BRICKS | TEMA

Grafica per l'inclusione

Fare grafica con e senza coding come strategia per lezioni inclusive e per aumentare la consapevolezza degli studenti verso il mondo digitale

a cura di:
Andrea Ferraresso

Inclusione, Grafica, 3D, Coding, Scratch, Python

Premessa

In questo articolo spiegherò come, da docente di Laboratorio di ICT al biennio di un liceo paritario, ho avuto modo di sperimentare quanto "fare grafica" al computer possa diventare strumento di inclusione.

Per inclusivo intendo che ogni studente può partecipare alle attività (spesso piccole "sfide") proposte in classe esprimendo la propria **creatività** e potendo scegliere l'approccio e il grado di difficoltà più adatti.

Partendo dalla storia dell'arte

Una delle principali fonti di ispirazione è stata la lettura di *Figure* di Riccardo Falcinelli. In questo libro l'autore cerca di spiegare il motivo per il quale alcune immagini diventano più famose di altre.

Falcinelli non si sofferma tanto sul simbolismo dell'opera quanto sul come è stata progettata e costruita. Si concentra, insomma, sul "meccanismo" che sta alla base dell'immagine.

Dal mio punto di vista, l'autore si dedica al ***reverse engineering*** della composizione delle opere trattate, rendendo estremamente piacevole, direi appassionante, lo studio dell'arte e accendendo nel lettore la voglia di "provarci".

Tenendo presente che mai come in questo periodo storico gli studenti vivono immersi nella cosiddetta **"società dell'immagine"** (pensiamo solo al fatto che tra giovani è molto più popolare Instagram che Facebook), ho sperimentato con i miei studenti diverse attività legate alla produzione di immagini digitali.

L'idea di fondo è stata quella di fare vedere ai ragazzi il *making of* di molte immagini che hanno intorno (copertine di libri, *post* sul *social network*, videogiochi ecc.).

Il tutto facendoli lavorare con strumenti gratuiti e caratterizzati da una curva di apprendimento accettabile.

In sintesi, un approccio allo studio dell'informatica effettuato a partire dall'aspetto grafico che, oltre ad essere inclusivo, presenta come positivo risvolto il fatto di rendere gli studenti **più consapevoli** del mondo digitale in cui sono immersi.

Rappresentazione digitale delle immagini

Come noto, nel mondo della computer grafica, le immagini vengono descritte usando la tecnica della grafica raster o quella della grafica vettoriale.

Nella **grafica raster** (o bitmap) l'immagine viene memorizzata come una matrice (una griglia) in cui a ogni elemento (pixel) è associato un colore, a sua volta definito da valori numerici.

Nella **grafica vettoriale**, invece, l'immagine finale è costituita da un insieme di primitive geometriche (punti, linee diritte e curve, forme...).

Semplificando, possiamo individuare due modalità di produrre immagini al computer:

- la prima, più nota, è quella di utilizzare **software applicativi** per l'elaborazione grafica;
- la seconda è attraverso il **coding**, cioè la programmazione.

Nel seguito dell'articolo presenterò le attività più rilevanti proposte da me in classe o sviluppate dagli studenti, suddivise per modalità di produzione.

Linee rette e curve con Scratch

Scratch (https://scratch.mit.edu/) è un ambiente di programmazione visuale basato su blocchi grafici sviluppato al **MIT Media Lab** e reso disponibile a partire dal 2004.

Considerato una delle possibili evoluzioni del linguaggio **Logo**, Scratch basa la sua facilità di utilizzo sull'intuizione del ricercatore Andrew Begel, che nel 1996 presentò LogoBlock, un linguaggio di programmazione grafico pensato per il mattoncino programmabile della Lego.

Scratch è anche un potente strumento **costruzionista** (con riferimento a Seymour Papert) che consente, come ben spiegato da Mitchel Resnick e David Siegel nell'articolo *A Different Approch To Coding*, di imparare a programmare per organizzarsi, esprimersi e condividere le proprie idee.

Come noto, con Scratch è possibile costruire storie animate, ma è anche possibile disegnare, sia usando l'**editor grafico** incorporato, sia sfruttando i comandi della categoria **"Penna"** , che ricordano da vicino la caratteristica più famosa del linguaggio Logo.

Grazie a Scratch è piuttosto semplice tracciare il perimetro di **poligoni regolari** (Figura 1). Il codice mostrato tocca più di un concetto del **pensiero computazionale** con Scratch (sequenze, ripetizioni, dati).

Tra l'altro, l'uso delle **variabili** mette in evidenza la possibilità, grazie allo studio degli angoli, di implementare un algoritmo di disegno valido per tutti i poligoni regolari.

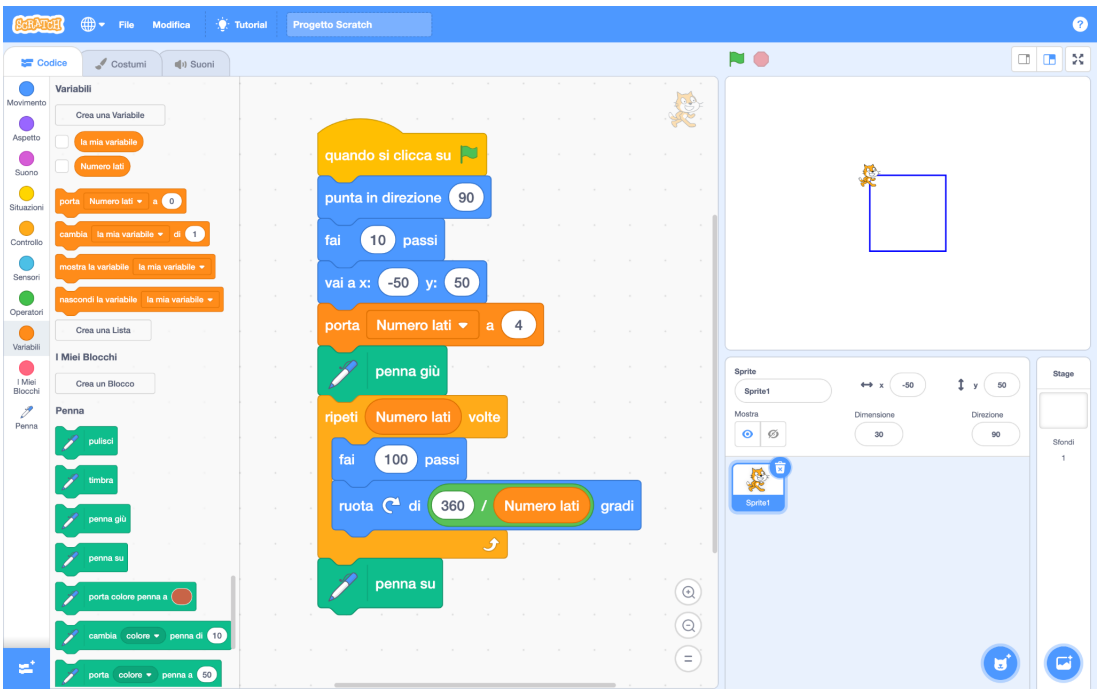

Figura 1 - Programma per disegnare un quadrato

Scratch offre la possibilità di vedere subito il risultato della propria attività di coding, pertanto è possibile sperimentare nuove soluzioni anche se si ha poco tempo.

In Figura 2, ad esempio, vediamo come creare immagini interessanti semplicemente rendendo variabile la dimensione del lato del quadrato e ripetendo più volte la rotazione dello sprite che disegna.

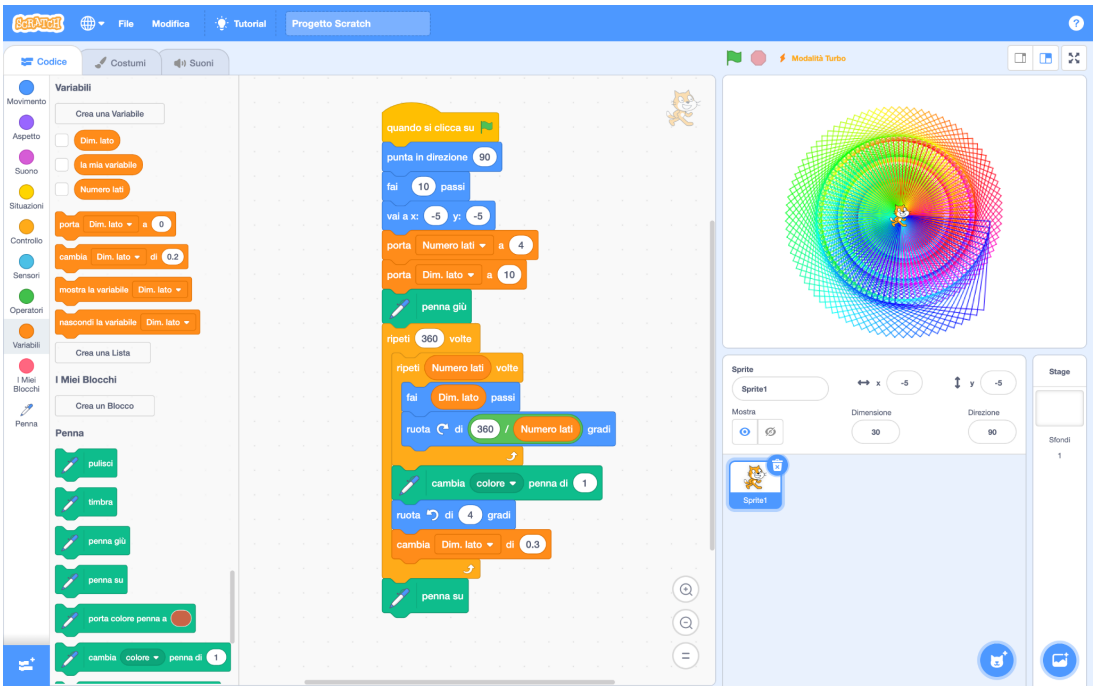

Figura 2 - Replica del quadrato con variazione ogni volta del lato e dell'angolo

Scratch permette di programmare singolarmente il comportamento di più di uno sprite. Se facciamo andare su e giù un gatto per il lato destro dello schermo, il risultato può non sembrare molto interessante (Figura 1).

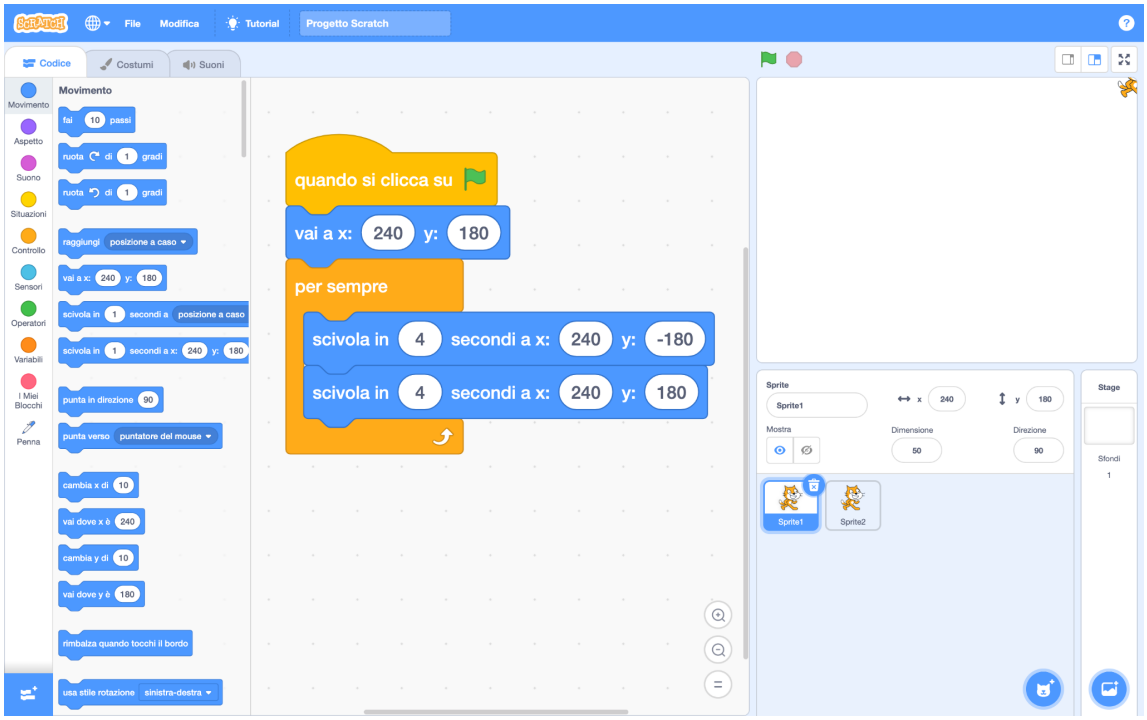

Figura 3 - Movimento di un gattino sullo schermo

Ma se aggiungiamo dieci gattini che lo "inseguono" e nel frattempo lasciano una traccia, allora potremmo ottenere in maniera semplice un'immagine altrimenti difficilmente realizzabile senza ricorrere a conoscenze matematiche non banali (Figura 4), che sono comunque applicabili in Scratch (con riferimento alle funzioni trigonometriche).

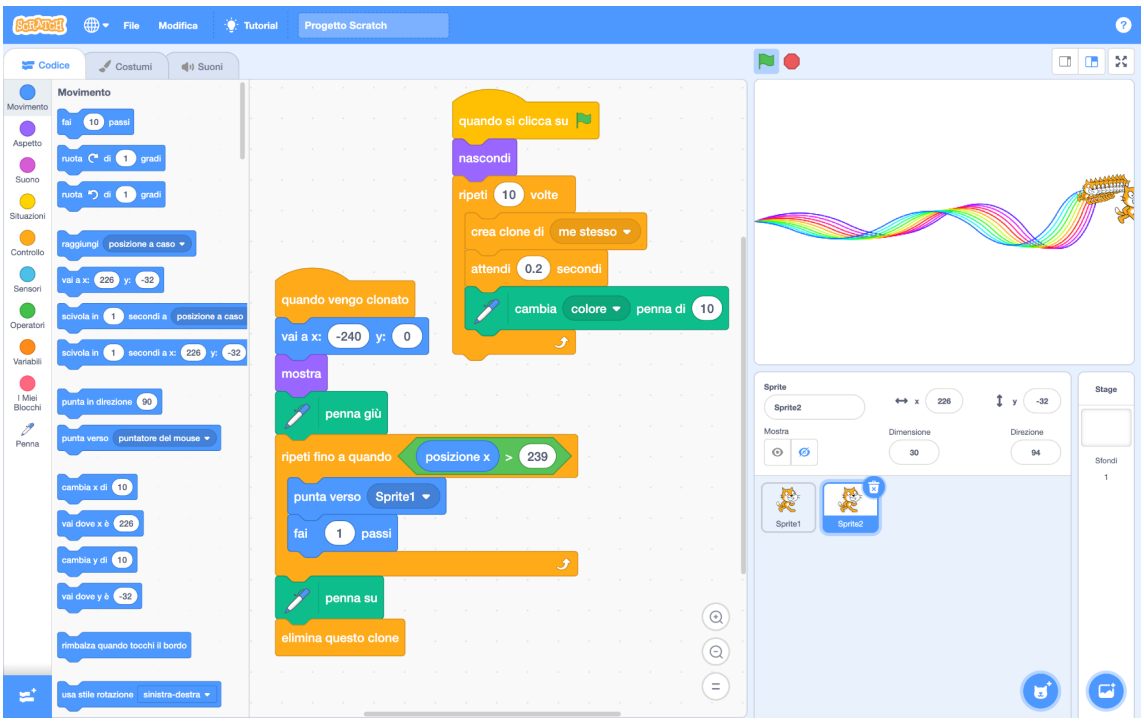

Figura 4 - Clonando il gattino

Alternative a Scratch

Se Scratch può risultare troppo "da bambini", una valida alternativa è costituita da **Snap!** (https://snap.berkeley.edu/), una piattaforma di coding derivata da Scratch e maggiormente orientata allo studio della *computer science*, grazie a delle caratteristiche più avanzate soprattutto per quanto riguarda la gestione delle variabili (Figura 5).

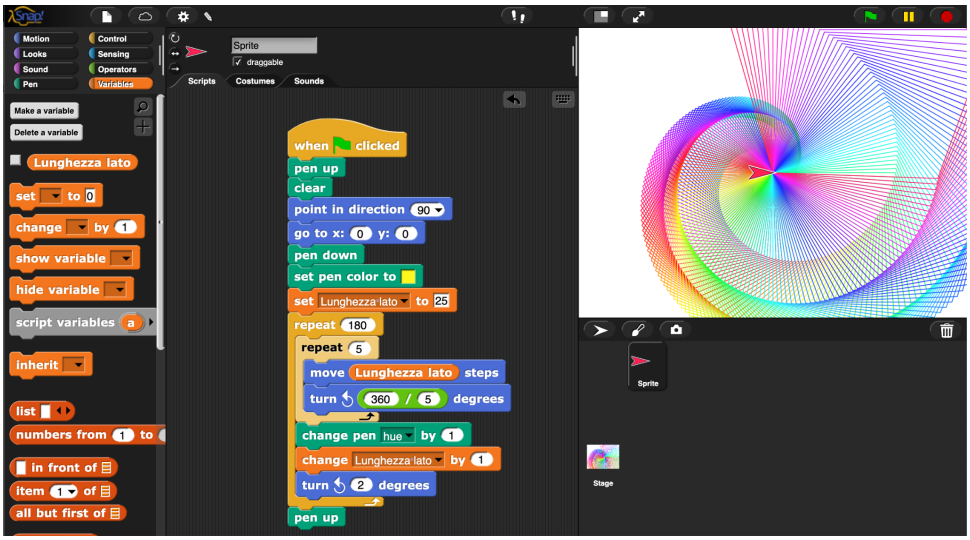

Figura 5 - Programma con Snap!

Il tuo nome in Python

Python (https://www.python.org/) è un linguaggio di programmazione sviluppato a partire dal 1991, ma solo negli ultimi anni si è diffuso notevolmente anche in ambito scolastico.

Lo troviamo inserito anche in **mBlock** (https://mblock.makeblock.com/en-us/) come modalità di programmazione di robot e altri device in alternativa ai blocchi in stile Scratch.

Nella libreria standard di Python, che raggruppa le funzionalità incluse nel pacchetto di installazione principale, troviamo **Turtle Graphics**.

Le funzioni grafiche disponibili sono chiaramente ispirate ai comandi di disegno del linguaggio Logo (Figura 6).

```
from turtle import *
color('blue', 'yellow')
begin_fill()
for i in range(0, 9):
    forward(300)
    left(200)
end_fill()
done()
```

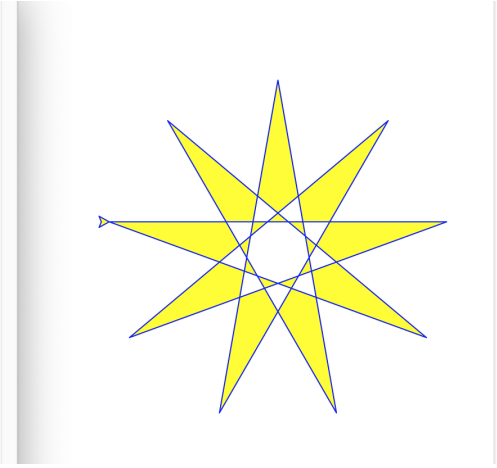

Figura 6 - Disegno di una stella con Python

Dopo aver imparato il funzionamento base di Turtle Graphics, è interessante cercare di sfruttarlo per scrivere il proprio nome, adattando quanto appreso per realizzare una sintesi grafica delle lettere che lo compongono (Figura 7).

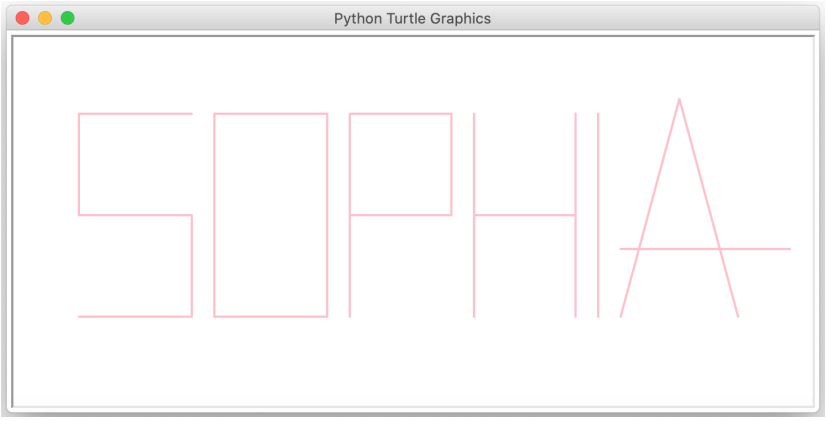

Figura 7 - Scrivere il nome con Turtle graphics

Alcuni studenti potrebbero arrivare a proporre soluzioni totalmente parametrizzate, quindi anche piuttosto sofisticate a livello di programmazione (Figura 8).

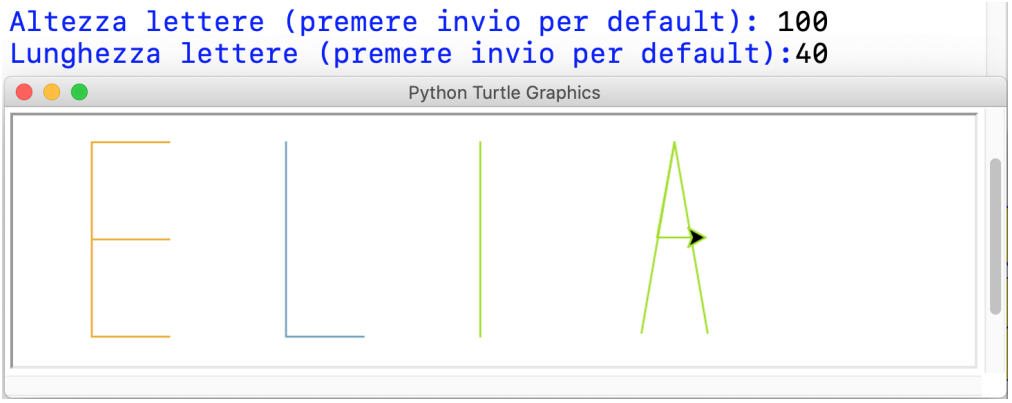

Figura 8 - Uso di parametri

Python e videogiochi

Sempre con Python, grazie alla raccolta di moduli denominata **Pygame**, è possibile creare videogiochi amatoriali anche di una certa complessità.

Uno degli effetti grafici più diffusi è lo scorrimento dello schermo, in cui l'area complessiva del gioco è più grande dell'area visibile.

Il solo fatto di dover realizzare un piccolo videogioco con uno strumento differente da Scratch, pone gli studenti di fronte alla necessità di **suddividersi i compiti**.

Chi cura l'aspetto grafico, sempre coordinandosi con il programmatore, dovrà utilizzare un programma per la grafica raster quale Photoshop o **GIMP** (che è gratuito e *opensource*).

La realizzazione di sfondi in grado di scorrere all'infinito pone l'ulteriore sfida di creare immagini "ripetibili" (Figura 9).

```
import pygame
import os
import sys

SCREEN_WIDTH = 480 # Larghezza dello s
SCREEN_HEIGHT = 480 # Altezza dello s

hero_x = 0
scroll_x = 0

# INIZIO PROGRAMMA
pygame.init()

screen = pygame.display.set_mode((SCRI

# Immagine di background
# convert() crea una copia dell'immag
# per mostrarla più rapidamente sullo
bg = pygame.image.load("./pavimento.p

# Orologio di Pygame
clock = pygame.time.Clock()

# Numero massimo di fotogrammi per secondo
MAX_FPS = 30
```

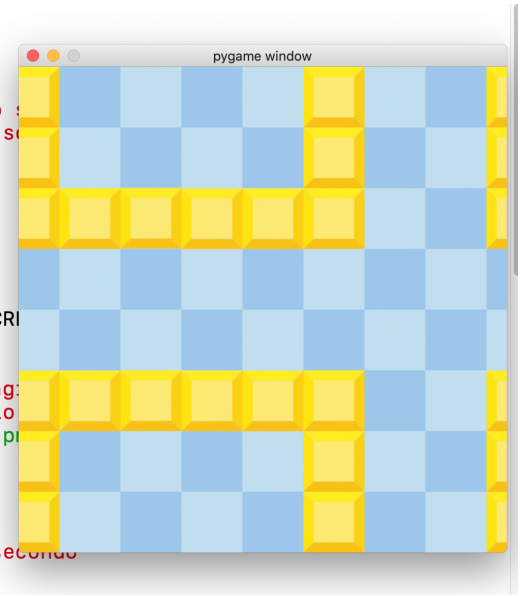

Figura 9 - Realizzazione di uno sfondo

L'Agenda 2030 con Canva

Canva (https://www.canva.com/) è una piattaforma per la progettazione grafica, gratuita nella versione base e con tariffe agevolate per scuole e per il mondo non profit. Può essere utilizzata per produrre grafica da visualizzare a video (soprattutto per i **social media**) o da stampare su carta.

Una delle possibili applicazioni in classe è proporre agli studenti di imparare a usarlo (*learning by doing*) creando un semplice *post* su uno degli obiettivi dell'Agenda 2030.

Si tratta di un'attività volutamente **interdisciplinare** che permette loro di far emergere ed esprimere le proprie idee in merito a tematiche rilevanti (Figura 10).

Figura 10 - Un esercizio sull'Agenda 2030

Soluzioni che sfruttano la modellazione 3D

Tinkercad (https://www.tinkercad.com/) è una piattaforma online per la progettazione grafica 3D, che nel tempo si è arricchita con una sezione dedicata al coding e una dedicata alla simulazione elettronica.

Benché non sia uno strumento per la modellazione tridimensionale ai fini del **rendering**, è molto più semplice da utilizzare rispetto ad applicazioni come Blender.

La possibilità di **estrudere** figure geometriche piane precedentemente create in grafica raster o vettoriale (magari con Inkspace) consente di creare con facilità immagini grafiche interessanti (Figura 11).

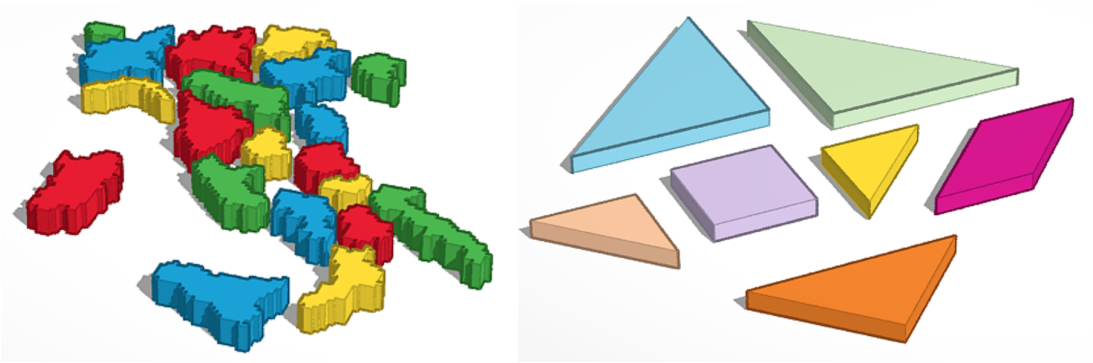

Figura 11 - Estrusione di figure piane con Tinkercad

Capire i colori

La rappresentazione di immagini in dispositivi elettronici avviene solitamente tramite il **modello colore RGB**, dove ciascuna tonalità di colore è data dalla "somma" dei tre colori rosso (Red), verde (Green) e blu (Blue).

Per capire meglio come funziona questo modello può essere utile far utilizzare la simulazione "Visione dei Colori" (Figura 12) presente in **PhET** (https://phet.colorado.edu/).

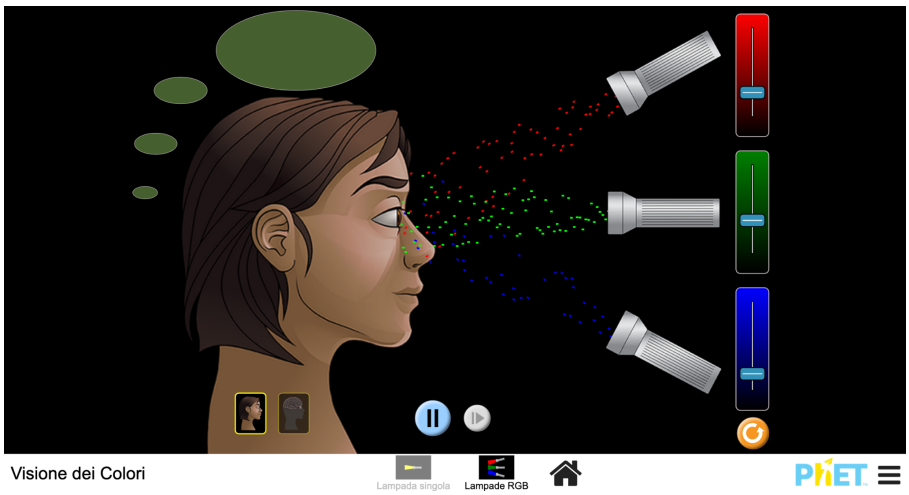

Figura 12 - Come funziona il modello colore RGB

Per generare **palette di colori** esistono invece numerosi *tool* gratuiti *online*, tra cui Adobe Color (https://color.adobe.com/it/create/color-wheel).

Da notare che la rappresentazione delle tonalità di colore usando la **notazione esadecimale** (Figura 13) apre la strada ad attività informatiche legate al cambio di base (passaggio da decimale a binario e infine a esadecimale).

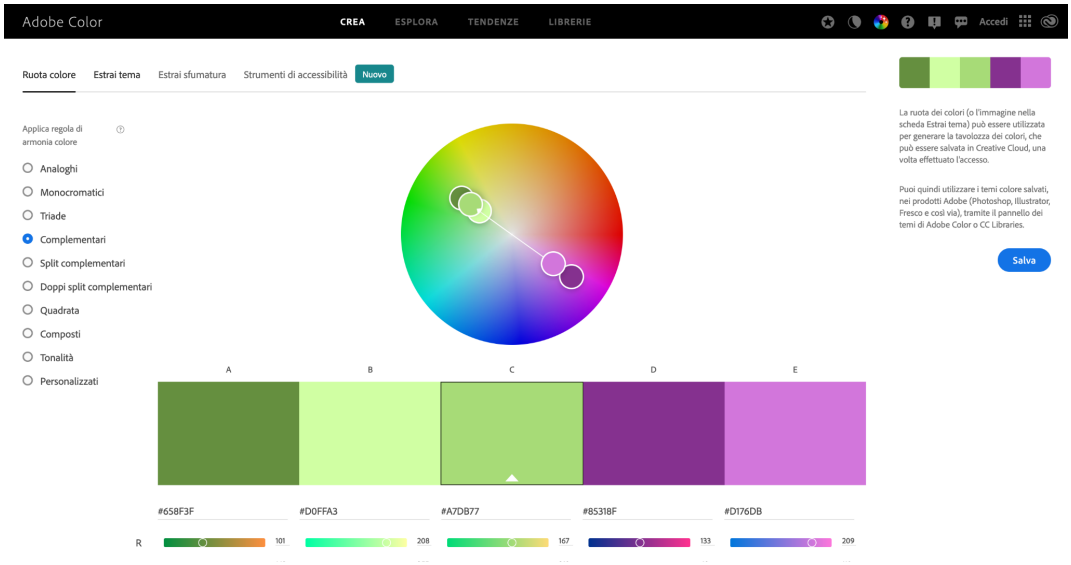

Figura 13 - I colori in formato esadecimale

Conclusioni

Quando si organizzano laboratori di ICT alla scuola secondaria di secondo grado, si può essere tentati a spingere sull'acceleratore, cercando magari di insegnare fin da subito le basi di un linguaggio di programmazione "serio", trattando algoritmi di ordinamento e strutture di dati e tutto quello che ci dovrebbe essere in un corso di programmazione informatica "tradizionale".

In realtà, ci si accorge in fretta che in molti casi gli studenti non si appassionano se non vengono coinvolti in attività in cui possano esprimersi senza dover attendere troppe ore di lezioni "introduttive".

Non raramente, poi, mancano alcune abilità legate al **pensiero computazionale** (ad esempio la capacità di scomporre i problemi o quella di individuare le sequenze che si ripetono). Ciò rende veramente difficoltoso il percorso di apprendimento, anche per il docente.

Concentrarsi, quindi, sull'aspetto grafico può dar vita ad attività motivanti e inclusive, poiché come abbiamo visto esistono molte maniere di produrre grafica al computer.

Ogni studente può trovare l'approccio più congeniale e non è raro che chi comincia interessandosi a programmi di grafica poi voglia capire meglio che cosa succede "dietro le quinte" e imparare quindi a programmare.

Va anche detto che negli esempi riportati in questo articolo abbiamo visto solo una piccola parte degli strumenti disponibili.

Per completezza andrebbero citati almeno altre altre due soluzioni utilizzabili in classe:

- Processing (https://processing.org/), un linguaggio di programmazione l'arte generativa («arte che crea arte»);
- Three.js (https://threejs.org/), una libreria JavaScript per il coding creativo.

Esiste poi tutto il mondo legato ai dati (***data visualisation*** e ***data storytelling***), dove anche in questo caso si può lavorare da "programmatori" (esistono numerose librerie per Python e JavaScript) o utilizzando uno dei tanti *tool* disponibili.

Andrea Ferraresso

andreaferraresso@gmail.com

Consulente informatico, alla fine del 2013 fonda il CoderDojo Fossò-Venezia e successivamente l'associazione non-profit Digiveneto, che da anni collabora con il polo scientifico dell'Università Ca' Foscari Venezia organizzando laboratori gratuiti per i più giovani.
Autore di libri su coding, robotica educativa e tecnologia per la scuola primaria e secondaria di primo grado., svolge attività di formatore per docenti su temi STEM/STEAM.
È docente di laboratori di informatica in due licei scientifici paritari di Padova.