Lorenzo Rossi, matr. 20031485
Università degli Studi del Piemonte Orientale
Dipartimento di Scienze ed Innovazione Tecnologica
Corso di Laurea Magistrale in Informatica
Anno Accademico 2020/2021

# Relazione per il progetto del corso di Sistemi Multimediali

Giugno 2021

# 1 Introduzione

Il progetto relativo al corso di Sistemi Multimediali ha consistito nella realizzazione di tre applicativi in Java che permettessero l'elaborazione di immagini secondo il principio del *compositing*, cioé la combinazione di elementi visivi provenienti da sorgenti diverse in una singola immagine risultante.

In particolare, è stato richiesto di realizzare delle classi che implementassero le seguenti tecniche:

- **Alpha Blending:** tecnica con cui due immagini vengono sovrapposte (previa assegnazione di un parametro $\alpha$ di trasparenza a ciascuna di esse);
- **Chroma Keying:** tecnica con cui i pixel di una certa gradazione cromatica (tipicamente verde e blu) di un'immagine in primo piano vengono sostituiti con i pixel dell'immagine designata come sfondo (principio utilizzato nella tecnica del *green screen*);
- **3D Keying (Primatte Keying):** ipotizzando che il *3D Color Cube* sia un volume cubico discreto di cui ogni lato misura da 0 a 255 e nel quale un punto (e quindi un colore RGB) è individuato dalla terna cromatica K = (R, G, B), allora è possibile definire al suo interno due sfere concentriche S1 ed S2, centrate in K e con raggio rispettivamente R1 ed R2. Le sfere sono tali che ogni pixel dell'immagine risultante dalla sovrapposizione delle due immagini sorgenti corrisponderà al pixel di background se il suo valore ricade all'interno del volume S1, ad una sovrapposizione dei pixel di foreground e background se ricade nel volume S2 - S1 ed al pixel di foreground se ricade all'esterno di S2.

# 2 Struttura del progetto

## 2.1 `sources`

Directory contenente le immagini sorgente utilizzate per il *compositing*. In particolare, la directory `sources` contiene le seguenti immagini:

- `sample-1.jpg`;
- `sample-2.jpg`;
- `sample-fg.jpg`;
- `sample-bg.jpg`;

## 2.2 `results`

Directory contenente le immagini risultanti dal *compositing*. I nomi dei file risultanti saranno dei seguenti tipi, con i valori utilizzati per la computazione al posto delle parentesi quadre:

- `alpha-blend-aF[0..10]-aB[0..10]-result.jpg`: risultato della tecnica di *Alpha Blending* con $\alpha_{fg}$ e $\alpha_{bg}$ rappresentati rispettivamente nel filename da `aF` ed `aB`, entrambi tra 0 ed 1 (moltiplicato per 10 per comodità di utilizzo dei filename);
- `chroma-key-[blue|green]-result.jpg`: risultato della tecnica di *Chroma Keying*, dove `[blue|green]` indica la chiave utilizzata per il processamento dell'immagine risultante;
- `primatte-key-R[r]G[g]B[b]-R1-[r1]-R2-[r2]-result.jpg`: risultato della tecnica di *3D Keying*, dove i valori della terna RGB seguono il colore a cui fanno riferimento (con `r`, `g` e `b` tra 0 e 255), seguiti dai raggi `r1` ed `r2` scelti per S1 ed S2.

## 2.3 `src.project`

Package contenente la classe astratta `Compositing` e tre sottopackage, ognuno dei quali ospita una classe omonima ed una classe `Test`, nella quale risiede un metodo `main` con cui è possibile testare le funzionalità dei packages.

I sottopackage contenuti in `src.project` sono i seguenti:

- `alphablending`
- `chromakeying`
- `primattekeying`

# 3 Scelte implementative

## 3.1 Compositing

Classe astratta presente nel package `src.project`, fornisce campi `protected` in cui memorizzare le immagini sorgenti e l'immagine risultante e metodi utili per caricare, visualizzare e salvare le immagini sorgenti e l'immagine elaborata.

La classe fornisce il metodo astratto `perform()`, da implementare nelle classi che ereditano da `Compositing`.

## 3.2 AlphaBlending

Classe contenuta nel package `src.project.alphablending` che eredita da `src.project.Compositing` ed implementa il metodo `perform()` per operare l'*Alpha Blending* di due immagini sorgenti. Questa classe fornisce anche dei metodi per impostare i valori di $\alpha$ dell'immagine *foreground* e dell'immagine *background*, e per ricavare il valore di $\alpha$ da quelli assegnati alle immagini sorgenti, da utilizzare nel corpo del metodo `perform()`.

## 3.3 ChromaKeying

Classe contenuta nel package `src.project.chromakeying` che eredita da `src.project.Compositing` ed implementa il metodo `perform()` per operare il *Chroma Keying* di due immagini sorgenti. Questa classe fornisce metodi per ricavare il valore del parametro $\alpha$, da utilizzare nel corpo di `perform()`, a seconda del valore cromatico dei pixel delle immagini sorgenti. `ChromaKeying` fornisce anche metodi per impostare il colore chiave ("blue" o "green") su cui operare il *Chroma Keying*.

## 3.4 PrimatteKeying

Classe contenuta nel package `src.project.chromakeying` che eredita da `src.project.Compositing` ed implementa il metodo `perform()` per operare il *3D Keying* di due immagini sorgenti. Questa classe fornisce i metodi per impostare i parametri per realizzare il *3D Keying*, tra cui i metodi per impostare il centro `K` ed i raggi `r1` ed `r2` delle sfere concentriche centrate nel punto `K` dell'*RGB Color Cube*.

È qui presente anche una classe statica `Point`, con cui viene rappresentato il punto `K` tramite le sue coordinate RGB, la quale fornisce il metodo statico `distance()` che restituisce la distanza tra due oggetti di tipo `Point`, da utilizzare all'interno del corpo del metodo `perform()` per poter calcolare il corretto valore di $\alpha$ per ogni pixel dell'immagine risultante, sintetizzandolo dai pixel delle immagini sorgenti.

# 4 Risultati

## 4.1 Alpha Blending

Sovrapposizione di figura 1 con $\alpha_{fg} = 0.5$ e figura 2 con $\alpha_{bg} = 1.0$ tramite tecnica *Alpha Blending*, risultante in figura 3.

Figura 1: Immagine in *foreground*.

Figura 2: Immagine in *background*.

Figura 3: Immagine risultante.

## 4.2 Chroma Keying

Sovrapposizione delle figure 4 e 5 con figura 6 tramite la tecnica del *Chroma Keying*, risultante rispettivamente nella figura 7 e nella figura 8.

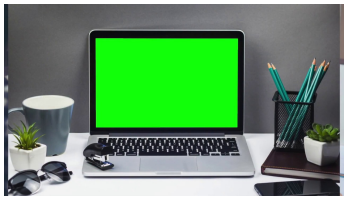

Figura 4: Immagine *foreground* (chiave *green*).

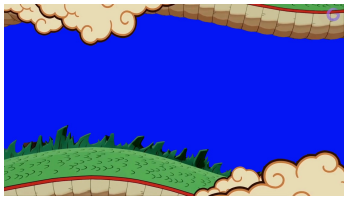

Figura 5: Immagine *foreground* (chiave *blue*).

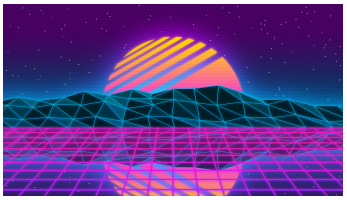

Figura 6: Immagine di *background*.

Figura 7: Immagine risultante dal *Chroma Keying* (chiave *green*).

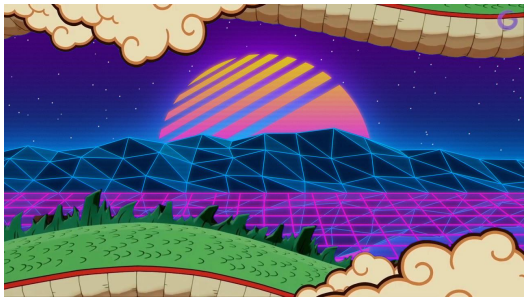

Figura 8: Immagine risultante dal *Chroma Keying* (chiave *blue*).

## 4.3 3D Keying

Sovrapposizioni delle figure 9 e 10 risultanti nelle figure 11, 12, 13, 14, 15 e 16, derivanti dalla variazione sia della terna cromatica K, sia dei raggi R1 ed R2 delle sfere concentriche S1 ed S2 centrate in K. Si noti che, per una terna cromatica K = (0, 200, 0), spostata quindi sulla gradazione verde, l'effetto risultante di figura 11 è sostanzialmente identico a quello ottenuto con il *Chroma Keying* in figura 7, ma in questo caso l'immagine di sfondo ha un colore più fedele a quello dell'immagine sorgente (in figura 7 ed 8 permane un alone del colore della chiave scelta).

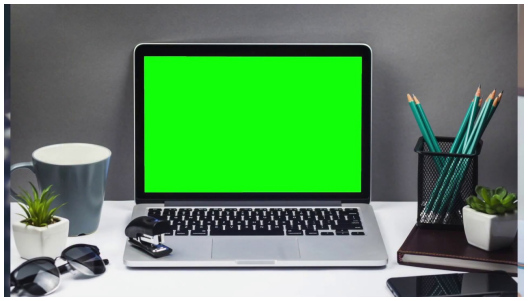

Figura 9: Immagine in *foreground*.

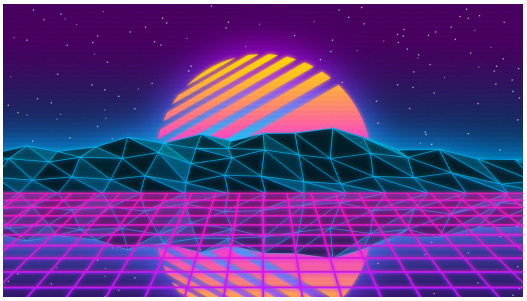

Figura 10: Immagine in *background*.

Figura 11:
RGB = (0, 200, 0),
R1 = 50, R2 = 100.

Figura 12:
RGB = (50, 50, 50),
R1 = 100, R2 = 200.

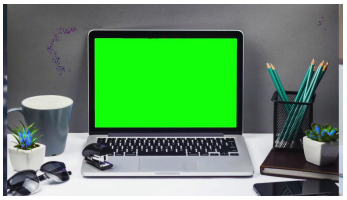

Figura 13:
RGB = (100, 100, 100),
R1 = 0, R2 = 100.

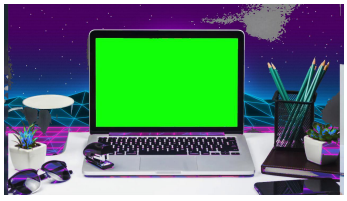

Figura 14:
RGB = (100, 100, 100),
R1 = 50, R2 = 100.

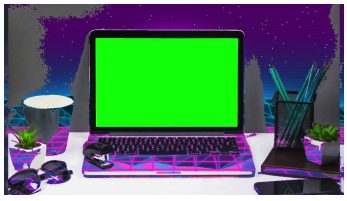

Figura 15:
RGB = (150, 150, 150),
R1 = 100, R2 = 200.

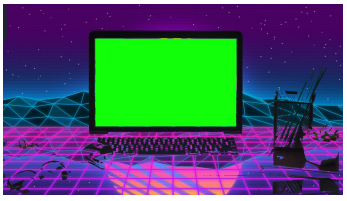

Figura 16:
RGB = (150, 150, 150),
R1 = 190, R2 = 200.