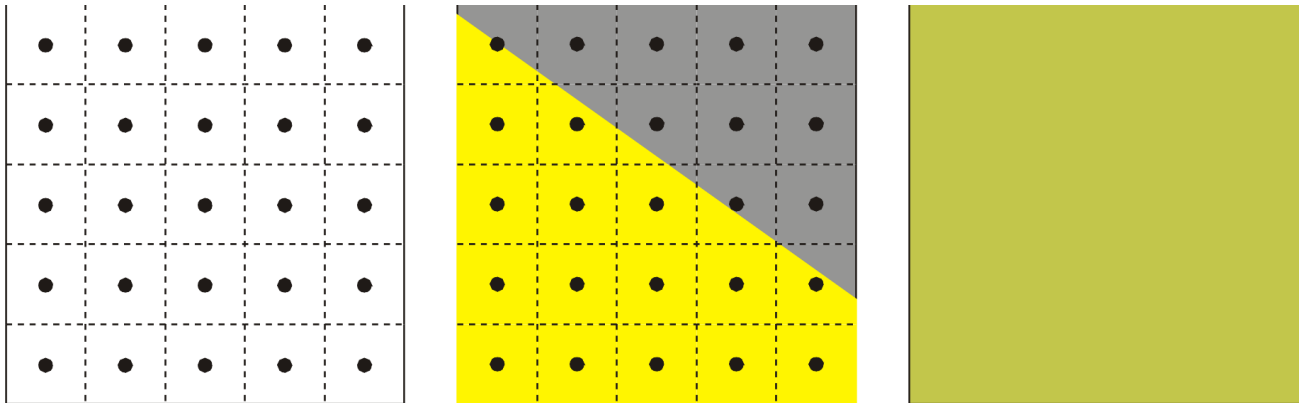
- L'antialiasing (AA) è una tecnica per ridurre l'effetto aliasing
  - Nel ray tracing l'antialiasing si ottiene generando più di un raggio per ogni pixel
  - Alcune tecniche
    - Aumentare la risoluzione
    - Campionamento uniforme
    - Campionamento random
    - Campionamento jittered
-

# Aumentare la risoluzione

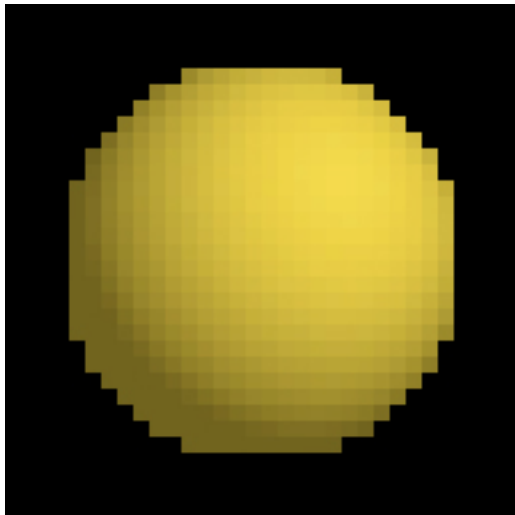
- Un semplice tecnica di AA consiste nell'aumentare la risoluzione dell'immagine
    - La dimensione del pixel deve essere ridotta
  - In generale non riduce di molto l'aliasing
-

# Campionamento uniforme

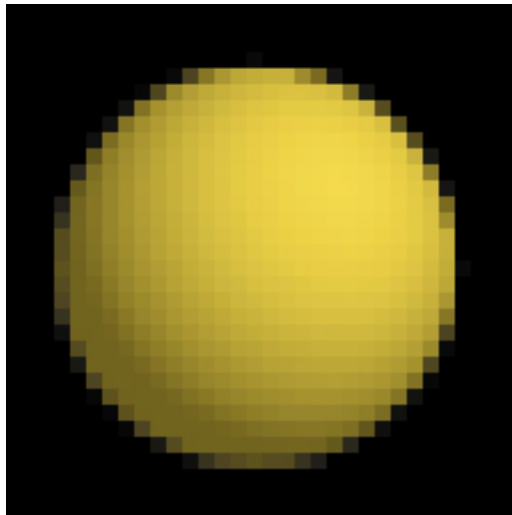
- Consiste nel suddividere il pixel con una griglia regolare di celle
- Il colore del pixel è calcolato come la media dei colori dei raggi generati per ogni cella



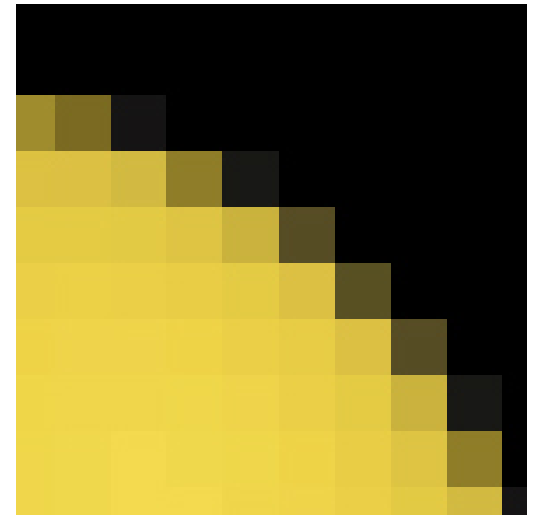
# Esempio campionamento uniforme



1 campione per pixel



16 campioni per pixel



# Metodo world:render\_scene

```
void world::render_scene(void) const {
    RGBColor      pixel_color;
    Ray           ray;
    float         zw          = 100.0;           // hardwired in
    int           n          = (int)sqrt((float)vp.num_samples);
    Point2D       pp;

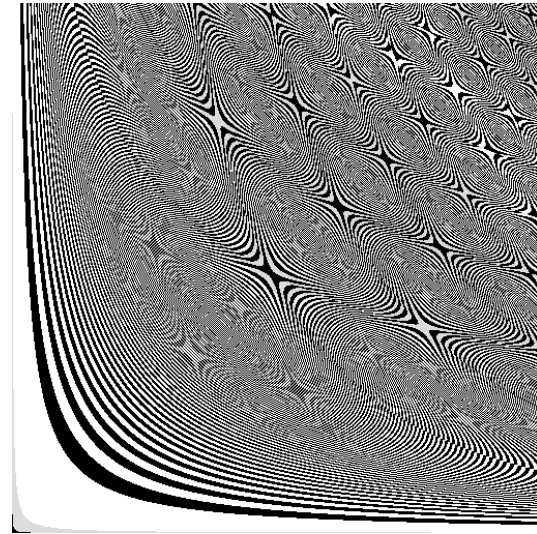
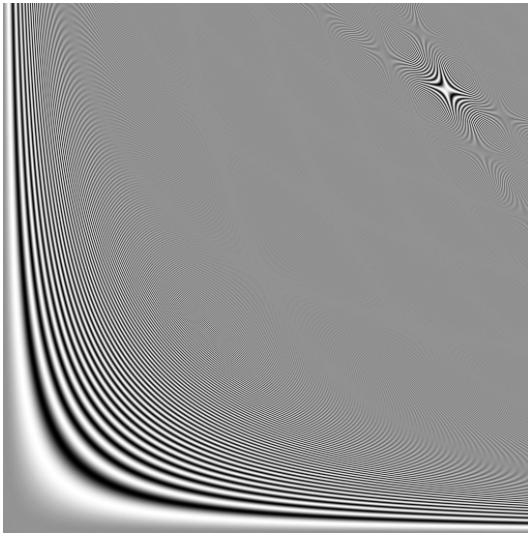
    ray.d = Vector3D(0, 0, -1);

    for (int r = 0; r < vp.vres; r++)           // up
        for (int c = 0; c <= vp.hres; c++) {    // across
            pixel_color = black;

            for (int p = 0; p < n; p++)
                for (int q = 0; q < n; q++) {
                    pp.x = vp.s * (c - 0.5 * vp.hres + (q + 0.5) / n);
                    pp.y = vp.s * (r - 0.5 * vp.hres + (p + 0.5) / n);
                    ray.o = Point3D(pp.x, pp.y, zw);
                    pixel_color += tracer_ptr->trace_ray(ray);
                }
            pixel_color /= (float)vp.num_samples;
            display_pixel(r, c, pixel_color);
        }
}
} q=0 0.5/10 q=1 1+0.5 / 10 q=n-1 n-0.5 / 10
```

# Campionamento uniforme

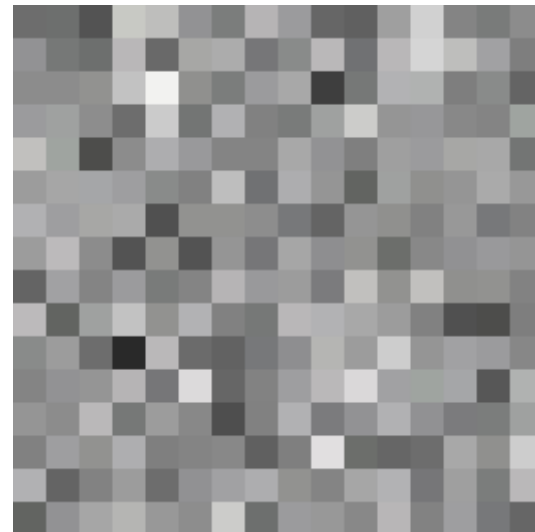
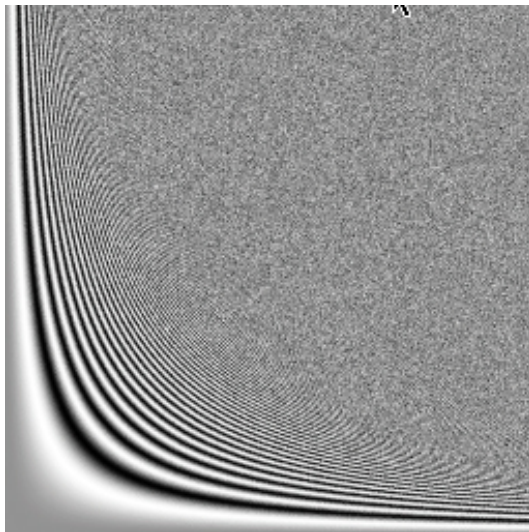
- Il campionamento uniforme riduce l'aliasing ma può essere ancora presente ad esempio come pattern di moirè



25 campioni per pixel

# Campionamento random

- L'idea è di generare dei raggi casuali all'interno del pixel in modo da sostituire l'aliasing con del rumore
- L'occhio umano riesce a tollerare molto meglio il rumore rispetto alla presenza di aliasing regolare
- Il rumore tuttavia rappresenta l'immagine in maniera meno accurata rispetto all'aliasing



# Metodo world:render\_scene

```
void world::render_scene(void) const {
    RGBColor      pixel_color;
    Ray           ray;
    float         zw          = 100.0;           // hardwired in
    int           n          = (int)sqrt((float)vp.num_samples);
    Point2D       pp;

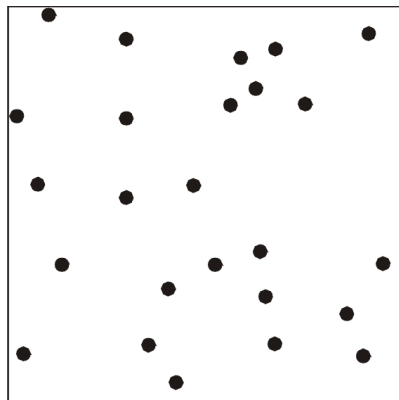
    ray.d = Vector3D(0, 0, -1);

    for (int r = 0; r < vp.vres; r++)           // up
        for (int c = 0; c <= vp.hres; c++) {    // across
            pixel_color = black;

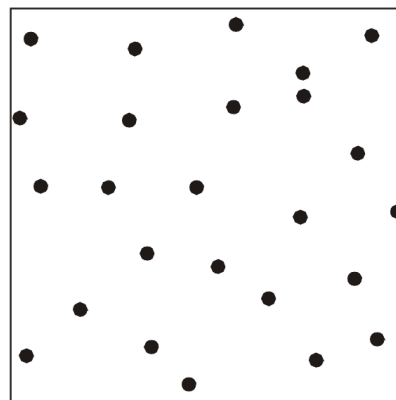
            for (int p = 0; p < n; p++)
                for (int q = 0; q < n; q++) {
                    pp.x = vp.s * (c - 0.5 * vp.hres + rand_float());
                    pp.y = vp.s * (r - 0.5 * vp.hres + rand_float());
                    ray.o = Point3D(pp.x, pp.y, zw);
                    pixel_color += tracer_ptr->trace_ray(ray);
                }
            pixel_color /= (float)vp.num_samples;
            display_pixel(r, c, pixel_color);
        }
    }
```

# Campionamento jittered

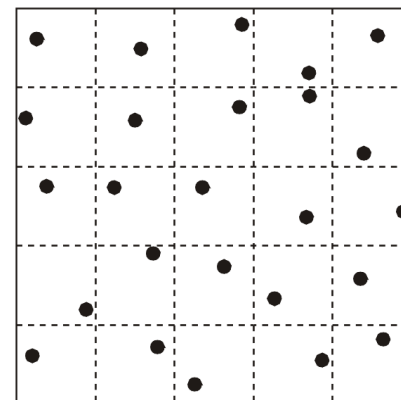
- Il campionamento random soffre di potenziali addensamenti o eccessiva separazione dei campioni
- Con il campionamento jittered il pixel è suddiviso in celle mediante una griglia regolare
- I raggi vengono campionati in maniera random all'interno di ogni cella



random

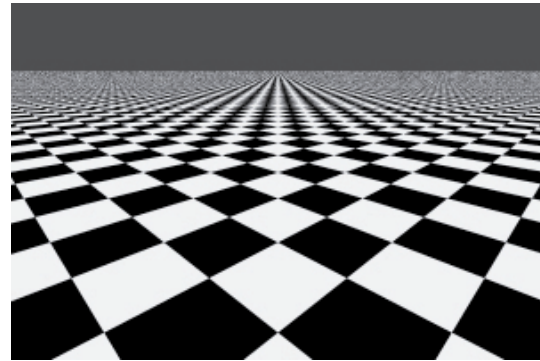
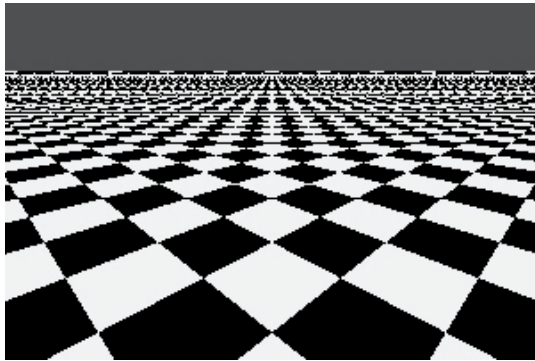


jittered



# Dettagli e antialiasing

- Le texture con molti dettagli sono soggette a problemi di aliasing



# Filtering

- Il filtering consiste nel determinare il colore di un pixel utilizzando i raggi dei pixel adiacenti
  - In alcuni contesti l'antialiasing è trattato insieme al filtering
- Per ottenere il colore finale i raggi vengono pesati in maniera differente
  - Box filter
  - Tent filter
  - Cubic filter
  - Gaussian filter

