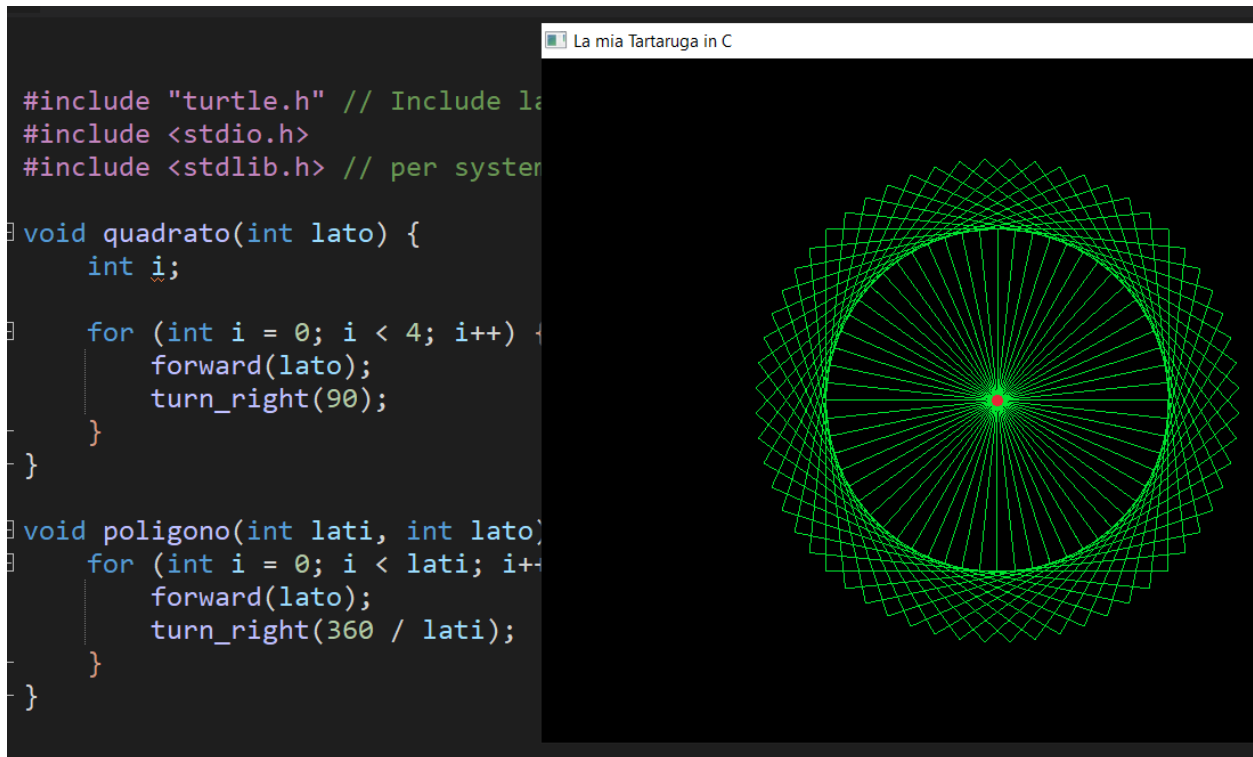


turtle.h



— La Grafica della Tartaruga in C con Raylib

Indice

1. [Introduzione](#)
 2. [Requisiti](#)
 3. [Installazione e configurazione](#)
 4. [Primo programma: un quadrato](#)
 5. [Come funziona la tartaruga](#)
 6. [Comandi di movimento](#)
 7. [Comandi di rotazione](#)
 8. [Penna e colore](#)
 9. [Posizionamento assoluto](#)
 10. [Pulire la tela](#)
 11. [Chiudere il programma: turtle_wait\(\)](#)
 12. [Esempi completi](#)
 13. [Riferimento API](#)
 14. [Domande frequenti](#)
-

Introduzione

`turtle.h` è una libreria **header-only** che porta la classica *grafica della tartaruga* (turtle graphics) nel linguaggio C, sfruttando [Raylib](#) per la finestra e il rendering.

L'idea è semplice: immagini una tartaruga che cammina su una tela. Puoi darle comandi come:

- **avanza** di N pixel,
- **gira a destra/sinistra** di N gradi,
- **alza/abbassa la penna**,
- **cambia colore**.

La tartaruga lascia una scia quando la penna è abbassata, esattamente come nel linguaggio Logo. È uno strumento didattico perfetto per imparare:

- la **geometria** (angoli, poligoni, coordinate);
 - la **programmazione** (cicli, funzioni, ricorsione);
 - i **frattali** (Koch, Sierpinski, alberi...).
-

Requisiti

- Un compilatore C (GCC, Clang, MSVC...).
- La libreria [Raylib](#) installata.
- Un ambiente con finestre grafiche.

Installazione di Raylib

Windows (MSYS2 / MinGW):

```
pacman -S mingw-w64-x86_64-raylib
```

Linux (Debian/Ubuntu):

```
sudo apt install libraylib-dev
```

macOS (Homebrew):

```
brew install raylib
```

In alternativa: binari precompilati da <https://www.raylib.com/>.

Installazione e configurazione

Metti `turtle.h` nella stessa cartella del tuo file `.c` (o in una cartella inclusa con `-I`).

Regola importante: `turtle.h` va incluso in **un solo file** `.c` del progetto. Contiene variabili `static`, quindi includerlo più volte creerebbe definizioni multiple.

Compilazione

```
gcc programma.c -o programma -lraylib -lm -lpthread -ldl -lX11
```

Su Windows con MinGW:

```
gcc programma.c -o programma.exe -lraylib -lopengl32 -lgdi32 -lwinmm
```

Primo programma: un quadrato

Il programma più semplice: disegnare un quadrato.

```
#include "turtle.h"

int main(void)
{
    /* Finestra 800x600, titolo "Il mio primo quadrato" */
    turtle_init(800, 600, "Il mio primo quadrato");

    /* Disegna un quadrato di lato 200 */
    for (int i = 0; i < 4; i++) {
        forward(200);
        turn_right(90);
    }

    /* Tieni la finestra aperta finché non la chiudi */
    turtle_wait();

    return 0;
}
```

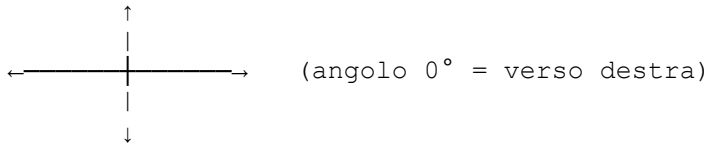
Cosa succede:

1. `turtle_init(800, 600, "...")` apre una finestra 800×600 con sfondo nero e mette la tartaruga al centro, rivolta verso destra (angolo 0°).
 2. Il ciclo `for` ripete 4 volte: avanza di 200 pixel, gira a destra di 90°.
 3. Alla fine la tartaruga torna al punto di partenza.
 4. `turtle_wait()` blocca il programma finché non chiudi la finestra (o premi ESC).
-

Come funziona la tartaruga

Il modello mentale

Immagina una tartaruga su un foglio:



- La tartaruga ha una **posizione** (x , y) sullo schermo.
- Ha un **orientamento** (angolo in gradi).
- Ha una **penna** che può essere su o giù.
- Ha un **colore**.

Angoli

Gli angoli seguono la convenzione dello **schermo** (asse Y verso il basso):

Angolo	Direzione
0°	→ destra
90°	↓ giù
180°	← sinistra
270°	↑ su

💡 **Attenzione:** è l'opposto della convenzione matematica standard (dove 90° è in alto). Questo perché in grafica lo schermo ha l'asse Y che punta verso il basso.

Coordinate

- L'origine (0, 0) è in **alto a sinistra**.
- La tartaruga parte al **centro** dello schermo: (width/2, height/2).
- Per una finestra 800×600, parte a (400, 300).

Comandi di movimento

forward(distanza)

Muove la tartaruga in **avanti** di distanza pixel nella direzione corrente.

```
forward(100);    /* avanza di 100 pixel */
```

Se la penna è abbassata, lascia una scia del colore corrente.

backward(distanza)

Muove la tartaruga **all'indietro** di distanza pixel.

```
backward(100);   /* equivalente a forward(-100) */
```

Comandi di rotazione

`turn_right(gradi)`

Gira a **destra** (senso orario) di gradi gradi.

```
turn_right(90);    /* ora la tartaruga punta in basso */
```

`turn_left(gradi)`

Gira a **sinistra** (senso antiorario) di gradi gradi.

```
turn_left(90);    /* ora la tartaruga punta in alto */
```

💡 **Nota:** le rotazioni sono **relative** alla direzione corrente.

Per impostare una direzione assoluta, usa `set_heading()`.

Penna e colore

`pen_down()`

Abbassa la penna: la tartaruga lascia una scia quando si muove.

```
pen_down();
```

`pen_up()`

Alza la penna: la tartaruga si muove senza lasciare scia.

```
pen_up();  
forward(100);    /* nessuna linea disegnata */  
pen_down();
```

Utile per spostarsi in un altro punto senza tracciare linee.

`set_color(Color color)`

Cambia il colore della penna. I colori sono quelli di Raylib:

```
set_color(RED);  
set_color(BLUE);  
set_color(GREEN);  
set_color(WHITE);  
set_color(YELLOW);  
set_color((Color){ 255, 128, 0, 255 });    /* arancione personalizzato */
```

Colori predefiniti utili: RED, GREEN, BLUE, YELLOW, ORANGE, PURPLE, PINK, WHITE, BLACK, GRAY, LIME, SKYBLUE, VIOLET, BROWN, GOLD.

Posizionamento assoluto

`go_to(x, y)`

Sposta **istantaneamente** la tartaruga alle coordinate (x, y) **senza disegnare** (anche se la penna è abbassata).

```
go_to(100, 100);    /* teletrasporto senza scia */
```

`set_heading(gradi)`

Imposta l'orientamento **assoluto** in gradi.

```
set_heading(0);      /* verso destra */
set_heading(90);     /* verso il basso */
set_heading(180);    /* verso sinistra */
set_heading(270);    /* verso l'alto */
```

Pulire la tela

`clear_screen(colore)`

Cancella tutto il disegno e riempie la tela con il colore specificato.

```
clear_screen(BLACK); /* tela nera */
clear_screen(WHITE); /* tela bianca */
```

💡 **Nota:** `clear_screen` **non** riposiziona la tartaruga: la sua posizione e il suo orientamento restano invariati.

Chiudere il programma: `turtle_wait()`

`turtle_wait()` è **obbligatoria** alla fine del programma. Fa tre cose:

1. Entra in un ciclo di rendering a 60 FPS.
2. Mostra la tela e la tartaruga (un cerchio rosso).
3. Resta in attesa finché non chiudi la finestra (o premi ESC).

```
turtle_wait();    /* blocca finché non chiudi */
```

Dopo la chiusura, libera la memoria della tela e chiude la finestra.

Importante: se non chiami `turtle_wait()`, il programma termina immediatamente e non vedrai nulla!

Esempi completi

Esempio 1: triangolo equilatero

```
#include "turtle.h"

int main(void)
{
    turtle_init(600, 600, "Triangolo");

    for (int i = 0; i < 3; i++) {
        forward(200);
        turn_left(120);    /* angolo esterno di un triangolo equilatero */
    }

    turtle_wait();
    return 0;
}
```

Esempio 2: poligono regolare generico

```
#include "turtle.h"

int main(void)
{
    turtle_init(800, 800, "Poligono");

    int lati = 7;
    float lato = 150.0f;
    float angolo = 360.0f / lati;

    for (int i = 0; i < lati; i++) {
        forward(lato);
        turn_right(angolo);
    }

    turtle_wait();
    return 0;
}
```

Prova a cambiare `lati` a 3, 5, 6, 8, 12... e osserva come cambia la figura.

Esempio 3: stella a 5 punte

```
#include "turtle.h"

int main(void)
{
    turtle_init(600, 600, "Stella");

    set_color(YELLOW);
    for (int i = 0; i < 5; i++) {
        forward(200);
        turn_right(144);    /* 180 - 36 */
    }

    turtle_wait();
    return 0;
}
```

Esempio 4: quadrato senza penna (spostamento)

```
#include "turtle.h"

int main(void)
{
    turtle_init(800, 600, "Salto");

    /* Primo quadrato */
    set_color(RED);
    for (int i = 0; i < 4; i++) {
        forward(100);
        turn_right(90);
    }

    /* Spostati senza disegnare */
    pen_up();
    forward(150);
    pen_down();

    /* Secondo quadrato, altro colore */
    set_color(BLUE);
    for (int i = 0; i < 4; i++) {
        forward(100);
        turn_right(90);
    }

    turtle_wait();
    return 0;
}
```

Esempio 5: fiocco di neve di Koch

```
#include "turtle.h"

void koch(float lunghezza, int livello)
{
    if (livello == 0) {
        forward(lunghezza);
    } else {
        koch(lunghezza / 3, livello - 1);
        turn_left(60);
        koch(lunghezza / 3, livello - 1);
        turn_right(120);
        koch(lunghezza / 3, livello - 1);
        turn_left(60);
        koch(lunghezza / 3, livello - 1);
    }
}

int main(void)
{
    turtle_init(900, 600, "Fiocco di Koch");
    set_color(SKYBLUE);

    /* Posiziona la tartaruga per centrare il fiocco */
    go_to(100, 450);
    set_heading(0);

    /* Tre lati del fiocco */
    for (int i = 0; i < 3; i++) {
        koch(700, 4);
        turn_right(120);
    }

    turtle_wait();
    return 0;
}
```

Esempio 6: albero binario ricorsivo

```
#include "turtle.h"

void albero(float lunghezza, int profondita)
{
    if (profondita == 0) return;

    forward(lunghezza);

    turn_left(30);
    albero(lunghezza * 0.7f, profondita - 1);

    turn_right(60);
    albero(lunghezza * 0.7f, profondita - 1);

    turn_left(30);
    backward(lunghezza);
}

int main(void)
{
    turtle_init(800, 800, "Albero");
    set_color(GREEN);

    go_to(400, 750);      /* base dell'albero */
    set_heading(270);     /* punta verso l'alto */

    albero(150, 8);

    turtle_wait();
    return 0;
}
```

Esempio 7: spirale colorata

```
#include "turtle.h"

int main(void)
{
    turtle_init(800, 800, "Spirale");

    Color colori[] = { RED, ORANGE, YELLOW, GREEN, BLUE, VIOLET };

    for (int i = 0; i < 100; i++) {
        set_color(colori[i % 6]);
        forward(i * 3.0f);
        turn_right(59);    /* angolo quasi primo per una spirale */
    }

    turtle_wait();
    return 0;
}
```

Riferimento API

void turtle_init(int width, int height, const char* title)

Inizializza la finestra e la tartaruga.

- width, height: dimensioni della finestra in pixel.
- title: titolo della finestra.
- La tartaruga parte al centro, angolo 0° (verso destra), penna abbassata, colore verde.
- Lo sfondo della tela è nero.
- Se chiamata più volte, non fa nulla (protegge da doppia inizializzazione).

void forward(float distance)

Avanza di distance pixel nella direzione corrente. Se la penna è abbassata, disegna.

void backward(float distance)

Indietreggia di distance pixel. Equivale a forward(-distance).

void turn_right(float degrees)

Gira a destra (orario) di degrees gradi. Rotazione **relativa**.

void turn_left(float degrees)

Gira a sinistra (antiorario) di degrees gradi. Rotazione **relativa**.

void pen_down(void)

Abbassa la penna (disegna).

void pen_up(void)

Alza la penna (non disegna).

void set_color(Color color)

Cambia il colore della penna.

void go_to(float x, float y)

Sposta la tartaruga in (x, y) senza disegnare.

void set_heading(float degrees)

Imposta l'orientamento assoluto in gradi.

void clear_screen(Color background_color)

Cancella la tela e la riempie con background_color.

void turtle_wait(void)

Entra nel ciclo di rendering e mantiene la finestra aperta finché non viene chiusa. Da chiamare **alla fine** del programma.

Domande frequenti

D: Posso includere turtle.h in più file .c? No. Contiene variabili static: includilo in **un solo** file .c.

D: Perché la tartaruga non si vede? Probabilmente non hai chiamato turtle_wait(). Senza di essa il programma termina subito dopo i comandi e la finestra si chiude.

D: Perché l'angolo 90° punta verso il basso e non verso l'alto? Perché le coordinate dello schermo hanno l'asse Y che punta verso il basso. È la convenzione della grafica, non della matematica.

D: Come faccio a disegnare senza lasciare scia? Chiama pen_up() prima di muoverti, poi pen_down() per ricominciare a disegnare.

D: Come faccio a spostarmi senza disegnare? Usa go_to(x, y) (spostamento istantaneo) oppure pen_up(); forward(...); pen_down();.

D: Posso cambiare la velocità del disegno? No, la tartaruga disegna istantaneamente. Il disegno appare tutto insieme all'apertura della finestra. Se vuoi un'animazione, dovresti modificare la libreria per aggiungere un'animazione tra un comando e l'altro.

D: Come faccio a cambiare il colore di sfondo? Usa clear_screen(colore) all'inizio del programma:

```
turtle_init(800, 600, "Titolo");  
clear_screen(WHITE);
```

D: Perché la tartaruga è un cerchio rosso? È solo un indicatore visivo della posizione e direzione. Non è configurabile senza modificare la libreria.

D: Posso salvare il disegno su file? Non direttamente. Dovresti modificare turtle_wait() per usare TakeScreenshot("file.png") di Raylib.

D: Come faccio a disegnare un cerchio? La tartaruga disegna solo linee. Per approssimare un cerchio, usa tanti piccoli segmenti:

```
for (int i = 0; i < 360; i++) {  
    forward(2);  
    turn_right(1);  
}
```

D: Posso usare turtle.h senza Raylib? No, dipende da Raylib.

D: La finestra non risponde durante l'esecuzione. Perché? I comandi della tartaruga (forward, turn_right, ecc.) modificano lo stato e disegnano sulla tela virtuale, ma **non** aggiornano la finestra. Solo turtle_wait() mostra il risultato. È una scelta progettuale: prima disegni tutto, poi mostri.

Concetti didattici

`turtle.h` è pensato per insegnare:

Concetto

Cicli

Funzioni

Ricorsione

Geometria

Modularità

Come si applica

Ripetere `forward + turn` per poligoni

Incapsulare pattern (es. `quadrato(lato)`)

Frattali (Koch, albero, Sierpinski)

Angoli, poligoni, coordinate

Separare logica di disegno e logica di controllo

Suggerimento:

Invece di scrivere ogni volta il ciclo del quadrato, crea una funzione:

```
void quadrato(float lato) {
    for (int i = 0; i < 4; i++) {
        forward(lato);
        turn_right(90);
    }
}

void poligono(int lati, float lato) {
    float angolo = 360.0f / lati;
    for (int i = 0; i < lati; i++) {
        forward(lato);
        turn_right(angolo);
    }
}
```

Poi usale:

```
set_color(RED);
quadrato(100);

pen_up();
forward(150);
pen_down();

set_color(BLUE);
poligono(6, 80);
```

Conclusione

`turtle.h` è un ponte tra la programmazione e la geometria: permette di vedere concretamente il risultato di cicli, angoli e ricorsioni. La filosofia è:

Prima disegni, poi mostri.

Sperimenta, prova pattern diversi, combina colori e forme. La tartaruga è il tuo pennello: buon divertimento!

