Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria del Cinema e dei Mezzi di Comunicazione
A.A. 2022/2023
Sessione di Laurea Marzo-Aprile 2023

Unire Dati e Video: Ricerca sull'Ottimizzazione della Produzione Automatizzata in Formato Lottie di Video in Motion Graphics.

Relatore
Prof. Giovanni Malnati
Co-Relatore
Prof. Matteo Ruffinengo
Tutor Aziendale
Luca Gonnelli

Candidata
Mara Salazzari
matr. s278147

Abstract

L'ambiente digitale nel quale si è immersi si sostiene sempre di più sul pilastro comunicativo del video: questo ha la capacità di catturare e sostenere l'attenzione di uno spettatore, più di quanto siano capaci altri strumenti di comunicazione quali immagini, newsletter o articoli. Il potere coinvolgente della musica e il piacevole effetto delle forme in movimento creano un prodotto di storytelling immediato ed avvincente.

Tra le diverse tipologie di video che possono essere prodotte ed analizzate ci si concentra sui video animati realizzati attraverso tecniche di motion graphics. Gli elementi di questi video - testi, forme, icone, disegni e diagrammi - interagiscono fra di loro e si evolvono nel tempo: sono strumenti che guidano l'occhio dell'osservatore attraverso lo spazio delle informazioni.

Le informazioni da comunicare originano spesso da un processo di raccolta di dati. Benché i dati possano apparentemente sembrare una collezione arida di numeri e lettere, questi nascondono dentro di sé una storia da comunicare, la storia di *chi* o *cosa* ha generato i dati, sia questo un singolo o una collezione di utenti, un evento nel mondo, uno specifico prodotto o servizio.

L'unione dei dati e dei video dà vita alla produzione di *data-driven* video e campagne di *video automation*: la produzione automatizzata di video personalizzati sulla base di specifici dati. La video automation permette dunque di portare un tocco personale ed una dimensione umana alla sterilità solo apparente dei dati e alla conformazione standard dei video. Diversificare un video in base a specifici dati permette di contestualizzarlo in un ambiente modellato sull'esperienza, sull'azione, sul pensiero di un utente, di un fenomeno o di un prodotto: i messaggi da convogliare diventano personali, efficaci e d'impatto.

Oggetto della trattazione è l'illustrazione di un processo di ricerca sulle metodologie di produzione automatizzata di video personalizzabili. La tesi è stata svolta come tassello conclusivo di un percorso di tirocinio svolto nell'azienda di *data visualization* e video automation Algo. Il focus - sia del tirocinio sia della tesi - è stato posto sull'ottimizzazione di un processo di produzione automatizzato di video attraverso l'utilizzo di Lottie, una libre-

ria open-source sviluppata da Airbnb che permette di renderizzare in real time, in un contesto web o mobile, animazioni realizzate in After Effects ed esportate tramite uno specifico plugin - Bodymovin - come un oggetto JSON, opportunamente conformato. È stata analizzata la struttura di un generico Lottie-JSON, e la relazione tra le informazioni contenute in esso e quelle impostate nel relativo progetto After Effects. È stata successivamente realizzata - e viene in questo contesto analizzata e descritta - una libreria di funzioni JavaScript allo scopo di velocizzare e facilitare il processo di personalizzazione di un generico video attraverso la manipolazione del Lottie-JSON che lo rappresenta. Infine, attraverso l'uso di questa libreria sono stati realizzati due progetti di video automation con la collaborazione di Algo, allo scopo di fornire un *tool* per la creazione di un'anteprima di un generico podcast. Viene analizzato il processo di produzione dei progetti, la metodologia di personalizzazione, in che modo la libreria è stata integrata in questo step, ed infine le prestazioni e i benefici da essa apportati.

Indice

Introduzione

Il lavoro e la ricerca discussi in questa testi sono stati prodotti nel contesto di un tirocinio svolto in Algo[1], uno studio di data visualization e video automation con sede a Torino. Algo nasce inizialmente come tech division di uno studio di design - Illo[2] - e si consolida successivamente come azienda separata. Il focus principale del loro lavoro è l'unione della motion graphics e dei dati. Algo infatti combina i mondi della *data science*, della programmazione e della motion graphics allo scopo di automatizzare la produzione di video data-driven. Si occupa di creare campagne video basate sui *dati*, siano essi provenienti da fenomeni esterni in costante evoluzione o da input specifici di un utente, e di fornire - se necessario - gli strumenti che permettono ai clienti la creazione dei video. Tra i prodotti finali vi sono video di data visualization, video AI-driven, esperienze web, progetti editoriali ed infografiche. Tra i clienti con i quali Algo ha lavorato vi sono esempi quali Bloomberg, Medici Senza Frontiere, The New York Times, Ferrero e molti altri prestigiosi nomi, sia italiani sia internazionali. Gli ambiti nei quali ha operato sono inoltre molteplici: sport, musica, finanza, *corporate*, campagne sociali e di sostenibilità. Collaborare con questa azienda ha permesso di poter svolgere un'interessante ricerca riguardo il ruolo della video automation nel panorama comunicativo odierno, e di poter lavorare sull'ottimizzazione dei processi di produzione della video automation, in modo tale da poter incontrare le esigenze tecniche e di design delineate dalle richieste dei clienti.

[1] Algo: https://algo.tv
[2] Illo: http://illo.tv

Capitolo 1

Animazione e Comunicazione: verso la Video Automation

1.1 Il ruolo del video nel digital landscape

Nel panorama digitale odierno nessun mezzo comunicativo ha lo stesso ruolo, la stessa importanza e la stessa diffusione del video. Il video, sempre di più, prevale su post testuali, immagini, newsletter ed articoli come veicolo di informazioni. È coinvolgente, avvincente, d'impatto e facilmente accessibile a chiunque abbia una connessione internet. Permette di poter comprendere ed immagazzinare informazioni in un processo rapido che non richiede lo stesso sforzo cognitivo della controparte testuale. Stimoli audiovisivi collaborano per creare un prodotto di storytelling che veicoli messaggi personali ed educativi in maniera piacevole e veloce. È più rapido e stimolante di un testo, è più coinvolgente di un *audio-only* podcast, e sembra essere un'ottima formula per poter catturare e trattenere l'attenzione di un utente nell'oceano di stimoli in cui è costantemente immerso.

Un video dà la possibilità a chi lo produce e diffonde - un'azienda, un brand, o un semplice utente digitale - di raccontare una storia, elemento che porta vantaggio ai due attori coinvolti nella comunicazione, il mittente e il destinatario. Il mittente riesce, attraverso una storia, non solo a trasmettere un messaggio e dei valori, ma a farsi conoscere e riconoscere, e a creare con il destinatario un legame emotivo alimentato dalla narrazione creata da suoni e forme in movimento, obiettivo più difficilmente raggiungibile con altri mezzi di comunicazione più distanti e formali. Un video - una storia - diventa anche fonte di intrattenimento e sorgente di emozioni, e in quanto tale viene apprezzato dall'utente destinatario, il quale sarà invogliato a dedicarvi del tempo. Contenuti video permettono poi a chi li produce di poter dare vita -

attraverso il tono comunicativo scelto, le tracce audio, la musica, gli elementi grafici e visivi e il modo in cui questi si evolvono nel tempo - alla propria personalità, identificata da quella che diventa una grammatica estetica di definizione e riconoscibilità del brand; rendendo qualcosa riconoscibile è più semplice creare con esso un legame. Ogni brand, azienda o creator può avvalersi di un linguaggio di comunicazione personale attraverso il video, rendendosi così distinguibile. Questo linguaggio, poi, non è fisso ma può essere soggetto a sperimentazione, in modo tale da scegliere il tono e la narrativa che risultano più efficaci e che meglio rappresentano i valori del creator.

Il video risulta essere uno strumento efficace per la buona riuscita di campagne di comunicazione online, per chiunque abbia una serie di informazioni e messaggi da trasmettere e per chiunque abbia un'audience online da raggiungere.

Per queste ragioni, sempre di più le aziende includono il video nelle proprie strategie e campagne di marketing. In uno studio sullo stato del video marketing condotto nel 2023 da Wyzowl[1], un'azienda di produzione di video animati, risulta che, sul campione analizzato, il 91% delle aziende ha incluso video nella propria strategia di marketing per il 2022, e grazie al loro contributo è stato riscontrato un aumento nella conoscenza del prodotto e del servizio offerto, della *brand awareness*, del traffico generato, e delle vendite. Risultati positivi sono anche stati riscontrati dagli utenti e consumatori, i quali hanno confermato che la fruizione di video ha aumentato la comprensione dei prodotto e ed invogliato al loro acquisto. Inoltre, è risultato che il 91% dei consumatori apprezza il video content prodotto dai brand e desidera vederne di più nel 2023, e che oltre la metà delle persone è più propensa a condividere un video online con i propri amici che un qualsiasi altro contenuto, rendendolo dunque il content più condivisibile. Un risultato emerso da questo studio, interessante anche per comprendere la permeablità del video al di fuori dell'ambito del marketing, riguarda la quantità di ore dedicate al consumo dei video: in media una persona guarda circa 17 ore di video content online per settimana.

[1]Wyzowl. *The State of Video Marketing 2023*. [Online; ultimo accesso 13/02/2023]. 2023. URL: `https://wyzowl.s3.eu-west-2.amazonaws.com/pdfs/Wyzowl-Video-Survey-2023.pdf`

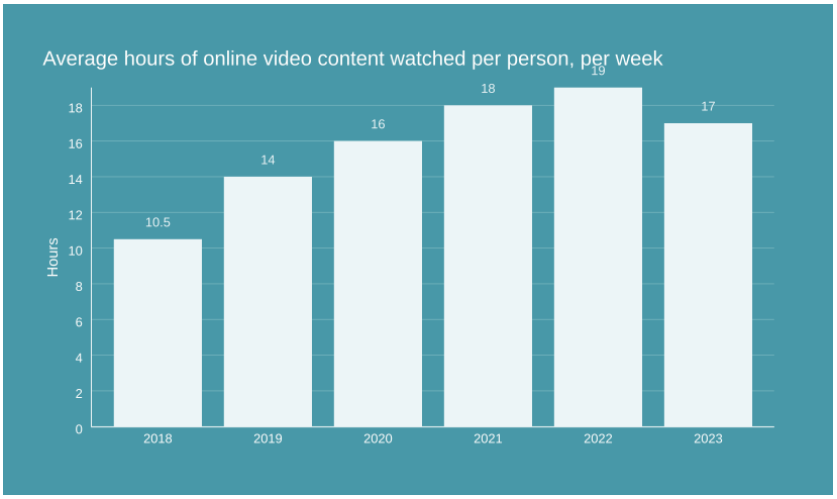


Figura 1.1: Consumo settimanale di video content negli anni.[2]

È chiaro dunque che gli utenti del web ricerchino ed apprezzino contenuti video, ed il mondo digitale che li ospita ne è consapevole. Siti web che contengono video - se opportunamente ottimizzati per la ricerca - vengono privilegiati da Google e altri motori di ricerca nel loro posizionamento. Anche solo empiricamente, si può notare come alcune tipologie di query - in primis query di tipo *how-to* - presentino tra i primi risultati offerti dai motori di ricerca proprio dei video. Secondo le statistiche fornite da Advanced Web Rankings[3], in Italia circa il 30% delle SERP (Search Engine Result Page) da Novembre 2022 a Febbraio 2023 contiene un video su desktop, numero che si innalza al 40% su mobile. Secondo i dati raccolti da Statista [4] il video online nel 2021 ha raggiunto un tasso di *audience reach* del 92% attraverso l'utenza internet globale.

[2]Riferirsi alla nota a piè di pagina 1.

[3]Advanced Web Rankings: `https://www.advancedwebranking.com/features-spread/`

[4]Statista. *Most popular video content type worldwide in 3rd quarter 2022, by weekly usage reach.* [Online; ultimo accesso 13/02/2023]. 2023. URL: `https://www.statista.com/statistics/1254810/top-video-content-type-by-global-reach/`

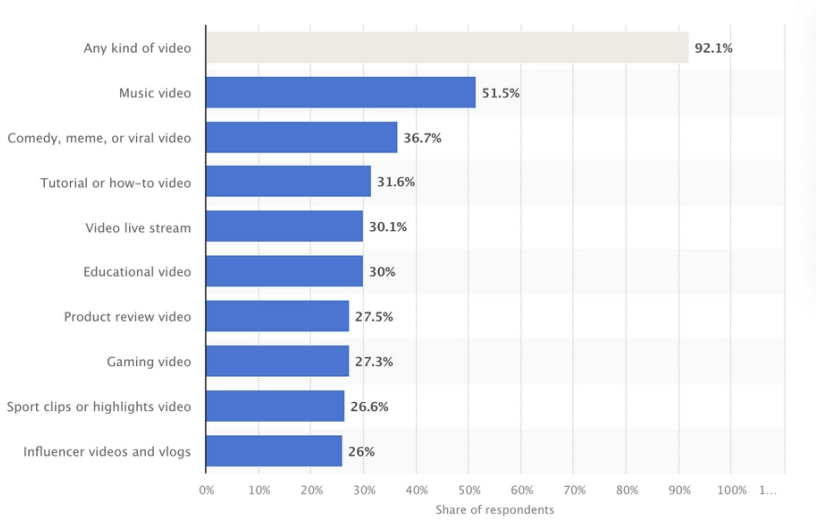


Figura 1.2: Tipologie popolari di video nel terzo trimestre del 2022 in base al reach settimanale.[5]

L'avvento del mobile ha contruibuito all'ascesa del video, rendendo la fruizione di video più semplice e a portata di mano: si sfuma sempre di più la separazione tra la dimensione quotidiana in *real life*, e la dimensione online, una si insidia nella vita dell'altro, coesistendo sullo stesso piano temporale, in un momento di fruizione di contenuti sempre presente. Basti pensare all'evoluzione della forma del contenuto nei social media. Se al loro avvento - complice anche lo stato dell'evoluzione tecnologica dell'epoca - il contenuto si presentava come post testuale, ora, il video è protagonista. I social media video-based diventano sempre più popolari: basti pensare alla crescente popolarità di TikTok, con una user base attiva superiore al miliardo di utenti nel mondo e con il primato di social app più utilizzata, o all'egemonia di YouTube, che risulta essere la seconda piattaforma social più utilizzata - dopo Facebook - con oltre 2 miliardi di utenti registrati globalmente.

[5]Riferisi alla nota a piè di pagina 4.

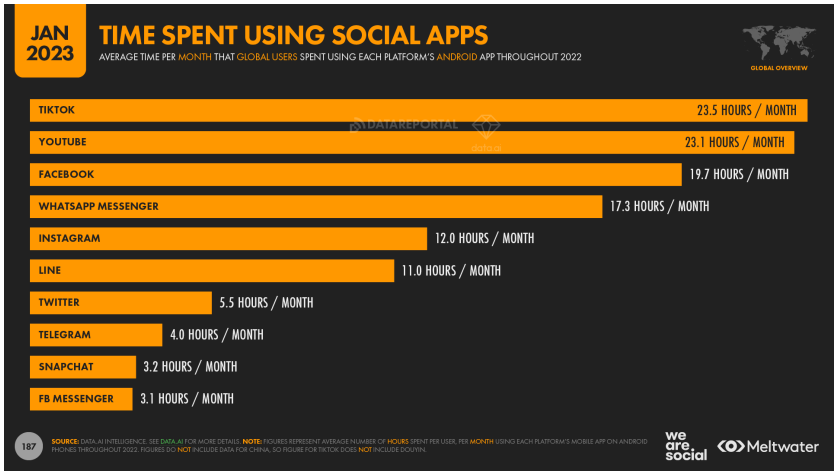


Figura 1.3: Tempo medio speso dagli utenti globali Android sulle diverse social app.[6]

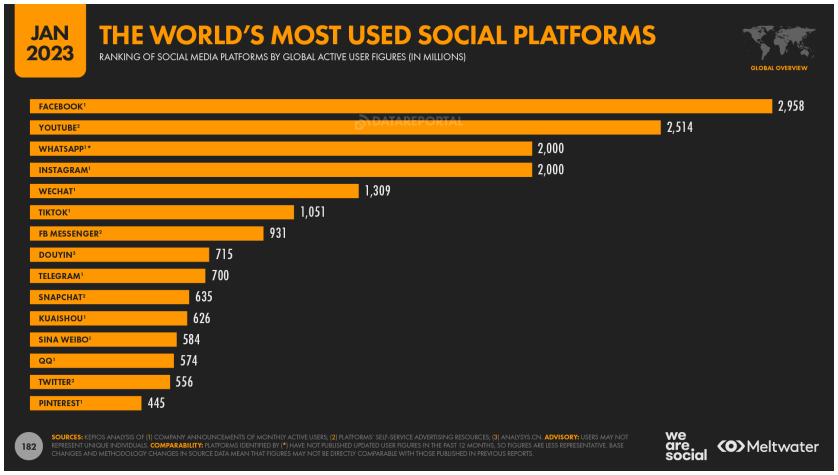


Figura 1.4: Classifica delle piattaforme social in base agli utenti globali attivi.[7]

Il video sembra dunque il giusto elemento per mettere d'accordo fruitore e piattaforma di fruizione. Da un punto di vista di realizzazione pratica, poi, richiede costi di produzione che possono essere esigui, garantendo un risul-

[6]Datareportal. *Global Social Media Statistics*. [Online; ultimo accesso 13/02/2023]. 2023. URL: `https://datareportal.com/social-media-users`

[7]Riferirsi alla nota a piè di pagina 6.

tato permanente nel tempo, e di facile condivisione nelle diverse piattaforme online.

Tramite queste analisi si evince dunque che il video content è uno strumento poliedrico che sfrutta il potere comunicativo audiovisivo per unire dimensioni differenti - informazione, intrattenimento, marketing, divulgazione - allo scopo di creare un punto di contatto emotivo tra mittente e destinatario di una comunicazione nell'ambiente online (e non solo), naturalmente saturo di stimoli ed informazioni. È proprio per questa ragione che diventa sempre più popolare e integrato nella vita digitale delle persone: *"in un mondo digitale impersonale, desideriamo connessione e personalità"* [8].

1.2 Motion graphics: animazione e comunicazione digitale

Video è un termine estremamente ampio, che racchiude all'interno di sé una varietà di tipologie comunicative e tecniche realizzative differenti. Nell'ambito della tesi si esplora nello specifico il ruolo del video animato, in particolar modo il video animato mediante tecniche di *motion graphics*.

Motion graphics si può tradurre letteralmente come *"grafiche in movimento"* [9], dove per grafiche si intende una moltitudine di diversi elementi visivi di design - forme, testi, immagini, etc. - il cui movimento è spesso accompagnato da musica ed effetti sonori. La motion graphics permette di *"spostare la conoscenza del design su nuovi mezzi, aggiungendo su di essi gli elementi di tempo e spazio - cioè creando movimento"* [10]: amplia il significato degli elementi grafici statici, definendone determinate caratteristiche di comportamento e il modo in cui si evolvono nel tempo, e donando così loro una vita, allo scopo di poter comunicare efficacemente con lo spettatore, rendendo l'assimilazione di informazioni rapida e piacevole.

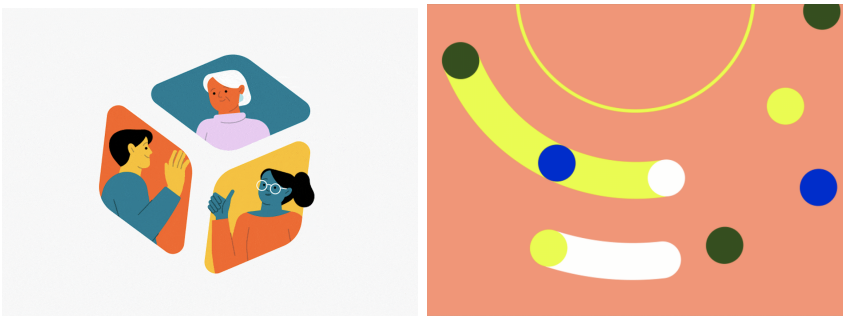
[8] Digital Marketing Institute. The *Key Role of Video Marketing*. [Online; ultimo accesso 13/02/2023]. 2018. URL: `https://digitalmarketinginstitute.com/blog/the-importance-of-video-marketing`

[9] Adobe. *Motion graphics explained: definition, history and examples*. [Online; ultimo accesso 17/02/2023]. URL: `https://www.adobe.com/uk/creativecloud/animation/discover/motion-graphics.html`

[10] Silveira F. *What is Motion Graphics?*. Mowe Studio. [Online; ultimo accesso 17/02/2023]. URL: `https://mowe.studio/what-is-motion-graphics/`

1.2.1 I principi della motion graphics e il suo rapporto con l'animazione

La motion graphics è una sottocategoria dell'*umbrella term* "animazione". Tuttavia, questa presenta caratteristiche e definizioni che la distinguono dall'animazione tradizionale più nota. Solitamente si compone di sequenze più brevi e - benché mantenga un metodo e uno scopo anche narrativo - è principalmente votata o all'intrattenimento o alla trasmissione di informazioni via *visual communication*. Gli elementi messi in movimento nella motion graphics, a differenza dell'animazione tradizionale, spesso non sono solo elementi organici e realistici, ma elementi di design: testi, colori, pattern e forme astratte. Anche quando gli elementi rappresentati provengono dal mondo reale - come nel caso di ambientazioni o personaggi - questi vengono opportunamente stilizzati in base alle specifiche richieste di design. Benché si ricerchi coerenza e riconoscibilità dell'elemento e del suo movimento con il corrispondente nel mondo reale, il fotorealismo non è lo scopo della motion graphics: gli elementi vengono animati e definiti anche in base ai messaggi da comunicare e obiettivi di design prestabiliti.



(a) Personaggi[11] (b) Forme astratte[12]

Figura 1.5: Frame di animazioni in motion graphics.

La motion graphics prende dunque dal mondo dell'animazione e dal mondo del design, contaminando uno con l'altro. Per questa ragione si basa sui principi cardine di entrambe le discipline.

[11]Illo. *Character Design for newsletter*. File GIF. [Online; ultimo accesso 17/02/2023]. URL: `https://dribbble.com/shots/18021763-Character-design-for-newsletter`

[12]Illo. *Dynamic loop with flashy colors*. File GIF. [Online; ultimo accesso 17/02/2023]. URL: `https://dribbble.com/shots/15177966-Dynamic-loop-with-flashy-colors`

I principi dell'animazione: i 12 Principi di Thomas e Johnston

L'animazione mette in moto elementi altrimenti statici, con lo scopo di rendere il movimento creato coerente con il movimento reale corrispondente, movimento a cui l'occhio dello spettatore è abituato. Benché il movimento debba essere verosimile - ovvero coerente con le regole della fisica - questo non deve necessariamente ricalcare la realtà, ma può essere stilizzato in modo tale da trasmettere determinate emozioni e messaggi. Nel 1981 due animatori della Disney, Frank Thomas e Ollie Johnston, introdussero nel loro libro "*The Illusion of Life: Disney Animation*"[13], i cosiddetti 12 principi dell'animazione, 12 regole estratte empiricamente dalla loro esperienza e dal loro lavoro da adottare per creare l'illusione di un movimento verosimile, per l'appunto *l'illusione della vita.*

Si descrivono brevemente questi principi:

- *Squash and Stretch* (Schiacciare ed Allungare). Le forme di un'animazione non devono apparire rigide, ma dotate di peso e flessibilità, dunque nel movimento che compiono si devono allungare e schiacciare sulla base delle interazioni con l'ambiente, mantenendo tuttavia il volume occupato nella scena costante. Per esempio, una palla che rimbalza risulta schiacciata durante l'impatto con il terreno, ed allungata durante la traiettoria in aria.

- *Anticipation* (Anticipazione). Si prepara lo spettatore ad un'azione in modo tale da renderla più realistica. Per esempio, se vi è un personaggio che salta, prima di muoversi verso l'alto, si abbassa per avere la spinta necessaria, oppure se vi è un elemento inizialmente off-screen da inquadrare, si può mostrare prima della sua comparsa un personaggio della scena guardare verso la sua direzione.

- *Staging* (Ambientazione). Si dirige l'attenzione dello spettatore attraverso la definizione dell'ambientazione - luci, inquadrature, etc. - in modo tale da far comprendere quale sia l'elemento di interesse all'interno di una scena.

- *Straght Ahead Action and Pose to Pose* (Azione diretta o Posa per Posa). Questo principio si riferisce al metodo di animazione: *frame by frame* dall'inizio dell'animazione fino alla fine, o *pose to pose*, disegnando prima le pose principali e poi i frame intermedi. La formulazione

[13]Thomas F. and Johnston O. *"The Illusion of Life: Disney Animation"*. United States. Abeville Press. 1981.

di questo principio risale ancora all'epoca pre-CG: in ambito digitale si lavora pose to pose, l'animatore definisce le pose principali, e il programma di computer grafica le pose intermedie di transizione.

- *Follow Through and Overlapping Action* (Azioni Ritardate e Azioni Sovrapposte). Follow Through indica che alcune parti di un oggetto continuano a muoversi anche dopo la fine del movimento del corpo principale. In particolar modo continuano a muoversi coerentemente con la direzione di moto, per poi essere riportate indietro alla posa di quiete. Se, per esempio, un oggetto con un'antenna si sta muovendo e si ferma improvvisamente, l'antenna continua a muoversi, prima in avanti e poi in indietro fino a tornare dritta. Overlapping Action indica che le diverse parti di un oggetto composto tendono a muoversi con diverse velocità. Per esempio, in un ciclo di camminata, il movimento delle braccia avviene con un ritmo diverso a quello della testa.

- *Slow In and Slow Out* (Accelerazione e Decelerazione). I movimenti nel mondo reale presentano fasi di accelerazione e decelerazione. La variazione di velocità nel movimento deve essere trasportata anche nell'animazione.

- *Arcs* (Archi). La maggior parte dei movimenti nel mondo reale - fatta eccezione per quelli meccanici - avviene su traiettorie non perfettamente rettilinee, ma arcuate. Nell'animazione, dunque, è preferibile evitare traiettorie perfettamente dritte.

- *Secondary Action* (Azione Secondaria). L'inserimento di un'azione secondaria per dare supporto al movimento principale può arricchire la scena e comunicare determinati messaggi. Per esempio in un ciclo di camminata, il modo in cui il personaggio muove la testa o muove le braccia non solo rende la camminata più realistica, ma può comunicare personalità, stati d'animo e situazioni differenti.

- *Timing* (Tempo). Il tempo in cui avviene un movimento dà informazioni riguardo alla natura dell'oggetto e dell'azione su di esso compiuto. Per esempio un oggetto pesante si muove più lentamente di un oggetto leggero.

- *Exaggeration* (Esagerazione). Questo è il principio che distingue l'animazione dal movimento reale. Spesso la resa grafica di un movimento completamente realistico può essere insoddisfacente per il messaggio

che si vuole comunicare. In base al tipo di animazione che si vuole ottenere - più caricaturale o più realistica - si possono esagerare i movimenti e i principi precedentemente esposti.

- *Solid Drawing* (Abilità nel disegno). Questo principio si riferisce al beneficio che trae un animatore che abbia una conoscenza estesa nel disegno e delle sue regole: forme, luci, ombre, prospettiva, anatomia, etc. Anche se in un contesto CG-based non è più necessario saper disegnare, una buona conoscenza tecnica può sicuramente aiutare l'animatore.

- *Appeal* (Fascino). Gli elementi di un'animazione devono essere in qualche modo interessanti ed accattivanti. Questo principio racchiude i precedenti.

Tali principi sono stati trasportati sul piano della motion graphics e per esso adattati. Jorge R.Canedo dello studio di motion design e animazione Ordinary Folk[14] ha ricavato dai 12 principi dell'animazione, 10 principi analoghi per il motion design[15]:

- *Timing, Spacing and Rhythm* (Tempo, Spazio e Ritmo)
- *Eases* (Accelerazione e Decelerazione)
- *Mass and Weight* (Massa e Peso)
- *Anticipation* (Anticipazione)
- *Arcs* (Archi)
- *Squash, Stretch and Smears* (Schiacciare, Allungare e Lasciare il segno)
- *Follow Through and Overlapping Action* (Azione Ritardata e Azione Sovrapposta)
- *Exaggeration* (Esagerazione)
- *Secondary or Layered Animation* (Animazione Secondaria o In Aggiunta)
- *Appeal* (Fascino)

[14]Ordinary Folk: `https://www.ordinaryfolk.co`

[15]@jrcanest (18 ott 2016) Lesson 2 of my @learnsquared Motion Design class is live! Talking about the 10 principles of Motion Design based on Disney's 12 principles.[Twitter Post; ultimo accesso 17/02/2023]. URL: `https://twitter.com/jrcanest/status/788441769864073216`

I principi del design nella motion graphics

Secondo quanto indicato nell'articolo *"Applying Design Principles To Motion Design"* [16]di Lusting A., tra i principi del design applicati alla motion graphics si possono considerare i seguenti.

- Ripetizione: l'utilizzo ripetuto di elementi o strutture di design. Questo principio viene impiegato per creare un tema comune all'interno della composizione, affnchè conferisca ad essa coerenza e la renda riconoscibile. Se nel graphic design la ripetizione avviene solo nello spazio, nella motion graphics avviene anche nella dimensione aggiuntiva del tempo (sono estremamente comuni, per esempio, animazioni cicliche).

- Ritmo: è l'elemento che indica *"come e quando vediamo cosa e dove"* [17]. Attraverso la ripetizione e l'alternanza di elementi grafici è possibile conferire alla scena un senso di ritmo e movimento (che può essere regolare, fluido o progressivo). Nella motion graphics il ritmo diventa fondamentale, non solo per quanto riguarda l'aspetto grafico degli elementi, ma anche il loro comportamento nel tempo. Spesso è presente anche una componente audio nei video in motion graphics, ed il corretto impiego del ritmo permette di legare la dimensione uditiva a quella visiva. La definizione del ritmo è anche cruciale per conferire significato alla scena. Ritmi lenti sono adatti a conferire all'animazione un senso di calma, ritmi veloci un senso di frenesia ed energia. Possono essere impiegati entrambi per alternare momenti di riposo e momenti di climax, necessari per una corretta struttura narrativa all'interno del video.

- Whitespace: è un termine che indica la porzione di spazio che non contiene elementi di interesse dal punto di vista del design. Il whitespace viene utilizzato per delineare ed esaltare gli elementi di focus della scena. Questo concetto permane nella motion graphics, e si amplia sulla dimensione del tempo come quella porzione del tempo nella quale non è definita alcuna animazione, e che crea contrasto invece con il precedente o successivo movimento degli elementi.

- Varietà: variare caratteristiche degli elementi in una scena permette sia di creare maggior interesse visivo, sia di focalizzare l'attenzione

[16]Lusting A. *Applying Design Principles To Motion Design* Fable. 2022. [Online; ultimo accesso 17/02/2023]. URL: `https://www.fable.app/blog/applying-design-principles-to-motion-design/`

[17]Riferisi alla nota a piè di pagina 16.

dello spettatore sugli elementi di contrasto. Nell'ambito della motion graphics, la varietà si applica sia nello spazio - sull'aspetto e sulla disposizione degli elementi - sia nel tempo - sul modo in cui questi si evolvono.

- Movimento: indica il modo in cui l'occhio dello spettatore viene guidato all'interno della scena. Questo principio è quello che più viene valorizzato nel passaggio da graphic a motion design: se nel graphic design il senso di movimento può essere creato con il dinamismo delle forme di elementi statici, nella motion graphics viene effettivamente realizzato tramite animazione di elementi e delle loro proprietà.

- Gerarchia: indica l'ordine in cui gli elementi vengono mostrati e la loro rilevanza nella scena. Strutturare una gerarchia permette di dare un ordine di importanza agli elementi, enfatizzando quelli su cui porre il focus, attraverso la posizione degli elementi, il loro colore, dimensione e forma, e, nel caso della motion graphics, movimento.

- Unità: indica l'armonia di tutti gli elementi della scena e racchiude gli elementi precedentemente analizzati.

Ulteriori principi del graphic design da applicare alla motion graphics sono quelli esposti dalla "Teoria della Gestalt". La Gestalt è una corrente psicologica teorizzata dallo psicologo Max Wertheimer e sviluppata dai suoi allievi Kurt Koffka e Wolfgang Köhler che concentra i suoi studi sulla percezione umana. Il principio alla base della teoria della Gestalt è che "il tutto è differente dalla forma delle sue parti", e che dunque la percezione di una scena va oltre la percezione di ogni elemento che la compone. Il cervello umano tende ad assegnare un significato a ciò che si sta osservando, organizzando le informazioni e trovando tra di esse nessi e legami logici e cognitivi. Sfruttando i principi della Gestalt, dunque, è possibile comunicare determinate informazioni anche in maniera "implicita", attraverso l'utilizzo di forme, colori, e la disposizione dell'elemento nello spazio.

Non esiste una lista definitiva di principi della Gestalt per il design universalmente riconosciuta, tuttavia i seguenti sono tra più utilizzati da graphic e motion designer:

- Principio della Vicinanza: elementi vicini tra di loro vengono percepiti come un'unità o appartenenti ad un unico gruppo.

- Principio della Somiglianza: elementi simili vengono percepiti come un'unità o appartenenti ad uno stesso gruppo. La somiglianza può

essere sulla base del colore, della dimensione, della forma, o dell'orientamento. L'intensità di somiglianza del colore è più efficace di quella della dimensione, a sua volta più efficace di quella della forma. Un elemento può essere reso come punto focale della scena rendendolo differente - un'anomalia - rispetto agli elementi circostanti, guidando così l'attenzione dello spettatore su un punto specifico (principio del punto focale).

- Principio di Figura/Sfondo: in una composizione si tende sempre a percepire alcuni elementi come figura, e gli altri come sfondo.
- Principio di Chiusura: la mente umana tende a percepire le forme come chiuse anche quando queste sono aperte, riempiendo le porzioni di spazio mancanti di un design o di un'immagine.
- Principio di Continuità: la mente umana tende a percorrere il percorso più semplice mostrato, dunque se gli elementi sono disposti secondo una particolare forma, lo sguardo tende a seguire quella forma e quell'ordine, senza essere distratti da altri stimoli.
- Principio di Simmetria: elementi simmetrici vengono percepiti come un'unità o appartenenti ad uno stesso gruppo.
- Principio del Destino Comune: elementi che presentano un movimento uguale tra loro e diverso da altri vengono percepiti come un'entità o appartenenti ad un unico gruppo.
- Principio di Esperienza Passata: l'esperienza modella la percezione, dunque se gli elementi di una composizione ricordano allo spettatore l'esperienza percettiva di un determinato oggetto, questi elementi vengono considerati come un'unica figura che rappresenta quell'oggetto.

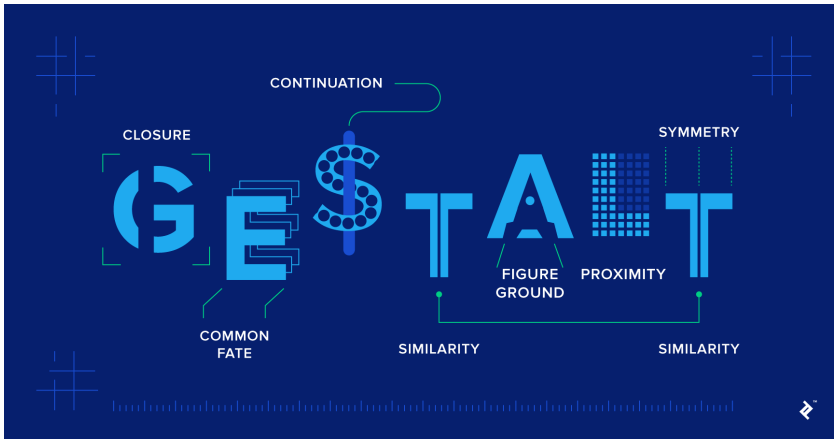


Figura 1.6: Infografica riassuntiva sui principali principi della Gestalt.[18]

I principi della Gestalt possono essere dunque utili per creare scene di motion graphics interessanti ed efficaci nella comunicazione visiva: l'aspetto e la disposizione degli elementi nella scena trasmettono implicitamente delle informazioni e possono rendere il messaggio più chiaro ed impattante.

1.2.2 Gli ambiti di applicazione della motion graphics

La motion graphics nasce grazie al cinema. Il primo utilizzo di motion graphics si deve infatti ai titoli di apertura dei film, creati affinché gli spettatori potessero sapere di più sugli attori della pellicola. Il suo primo grande sviluppo ha poi inizio nel dopoguerra, grazie alle sperimentazioni dei graphic designer Saul ed Elaine Bass, Maurice Binder e Pablo Ferro. Il loro approccio creativo alla motion graphics ha permesso che i titoli di apertura non fossero presenti solo per fornire informazioni tecniche sul film, ma anche per instaurare in sala l'atmosfera richiesta, coinvolgere spettatori distratti ed introdurli al film stesso.

[18]Philips Miklos. *How to Use Powerful Gestalt Principles in Design (with Infographic)*.[Online; ultimo accesso 18/02/2023]. URL: `https://www.toptal.com/designers/visual/infographic-gestalt-principles-of-design`

Figura 1.7: Frame dei titoli di apertura del film *The Seven Year Itch* (Billy Wilder, 1955) prodotti da Saul Bass.[19]

Questo modo nuovo di giocare e disporre di forme e parole ha rappresentato il punto di svolta per la crescente popolarità della motion graphics, la quale si è espansa per prima nei settori cinematografici e televisivi. Di pari passo con lo sviluppo tecnologico - che ha fornito nuovi schermi di visualizzazione e programmi di computer grafica progressivamente più accessibili - la motion graphics ha iniziato a diffondersi: da promo televisive e video musicali, fino alle *web-experience* e alla *data visualization.*

Oggigiorno la motion graphics vive in ogni schermo: ovunque vi sia uno schermo - cinema, televisori, smartphone, interfacce grafiche o più generalmente oggetti smart - è probabile che vi siano elementi di motion graphics.

I video realizzati in motion graphics sono estremamente versatili e caratterizzati da benefici che li rendono ideali per diversi scopi: sono accattivanti e memorabili, esteticamente piacevoli, efficaci nella comunicazione visiva di informazioni e più rapidi ed economici della controparte *live-action.* Per questa ragioni la motion graphics viene utilizzata oggigiorno per una moltitudine di scopi: creare *ads*, sequenze di titoli per film e serie televisive, video musicali, video esplicativi per la condivisione di dati o informazioni, e molto altro.

Si descrivono in seguito brevemente alcuni dei principali ambiti di applicazione.

Film e Serie TV

La motion graphics nasce nell'ambito cinematografico, e vi rimane ancora di fondamentale importanza, così come nell'ambito televisivo. Sempre più curati, il momento dei titoli di testa o di coda diventa in primis un'esperienza di intrattenimento, rappresenta un ottimo strumento per introdurre il tono della serie o del film attraverso l'estetica e la musica utilizzata, e diventa

[19]Movie Titles *Saul Bass: The Seven Year Itch (1955) title sequence.* 6 set 2017. [Online; ultimo accesso 17/02/2023]. URL: `https://www.youtube.com/watch?v=Gg1MA03VZB4`

anche un elemento di riconoscibilità del prodotto (soprattutto per prodotti seriali, quali saghe cinematografiche o serie televisive).

Si portano come esempio i titoli di apertura della serie televisiva prodotta da Hulu, Candy - che narra le vicende di una casalinga degli anni '80 accusata di aver commesso un omicidio. I titoli sono stati realizzati dall'agenzia creativa Imaginary Forces[20] attraverso un'estetica retrò, trasportando sulla dimensione testuale e visiva lo stereotipo della "donna di casa", e la sua conseguente disgregazione: la sequenza è composta da una serie di scene che raffigurano squarci di manuali di istruzioni e testi riguardanti attività stereotipicamente femminili, scene in cui, in una catena di causa-effetto, gli elementi - in primis i testi - si disassemblano e si sgretolano.

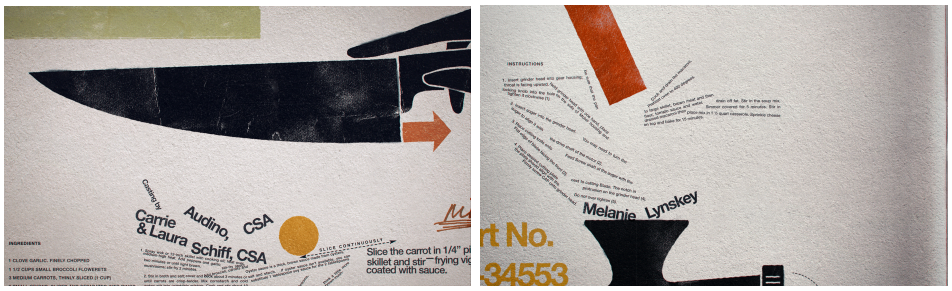


Figura 1.8: Frame dei titoli di apertura della serie *Candy* (Veith et al. 2022).[21]

Musica

Un ambito in cui la motion graphics è massicciamente presente è quello musicale. I principi di ritmo e movimento, e l'integrazione di musica ed elementi sonori a elementi tipografici, forme ed immagini rendono la motion graphics particolarmente adatta a questo campo, per esempio per creare *lyrics video* o arricchire video musicali.

Si porta come esempio il video musicale della canzone Do I Wanna Know? del gruppo britannico Arctic Monkeys. La celebre canzone, caratterizzata da un beat riconoscibile, è accompagnata da un video interamente realizzato in motion graphics. Inizialmente è presente una forma d'onda che reagisce al riff di chitarra e alla voce principale, forma d'onda che poi si evolve e muta - fino a diventare una vera e propria *character animation* - in base

[20]Imaginary Forces: `https://imaginaryforces.com`

[21]Imaginary Forces. *Candy Main Title*. 9 mag 2022. [Online; ultimo accesso 17/02/2023]. URL: `https://www.youtube.com/watch?v=MvU6cJqgCCk`

all'evoluzione della canzone e ai suoi momenti di interesse (*lyrics* particolari, cori, ritornello, etc.).

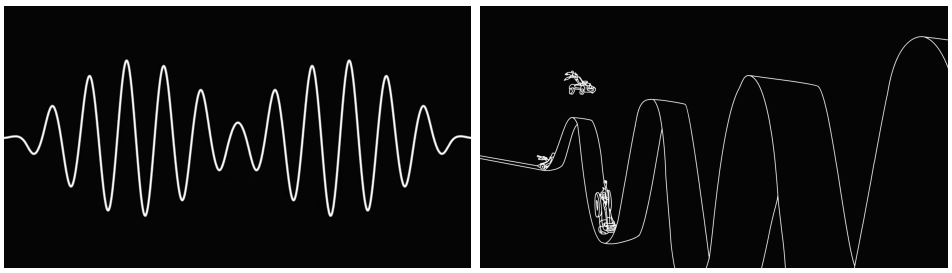


Figura 1.9: Frame del video musicale di *Do I Wanna Know?* (David Wilson, 2013) della band Arctic Monkeys.[22]

Questo esempio presenta poi un ulteriore elemento di interesse che va oltre al video musicale in sé: la cover dell'album da cui è tratta la canzone rappresenta proprio la forma d'onda protagonista del video. Il video musicale permette di unire dunque i due elementi - canzone ed album - e di creare un'esperienza di fruizione coerente, dando vita al simbolo che all'epoca dell'album rappresentava l'intera band.

Branding e Marketing

Un ambito in cui la motion graphics è sempre più presente, soprattutto con l'avvento dei social media, è quello del marketing e del branding aziendale, per la produzione di pubblicità *story-driven*, *eye-catching* post animati, video esplicativi, video centrati sui prodotti offerti, loghi animati, etc.

Con l'uso della motion graphics è possibile creare un linguaggio visivo coerente attraverso i diversi contenuti (specialmente digitali) del brand, anche dando vita ad elementi finora statici come i loghi. I loghi animati, per esempio, presentano uno stimolo visivo ed emotivo più forte della loro controparte statica, e per questa ragione aiutano a rendere il brand più memorabile. Inoltre, dando ed esso vita, definendo la sua animazione, il ritmo ed il modo in cui evolve, un logo si carica di maggiore significato, ed è più efficace nella comunicazione dei valori offerti dall'azienda. L'utilizzo della motion graphics nel marketing va ben oltre l'animazione dei loghi: può essere utilizzata per esempio per creare video sui prodotti offerti (particolarmente efficaci quando il prodotto non è "esteticamente forte" come per esempio nel caso di un

[22]Arctic Monkeys. *Arctic Monkeys - Do I Wanna Know? (Official Video)*. 19 giu 2013. [Online; ultimo accesso 17/02/2023]. URL: `https://www.youtube.com/watch?v=bpOSxM0rNPM`

servizio digitale), video esplicativi (in quanto è possibile rendere più semplici concetti complessi, in modo anche piacevole), video che raccontino la storia del brand, o elementi visivi da inserire nelle piattaforme social e website dell'azienda in modo tale da essere riconoscibili. Utilizzando font, colori, suoni, movimenti, e scegliendo il tono comune da utilizzare - che sia questo, elegante, informale, sofisticato, etc. - è possibile creare video e elementi di motion graphics che rappresentino la personalità di un brand e raccontino la sua storia, creando con il (potenziale) cliente una connessione emotiva, e rimanendo impressi nella sua memoria. Inoltre, i video sono facilmente adattabili su diversi *device*, e sono di facile condivisibilità nelle piattaforme social (più di immagini e post testuali).

Si presentano due esempi di brand noti, Netflix e Rai.

L'intro delle serie e dei film prodotti da Netflix è istantaneamente riconoscibile, sia grazie all'animazione definita sul logo, sia grazie al tipico suono che l'accompagna. Il logo di Netflix si anima in un insieme di nastri di luce che si avvicinano all'occhio dello spettatore per poi accompagnarlo - con una schermata a nero introduttiva - al prodotto. Questa animazione crea uniformità e riconoscibilità nei prodotti del brand. Il design utilizzato, inoltre, è strettamente legato alla storia del brand stesso: i nastri di luce nella figura rappresentano le *thumbnail* delle serie televisive originali Netflix disposte in verticale. L'animazione è stata creata con lo scopo di mostrare "lo spettro delle storie, linguaggi, fan e creators che rendono Netflix magnifico". [23]

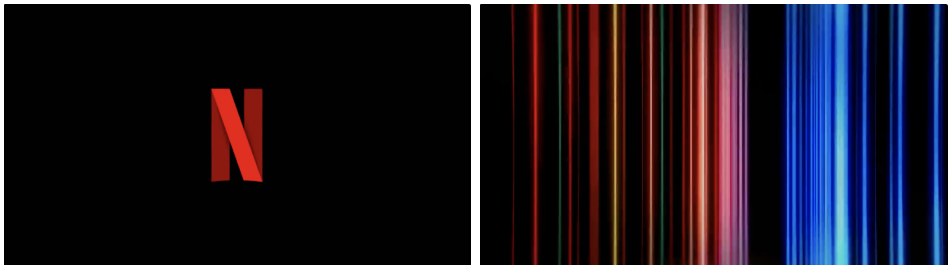


Figura 1.10: Frame del logo animato di Netflix.[24]

Il branding della Rai introdotto nel 2016 è un esempio di branding efficace per un'emittente televisiva. Per un broadcaster televisivo creare un linguaggio coerente è fondamentale: logo, pubblicità e video di intro e outro

[23] @netflixqueue (1 Feb 2019). *SOME PERSONAL NEWS: Starting today there's a new logo animation before our originals. It shows the spectrum of stories, languages, fans, & creators that make Netflix beautiful — now on a velvety background to better set the mood. And before you ask: no, the sound isn't changing.* [Ultimo accesso 17/02/2023]. URL: `https://twitter.com/netflixqueue/status/1091342921897406465?s=20`

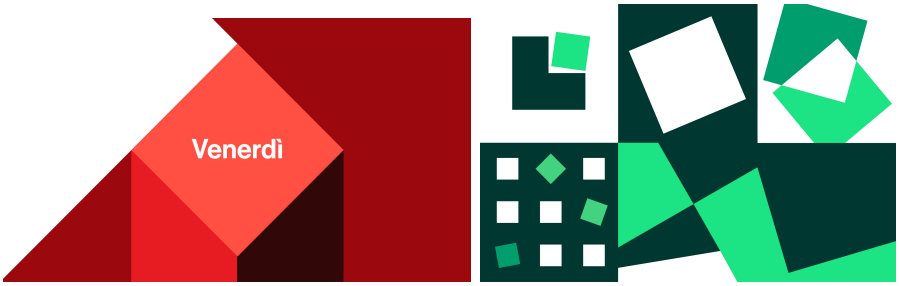
[24] Riferisi alla nota a piè di pagina 22.

- elementi animati - devono permettere allo spettatore di riconoscere istantaneamente il canale e l'emittente, e con esso i valori comunicativi associati. Il branding della Rai raggiunge questi obiettivi: ogni canale della rete risulta visivamente coerente, ed è possibile identificare subito l'appartenenza del canale all'emittente televisiva, pur mantenendo per ognuno di questi una rappresentazione grafica unica che permetta di comunicare i valori e la "personalità" del canale stesso. Inoltre "attraverso il nuovo rebranding, la Rai ha sviluppato un'identità grafica capace di esprimere la propria forza su qualsiasi piattaforma" [25] facilitando l'adattamento del linguaggio comunicativo sul web.

Figura 1.11: Elemento principale di design e valore associato di diversi canali Rai.[26]

[25]Bagatti R. (Chief of Brand and Creative RAI). URL: `https://www.digitalic.it/tech-news/rebranding-rai-roberto-bagatti`

[26]Digitalic *Rebranding Rai, Roberto Bagatti racconta l'evoluzione del design.*. [Online; ultimo accesso 17/02/2023]. URL: `https://www.digitalic.it/tech-news/rebranding-rai-roberto-bagatti`



(a) Elementi di design di Rai 2. (b) Elementi di design di Rai 3.

Figura 1.12: Elementi di design di diversi canali Rai.[27]

Informazione e Data Visualization

I video in motion graphics rendono la comunicazione visiva di informazioni un processo piacevole e coinvolgente. Per questa ragione vengono spesso utilizzati per scopi educativi ed informativi. La motion graphics può essere utilizzata sia per trasmettere informazioni con uno stile più narrativo e story-based, per esempio per scopi giornalistici, sociali o commerciali, sia per trasmettere informazioni con uno stile più scientifico e *data-based* per esempio per uno scopo più divulgativo o di business analysis. All'interno del vasto mondo della comunicazione di informazioni, uno dei campi nei quali la motion graphics risulta essere particolarmente efficace è quello della *data visualization*, ovvero della rappresentazione dei dati attraverso l'utilizzo di mezzi visivi come diagrammi, grafici, infografiche ed anche animazioni. Lo scopo della data visualization è dunque quello di rappresentare le informazioni tratte dalle operazioni di analisi sui dati, in modo tale da sottolinearne e mostrarne il significato, affinchè siano facili da comprendere. La data visualization è una disciplina estremamente ampia e per rappresentare analisi complesse di grandi data set in ambiti specifici esistono software dedicati. Tuttavia, la motion graphics è uno strumento incredibilmente utile per la comunicazione di dati per le general audience, poiché rende la loro visualizzazione coinvolgente e la loro comprensione rapida. L'animazione mette in moto infografiche finora statiche: permette di dare vita ai dati, e di narrare tramite essi una storia. Costruendo una relazione logica tra il movimento degli elementi che rappresentano i dati, è possibile guidare l'occhio dell'osservatore attraverso le informazioni, permettendo di focalizzare l'attenzione sui punti di interesse, rendendo più efficace e permanente nella memoria il

[27]Eloise Studio. *RAI Rebrand.* [Online; ultimo accesso 17/02/2023]. URL: `https://eloisa.studio/rai-rebrand`

messaggio che si vuole trasmettere. Con i colori, con la disposizione nello spazio e soprattutto con il movimento ed evoluzione degli elementi è possibile rendere la metabolizzazione delle informazioni più graduale e più chiara, mostrare il cambiamento dei dati in base a specifiche dimensioni di analisi, creare coerenza visiva con dati in relazione tra di loro, e porre l'accento su risultati di analisi, pattern e relazioni tra i dati che altrimenti potrebbero essere trascurati. Anche l'utilizzo dell'audio può essere utile per conferire significato ai dati visualizzati, costruire relazioni, e porre focus su determinati elementi di animazione. Si possono anche introdurre metafore visive legate al mondo dell'animazione e dell'intrattenimento nella sfera formale dei grafici per renderli più accattivanti e comprensibili, se lo scopo di divulgazione lo permette. L'utilizzo della motion graphics aiuta la costruzione di una vera e propria narrativa con i dati, tramite la loro personalizzazione e contestualizzazione, ed è dunque un elemento di supporto per il passaggio da data visualization a *data storytelling*.

Nell'esempio presentato, un *bar graph* e *line graph* animato e prodotto da Algo, l'animazione permette di poter osservare nella stessa scena prima i dati di ogni campo di analisi e successivamente la relazione tra di essi, e di poterli istantaneamente collegare - tramite associazione dei colori - alle etichette appropriate.

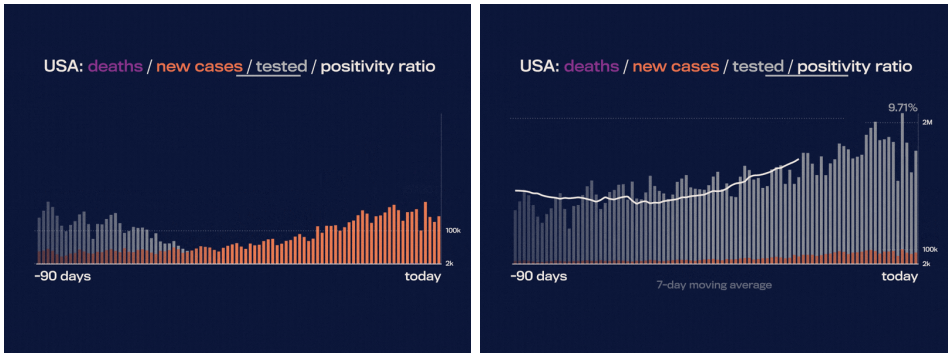


Figura 1.13: Frame dell'animazione di un *bar graph* e *line graph*.[28]

User Interface e User Experience (UI/UX)

La motion graphics può essere utilizzata per migliorare l'esperienza di utilizzo di un'interfaccia digitale, non solo da un punto di vista puramente estetico,

[28] Algo. *Bar graph daily update*. File GIF. [Online; ultimo accesso 17/02/2023]. URL: `https://dribbble.com/shots/14696840-Bar-graph-daily-update`

ma anche pratico: elementi di motion graphics sono utili per dialogare con l'interfaccia.

Possono essere utilizzati per rappresentare il feedback dell'interfaccia stessa, la quale comunica attraverso una metafora visiva il suo stato e la sua reazione all'input dell'utente. Può essere utilizzata per categorizzare le informazioni all'interno dell'interfaccia e rendere la navigazione attraverso di esse più semplice. Possono rendere più interessanti le microinterazioni, comunicare tramite *loading animation* stati di transizione, e in questi intrattenere e ridurre la percezione del tempo di attesa. In generale, aiutano a rendere l'applicativo più naturale, *responsive* e piacevole, facilitando la connessione tra utente ed interfaccia, e rendendo l'esperienza di interazione più organica.

Per esempio, l'attivazione di Siri su macOS e iOS è accompagnata da una specifica e riconoscibile animazione che permette all'utente di capire che il servizio è in azione.

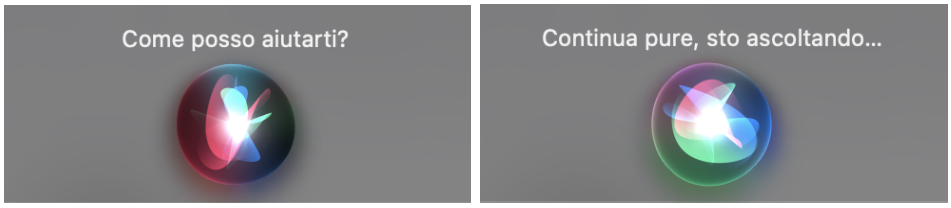


Figura 1.14: Frame dell'animazione di Siri su macOS.

1.2.3 Verso il video custom: automazione e personalizzazione

La presenza sempre più importante dei video, e il loro ruolo principale nelle campagne di comunicazione online, ha portato alla creazione di nuovi paradigmi e all'evoluzione dei loro processi di utilizzo e creazione. Sempre di più il video viene utilizzato come veicolo di informazioni, come strumento educativo, o come tool di marketing.

Si creano situazioni in cui è necessario creare una grande quantità di video da produrre, il cui contenuto è tuttavia variabile. Si pensa, per esempio, ai video informativi sullo stato di un fenomeno in osservazione, o ai video-recap dei dati di consumo dei clienti di un'azienda: questi vengono personalizzati sulla base di dati specifici. In questi casi non è possibile pensare di creare manualmente ogni singolo video, poiché il dispendio di risorse sarebbe eccessivamente elevato. Si parla dunque di *video automation*, ovvero la produzione automatizzata di video personalizzati in base a specifici dati di riferimento. L'idea che vi è alla base di un processo di video automation è dunque quella di disporre di un template generico - in cui le diverse feature audiovisive che

dovranno essere modificate presentano valori *placeholder* temporanei - e personalizzare con un processo automatico di scripting il singolo video in base ai dati presi in considerazione.

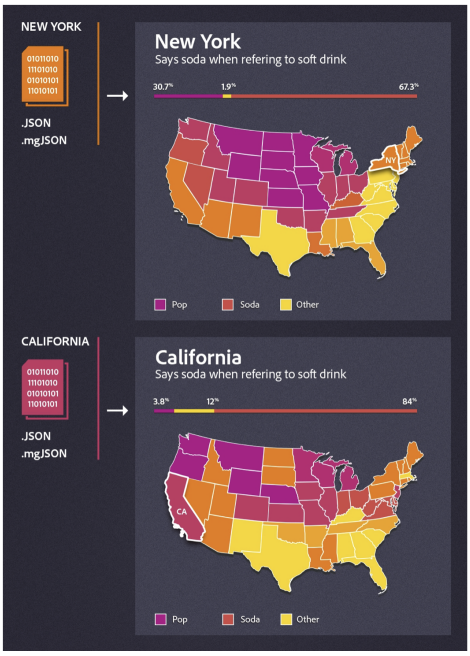


Figura 1.15: Mappa personalizzata in base ai dati in input.[29]

La video automation unisce dunque due dimensioni che apparentemente sembrano opposte: l'automazione di un processo che richiede sempre meno intervento umano, e la personalizzazione del risultato sulla base di dati riguardanti un particolare utente, fenomeno, prodotto o servizio. Automazione e personalizzazione sono uno al servizio dell'altra.

I pilastri sui quali si fonda la video automation - video e dati - spaziano in diversi ambiti. Per questa ragione, i video personalizzati possono essere utilizzati per diversi scopi.

Poiché cambiano sulla base di dati presi in considerazione, uno degli ambiti più organici di applicazione dei video personalizzati si ritrova nella data visualization. Come esposto precedentemente, rappresentare dati con un video in motion graphics, dove il movimento e l'evoluzione delle forme e la presenza dell'audio possono guidare l'occhio dell'osservatore attraverso le informazioni, è un ottimo modo per presentare dati ad un pubblico generico. Disporre di un processo di produzione che adatti automaticamente il video

[29] *Work with data-driven animation.* URL: `https://helpx.adobe.com/after-effects/using/data-driven-animations.html`

in uscita in base ai dati da visualizzare permetterebbe di risparmiare risorse e velocizzare la comunicazione delle informazioni. La data visualization poi, può essere declinata per diversi scopi: può essere semplicemente il servizio offerto dall'ente che la produce, può essere inclusa nella diffusione di notizie ed articoli, o essere utilizzata come supporto a campagne di comunicazione di diversa natura.

Un altro campo di utilizzo - vicino alla data visualization - è la diffusione di notizie. Il mondo delle news è sicuramente un ambiente in cui le informazioni si evolvono costantemente, e dove lo scopo principale è rendere queste informazioni visibili e consultabili al pubblico il più rapidamente possibile: è un ambito che giova sicuramente della reattività e della rapidità di produzione della video automation.

Un ulteriore ambito di applicazione per i video personalizzati, meno apparente ma altrettanto presente, può essere quello del marketing. Questi possono agire - dal punto di vista della customizzazione - sul prodotto o sull'utente-consumatore.

Nel primo caso i video vengono personalizzati in base ai dati che descrivono un prodotto o servizio offerto, in modo tale da creare un display su di esso personalizzato. Avere un template sempre uguale per offrire un supporto audiovisivo ad un prodotto permette di velocizzare i tempi di creazione di contenuti, garantendo allo stesso tempo coerenza visiva e semantica attraverso i diversi prodotti, ed apportando dunque un beneficio alla *brand identity* e *brand awareness*.

Nel secondo caso i video vengono personalizzati in base ai dati di consumo di un cliente. In questo caso risultano utili per creare un legame unico, speciale, - per l'appunto *personale* - tra il singolo utente e il creator o brand che lo produce: l'utente si sente visto ed ascoltato. Quelli che prima erano dati discretamente utilizzati dall'azienda a scopi di studio e di analisi, vengono ora condivisi con la stessa persona che li ha prodotti, e diventano uno strumento per riconoscere e distinguere un utente dall'altro, per far sentire e far capire all'utente di essere speciale. I video personalizzati permettono dunque di poter creare campagne di comunicazione "1:1": il brand non comunica più con una utenza collettiva, ma singolarmente con ogni utente. In questo modo aumenta il tasso di fidelizzazione e lealtà dell'utente-consumatore al brand.

Per quanto riguarda l'utilizzo dei video automatizzati nelle campagne di marketing, come esposto nell'articolo scritto da Melvin J. per Wyzowl, *Personalized Video: Complete Guide (Benefits, Tips, Examples)* [30], questi

[30]Melvin J. Personalized Video: Complete Guide (Benefits, Tips, Examples). Wyzowl. (10 feb 2023). [Online; ultimo accesso 17/02/2023]. URL: `https://www.wyzowl.com/personalized-video/`

presentano diversi benefici:

- Risultano essere particolarmente coinvolgenti per il consumatore, ed utili per aumentare i tassi di *customer retention* (la capacità dell'azienda di mantenere i propri clienti, fidelizzandoli) e di *customer conversion* (la percentuale di potenziali clienti che compie l'azione obiettivo della strategia di marketing).
- Aumentano la *brand awareness*.
- Aumentano il traffico sui siti e sulle *landing pages* del brand.
- Risultano essere un elemento e un servizio attraente ed interessante agli occhi dell'utente.

I video personalizzati si possono distinguere e classificare sulla base di diversi elementi di variabilità al suo interno.

Input e recupero dei dati

I dati su cui viene customizzato un video possono essere:

- Dati variabili recuperati automaticamente senza intervento umano. È il caso di dati provenienti da database e da pagine web o dati generati da dispositivi muniti di sensori. Poiché questi dati sono variabili nel tempo e la loro produzione è periodica, non è necessario inserire esplicitamente i dati di personalizzazione, questi vengono recuperati attraverso un servizio opportunamente temporizzato. È il caso di video che vengono prodotti - per esempio - per offrire un recap periodico nel tempo su un fenomeno monitorato ed in costante evoluzione. In questo caso non è necessario triggerare esplicitamente il render e la pubblicazione del video, ma questa può avvenire automaticamente.
- Dati statici generati da un utente. Si può trattare di dati statici globali come risultati di sondaggi, o di dati che rappresentano una serie di scelte di input di uno specifico utente. Vengono inseriti manualmente. In questo caso è necessario avviare esplicitamente il render e il delivery del video.
- Dati recuperati automaticamente ed eventualmente modificati dall'utente. Questa modalità unisce le due precedentemente analizzate, in modo tale da automatizzare e velocizzare il processo di recupero dati dando comunque la possibilità di apportare modifiche e correggere eventuali errori.

Durata

I video personalizzati possono avere durata:

- Fissa. A prescindere dai dati o dalle scelte dell'utente, il video in uscita ha sempre la stessa durata.
- Variabile. La durata del risultano di customizzazione può variare da video a video. La variabilità della durata può dipendere dal dato di personalizzazione fornito in input - per esempio se il video è un supporto ad una traccia audio inserita in input il video conformerà la propria durata in base a quella dell'audio - oppure può derivare da una formula di costruzione modulare del video. È possibile, infatti, suddividere un video in un insieme di singoli moduli customizzabili ed organizzabili secondo il criterio dell'utente. La presenza o assenza di diversi moduli porta a video con durata variabile.

1.2.4 Case Studies

Per comprendere le forme e gli utilizzi della video automation, si presentano alcuni *case studies*.

Spotify Wrapped 2021: Music Aura

Spotify Wrapped è una delle più celebri *online-shared* campagne di marketing al mondo, e rappresenta un esempio perfetto del potenziale del connubio tra dati e video. Promossa da Spotify, il celebre servizio di streaming audio, Spotify Wrapped è una campagna di marketing user-centrica che permette agli utenti della piattaforma di ripercorrere, ogni dicembre, il loro "anno in musica". I dati di ascolto di ogni utente raccolti durante l'anno vengono elaborati per creare una serie di video-grafiche che riassumono le loro abitudini musicali: canzoni ed artisti più ascoltati, minuti totali di ascolto, nuovi artisti scoperti, generi musicali preferiti, etc.

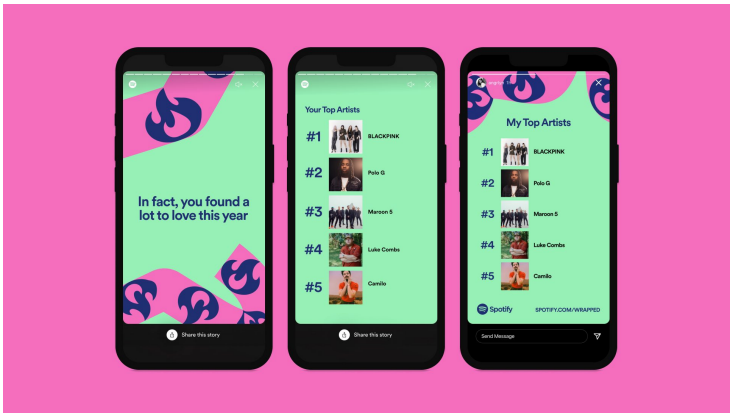


Figura 1.16: Esempio di una feature di Spotify Wrapped 2021.[31]

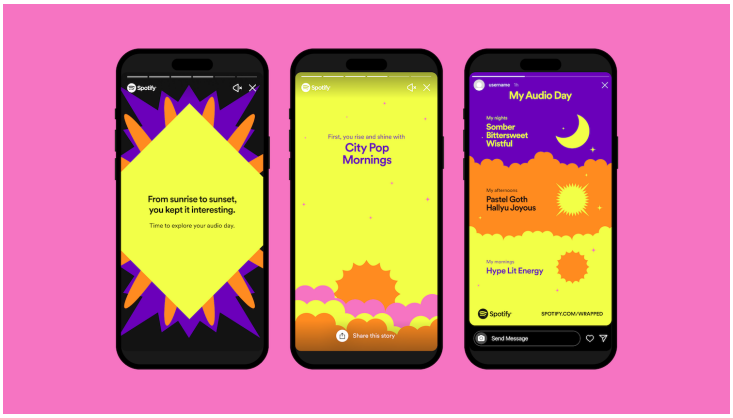


Figura 1.17: Esempio di una feature di Spotify Wrapped 2022.[32]

Spotify Wrapped utilizza una pratica digitale ormai inevitabile - la raccolta di dati - per cambiarne la narrativa portante: l'applicazione utilizza e mostra piuttosto apertamente l'informazione ottenuta, non "spia segretamente" il proprio utente, ma lo conosce e dà la possibilità all'utente di conoscersi. Il dato passa da puro strumento di analisi a contenuto portante di una campagna di marketing "fatta su misura" per ogni utente, è miccia per nuovi contenuti e dà vita ad un momento condiviso tra piattaforma e utente che porta beneficio a entrambi. Benché infatti la campagna sia stata portata

[32] *The Wait Is Over. Your Spotify 2021 Wrapped Is Here.* Spotify - For the Record. 1 dec 2021. [Online; ultimo accesso 20/02/2023]. URL: `https://newsroom.spotify.com/2021-12-01/the-wait-is-over-your-spotify-2021-wrapped-is-here/`

[32] *Everything You Need To Know About 2022 Wrapped* Spotify - For the Record. 30 nov 2022. [Online; ultimo accesso 20/02/2023]. URL: `https://newsroom.spotify.com/2022-11-30/everything-you-need-to-know-about-2022-wrapped/`

avanti in primis a beneficio dell'azienda, l'utenza di Spotify sembra apprezzarla ed attenderla di più di anno in anno, tanto che sulle diverse piattaforme social se ne inizia a parlare ben prima del suo arrivo.[33]

Si fa leva sulla condivisibilità della campagna e sulla grande quantità di *user generated content* (UGC) che questa crea per aumentare la brand awareness e la popolarità della piattaforma. Questa supporta e permette la condivisione diretta dei risultati ottenuti dal proprio Spotify Wrapped, permettendo di condividere separatamente i diversi moduli, così che gli utenti possano scegliere quali contenuti mostrare ad altri e quali tenere per sé. Tramite la condivisione online si viene a creare una sorta di ciclo di produzione di UGC continuo: un utente condivide online il proprio Spotify Wrapped, il quale viene visualizzato da un altro utente che per non "rimanere indietro" condivide a sua volta il proprio risultato. L'utente ha piacere nello scoprire e nel farsi conoscere - sia dalla piattaforma sia dall'utenza social: può paragonare le proprie abitudini a quelle di amici, familiari, e in generale di altri utenti, creando così momenti di conversazione e condividendo i propri pensieri, le predizioni, e i *meme* riguardanti i loro report annuali.

Spotify Wrapped rappresenta dunque uno strumento per aumentare la popolarità e il successo della piattaforma: secondo Protocol la campagna di Spotify Wrapped 2020 ha coinvolto più di 90 milioni di utenti e ha portato ad un aumento del 21% nei download dell'applicazione nella prima settimana di dicembre.[34] Le campagne dei due anni successivi sono state ancora più popolari, presentando dati di engagement in crescita tra 2020 e 2021 e tra 2021 e 2022.[35] La sua diffusione è così dilagante e la sua posizione nella cultura popolare digitale è così salda che si viene a creare un fenomeno di FOMO (*Fear of Missing Out*, paura di perdersi un'esperienza che altri stanno vivendo) per la porzione di popolazione online che non utilizza Spotify - e per questa ragione altri servizi di streaming audio hanno introdotto anch'essi la loro versione di recap annuale (come Youtube Music o Apple Music). Spotify Wrapped è diventato un elemento così intrinsecamente legato alla piattaforma e sempre più condiviso online che rappresenta anche una discriminante di differenziazione tra Spotify ed altre piattaforme di streaming audio.

[33] Tatton Z., *Spotify Wrapped: What Makes The Campaign So Successful?*. For The Curious. 13 dec 2022. [Online; ultimo accesso 20/02/2023]. URL: `https://www.forthecurious.co.uk/news/spotify-wrapped-what-makes-the-campaign-so-successful/`

[34] *The art and science of Spotify Wrapped.* Protocol. [Online; ultimo accesso 20/02/2023]. URL: `https://www.protocol.com/newsletters/sourcecode/spotify-wrapped`

[35] Woods K. *Spotify Wrapped: What marketers can learn from the viral campaign.* SproutSocial. 7 dic 2022. [Online; ultimo accesso 20/02/2022]. URL: `https://sproutsocial.com/insights/spotify-wrapped/`

Spotify personalizza e distingue le campagne di anno in anno, in modo tale da introdurre sempre elementi di novità che incoraggino l'engagement. Cambiano non solo il tema, le forme e i colori utilizzati - sempre coerenti con il brand Spotify - ma anche i dati visualizzati. Alcuni dei dati più standard vengono resi disponibili ogni anno. Si tratta di dati che richiedono un'elaborazione logica piuttosto semplice e diretta, ma non per questo risultano meno soddisfacenti per l'utente da scoprire e condividere online, come il caso del numero di minuti di ascolto del proprio artista preferito, o la classifica delle canzoni più ascoltate.

Altri invece rappresentano una novità di anno in anno. Questi richiedono spesso un processo di analisi ed un'elaborazione più complessa: per esempio, Spotify Wrapped 2022 permetteva agli utenti di conoscere la propria *personalità musicale* - categorizzata con una struttura che emula il test delle 16 personalità MBTI[36] - mentre Spotify Wrapped 2021 permetteva di scoprire la propria *aura musicale*. Queste informazioni non sono ovvie, ma vengono ricavate - sempre sulla base dei dati di ascolto - grazie ad un'analisi ben precisa, un linguaggio definito a priori ed uno storytelling creativo.

Si analizza nello specifico la feature Audio Aura di Spotify Wrapped 2021. L'Audio Aura è una visualizzazione grafica dell'aura definita dai due mood musicali più ascoltati da un utente.

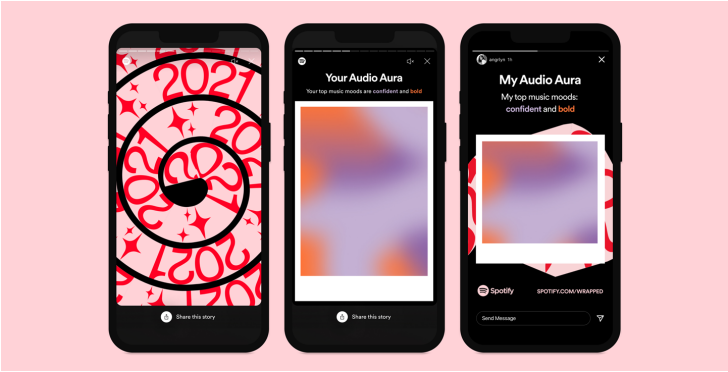


Figura 1.18: Esempio di feature Audio Aura.[37]

Per creare e definire questa feature, Spotify ha collaborato con un'aura reader, Mystic Michaela, la quale definisce un'aura come "la propria firma energetica personale" e afferma che un'aura reader sia capace di visualizzare

[36] 16 Personalità MBTI: `https://www.16personalities.com/it/tipi-di-personalita`

[37] *The Wait Is Over. Your Spotify 2021 Wrapped Is Here.* Spotify - For the Record. 1 dec 2021. [Online; ultimo accesso 20/02/2023]. URL: `https://newsroom.spotify.com/2021-12-01/the-wait-is-over-your-spotify-2021-wrapped-is-here/`

un'aura come una combinazione di colori, ognuno dei quali rappresentativo dei tratti caratteristici di una persona.[38] Per Spotify Wrapped 2021 si è deciso dunque di creare un'aura riguardante la personalità musicale di un ascoltatore. Si sono considerati i due principali mood musicali dell'utente, assegnando per ogni mood un colore, ed una percentuale di peso in base ai dati di ascolto. Il risultato ottenuto è una grafica che mostra un vorticoso gradiente di colori che rappresenta l'abitudine musicale dell'ascoltatore.

Nell'articolo "*The Audio Aura Story: Mystical to Mathematical*" di Taino Z., si spiega più nel dettaglio il processo di realizzazione.[39] Per definire una Audio Aura è stato necessario definire un metodo di identificazione dei due mood musicali maggiormente presenti nella storia d'ascolto di un utente, e successivamente - una volta identificata questa coppia - calcolare il peso associato ad ognuno di questi, ovvero quanto della musica ascoltata da ogni utente rientra nel mood considerato. Utilizzando un dataset di *mood descriptor* (descrittori dei mood) delle diverse tracce musicali, si ottiene la coppia di mood principali e i relativi pesi tramite le seguenti operazioni:

- Si conta il numero di stream per ogni traccia e si recupera il mood descriptor principale determinato dal dataset relativo.
- Si inserisce il mood dentro una delle macro-categorie definite.
- Si trova tramite aggregazione il numero totale di stream per ogni categoria di mood e si considera il mood descriptor con il maggiore numero di stream.
- Si compie un ultimo calcolo per trovare la percentuale di stream di un mood sul numero di stream totale.

Nell'esempio mostrato, le tracce con un maggiore numero di stream appartengono ai mood *happy* e *calm*.

[38] *Learn More About the Audio Aura in Your Spotify 2021 Wrapped With Aura Reader Mystic Michaela.* Spotify - For the Record. 1 dec 2021. [Online; ultimo accesso 20/02/2023]. URL: `https://newsroom.spotify.com/2021-12-01/learn-more-about-the-audio-aura\-in-your-spotify-2021-wrapped-with-aura-reader-mystic-michaela/`

[39] Taino Z. *The Audio Aura Story: Mystical to Mathematical.* Spotify R&D Engineering. 17 dec 2021. [Online; ultimo accesso 20/02/2023]. URL: `https://engineering.atspotify.com/2021/12/the-audio-aura-story-mystical-to-mathematical/`

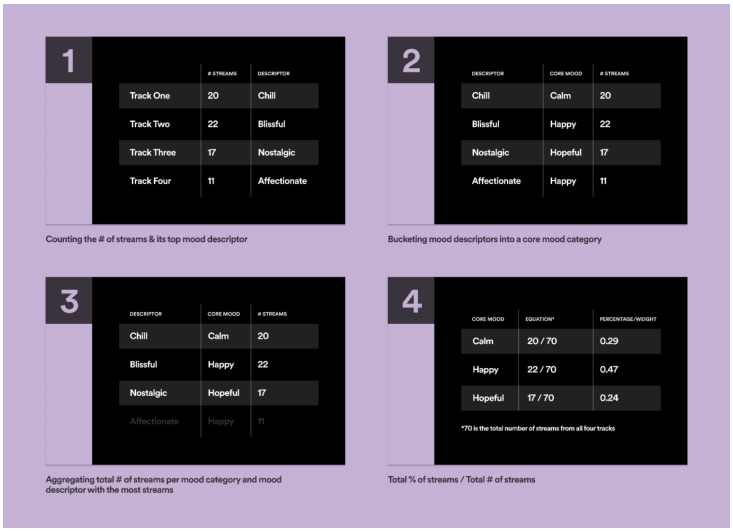


Figura 1.19: I quattro passaggi fondamentali per ottenere i mood e i relativi pesi.[40]

Appresi i mood da visualizzare e i relativi pesi, Il passo successivo è definire la resa grafica finale, a partire dai colori, scelti con la consulenza di Mystic Michaela secondo il seguente schema:

- Viola: mood passionale ed energetico
- Verde: mood calmo, analitico ed introspettivo
- Rosa: mood ottimista ed innocente
- Arancione: mood ribelle, audace e sicuro
- Giallo: mood motivato e concentrato
- Blu: mood triste ed emotivo

Per generare variabilità nei colori da inserire nell'immagine, si considera il peso associato al mood: più questo è presente nella storia di ascolto, dunque più peso ha, più scuro il colore risultante, e viceversa.

[40]Riferisi alla nota a piè di pagina 39.

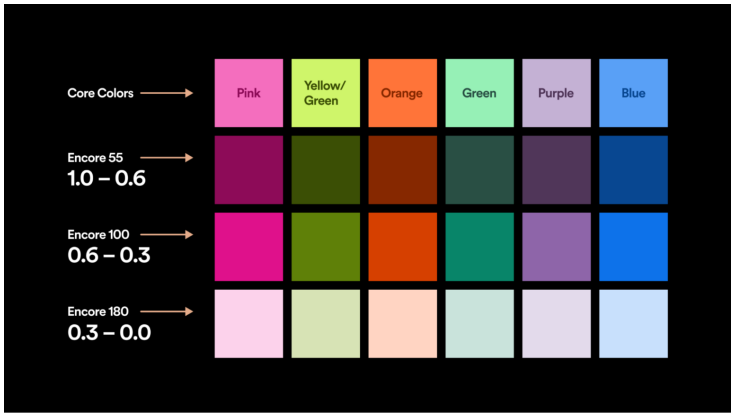


Figura 1.20: Associazione dei colori in base ai pesi.[41]

Il design finale è stato poi realizzato tramite uno degli elementi di design principali di Spotify Wrapped 2021 - il *ribbon* (nastro) - e tramite un effetto di sfocatura in modo tale da conferire al risultato finale un'aria eterea.

Figura 1.21: Creazione del design finale.[42]

Secondo Mystic Michaela, l'Audio Aura dà la possibilità alle persone di poter riflettere su se stessi, su chi si è, su cosa si sta cercando, sulle proprie abitudini e cosa queste stiano tentando di comunicare. Dopotutto, ascoltare musica è un'azione compiuta soli con sé stessi, non si devono giustificazioni e non si creano scuse per un momento totalmente personale, si compiono scelte totalmente proprie: l'Audio Aura è un mezzo per rilevare l'energia di una persona e i suoi desideri, ne è una vera e propria estensione.

[41]Riferisi alla nota a piè di pagina 39.
[42]Riferisi alla nota a piè di pagina 39.

È evidente come si possa creare qualcosa di puramente personale, quasi spirituale e mistico, a partire da una semplice raccolta di dati, una base puramente matematica ed analitica. Spotify Wrapped dunque è una vera e propria esperienza che va oltre il rendere il dato piacevole da vedere: i dati permettono di fare un'analisi sulle abitudini dell'utente e raccontare una vera e propria storia, la storia dell'utente attraverso la musica.

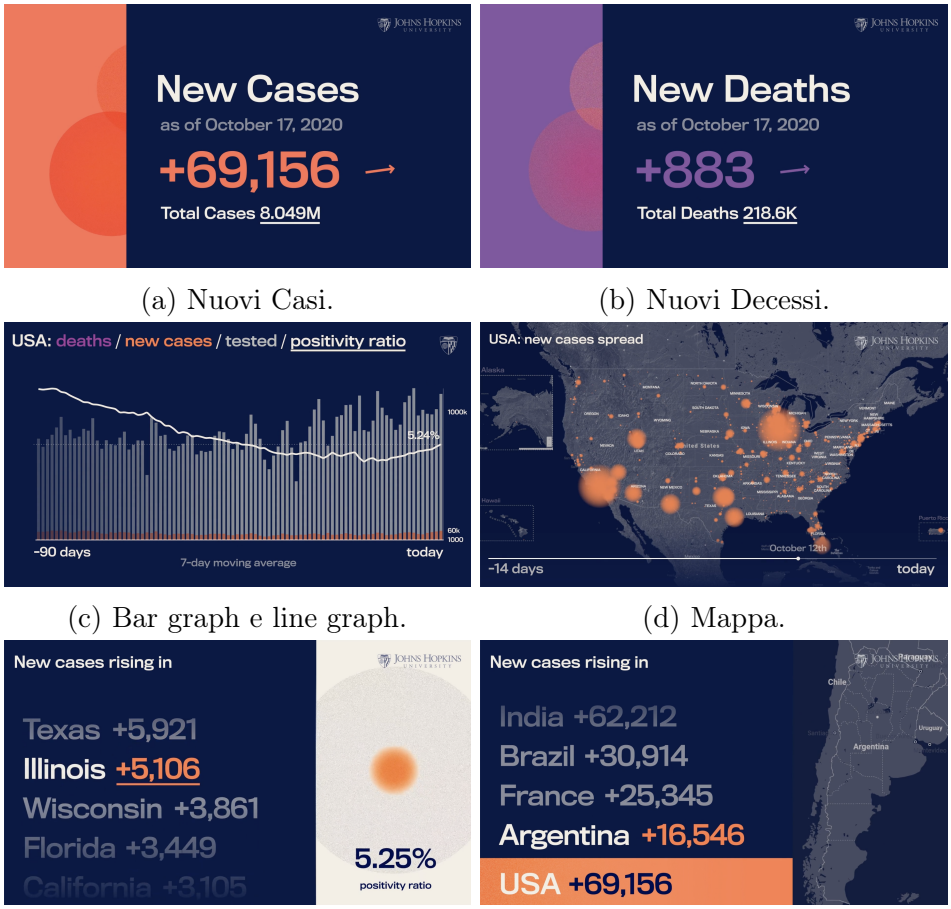
Algo: Johns Hopkins University - Daily Covid Tracker

Il secondo case study analizzato è una campagna video realizzata da Algo per la Johns Hopkins University (JHU) con il supporto di Bloomberg Philanthropies per il tracciamento dell'evoluzione della pandemia da Covid-19 negli Stati Uniti, Daily Covid Tracker. Lo scopo della campagna è quello di creare brevi video a cadenza giornaliera per fornire un aggiornamento sullo stato corrente dell'evoluzione della pandemia negli Stati Uniti, insieme ad una rapida overview dello stato globale del contagio. La campagna di video automation si basa su dati recuperati giornalmente ed automaticamente dal data feed open source fornito dalla JHU. I dati vengono elaborati ed integrati, e mostrati attraverso un video opportunamente personalizzato, affinchè si possano comunicare correttamente ed efficacemente le informazioni principali riguardanti l'evoluzione della pandemia giorno per giorno.

Ogni video è composto da una scena introduttiva, da una *call to action* finale, e da una serie di moduli di comunicazione dei dati. I moduli sono i seguenti:

- Nuovi Casi: modulo che mostra i nuovi casi rispetto al giorno precedente, ed il numero di casi totali (Fig.1.22(a)).
- Nuovi Decessi: modulo che mostra i nuovi decessi rispetto al giorno precedente, ed il numero di decessi totali (Fig.1.22(b)).
- Bar Graph e Line Graph riassuntivo: modulo che mostra tramite una serie di bar graph successivi i dati riguardanti il numero di persone testate, i nuovi casi ed i decessi, e tramite un line graph il tasso di positività (Fig.1.22(c)).
- Mappa: modulo che mostra progressivamente i dati di diffusione dei nuovi casi nei diversi stati per i 14 giorni precedenti fino al giorno corrente (Fig.1.22(d)).
- Aumento dei Nuovi Casi - Stati: modulo che mostra gli stati che hanno registrato un maggior aumento dei nuovi casi ed il relativo tasso di positività (Fig.1.22(e)).

- Aumento dei Nuovi Casi - Globale: modulo che mostra i Paesi che hanno registrato un maggior aumento dei nuovi casi (Fig.1.22(f)).



(a) Nuovi Casi. (b) Nuovi Decessi.

(c) Bar graph e line graph. (d) Mappa.

(e) Nuovi Casi - Stati. (f) Nuovi Casi - Globale.

Figura 1.22: Moduli di visualizzazione dei dati.[43]

Il video si modifica adattando i diversi grafici, le mappe e i campi testuali in base alle informazioni recuperate dalla sorgente dati, ma non solo: il design e la logica di personalizzazione sono stati pensati affinché l'aspetto del video risultante possa riflettere il *significato* dei dati sul quale è basato, così da fornire uno strumento di comunicazione rapido ed efficace. In particolar modo, la palette di colori, le forme e i *clue* sonori sono gli elementi audiovisivi che mutano in base al significato estrapolato dai dati.

[43] Algo. *Johns Hopkins University - Daily Covid Tracker*. [Online; ultimo accesso 25/02/2023]. URL: `https://algo.tv/jhu-covid-tracker`

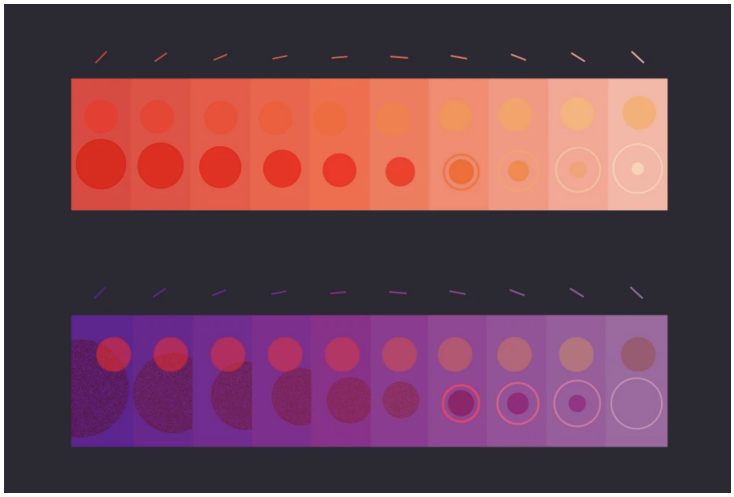


Figura 1.23: Adattamento degli elementi della scena in base ai dati.[44]

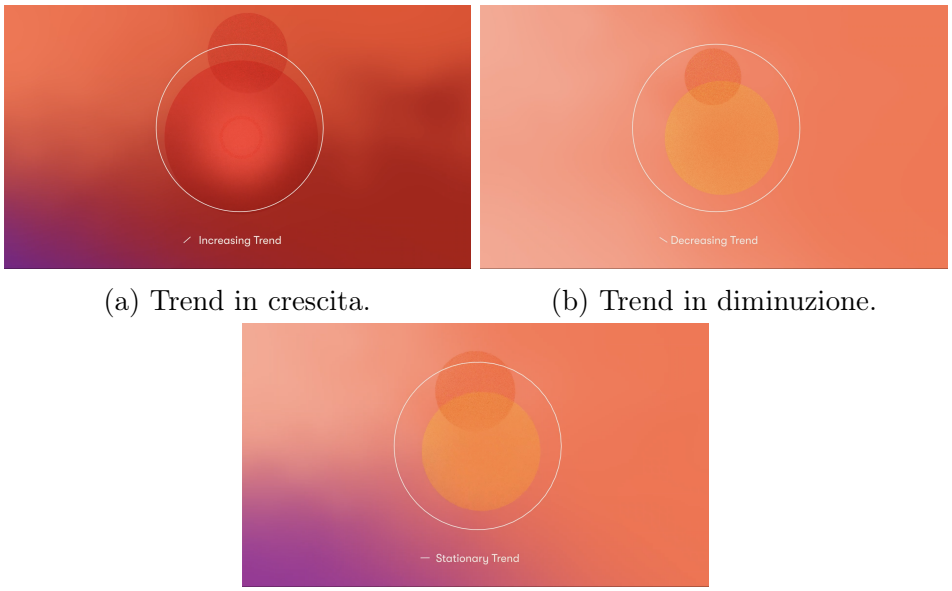
Si considera la personalizzazione dei colori. Le due categorie di dati principali su cui si basano i diversi moduli del video sono l'incremento di casi e decessi registrati, a cui vengono associati rispettivamente il colore arancione ed il colore viola.

Questi colori reagiscono e mutano in base ai dati. La tonalità specifica del colore cambia in modo tale da fornire - oltre al dato scientifico (il numero o il grafico di riferimento) - anche un *clue* visivo di immediata comprensione che permetta allo spettatore di assimilare efficacemente le informazioni. Se i nuovi casi sono in crescita, il colore arancione associato diventa più intenso e virante al rosso, se in diminuzione diventa più tenue e virante al rosa. Lo stesso avviene per il colore viola dei nuovi decessi.

Figura 1.24: Adattamento del colore in base ai dati.[45]

[44]Riferirsi alla nota a piè di pagina 43.
[45]Riferirsi alla nota a piè di pagina 43.

I colori non sono l'unica componente del video che si adatta ai dati, anche le forme e i suoni variano per poter rafforzare le informazioni con elementi audiovisivi positivi, negativi o neutrali. In questo modo non si rende solo la metabolizzazione delle informazioni più rapida e semplice, ma ci si accerta anche che ogni video abbia un design data-based risultante unico.



(a) Trend in crescita. (b) Trend in diminuzione.

(c) Trend stabile.

Figura 1.25: Visualizzazione dei trend. [46]

Il modo in cui un dato viene visualizzato può fornire dunque informazioni aggiuntive rispetto a quelle del dato stesso. Per questa ragione, un'altra scelta di design su cui è stata posta l'attenzione è l'espediente grafico di rappresentazione dei dati (nelle scene prive di una rappresentazione standard come può essere un bar graph o line graph). Questi vengono raffigurati con dei semplici cerchi dotati di una texture animata rumorosa. La scelta è stata fatta in modo che l'effetto risultante evochi una moltitudine di persone che si muove randomicamente, ed anche per simulare l'effetto di un'immagine vista al microscopio. Inoltre i bordi delle forme sono stati sfumati per enfatizzare la natura fluida delle informazioni e quella sfuggente del virus: altri casi possono esistere e diffondersi nonostante gli accurati processi di monitoraggio e testing.

[46] Riferirsi alla nota a piè di pagina 43.

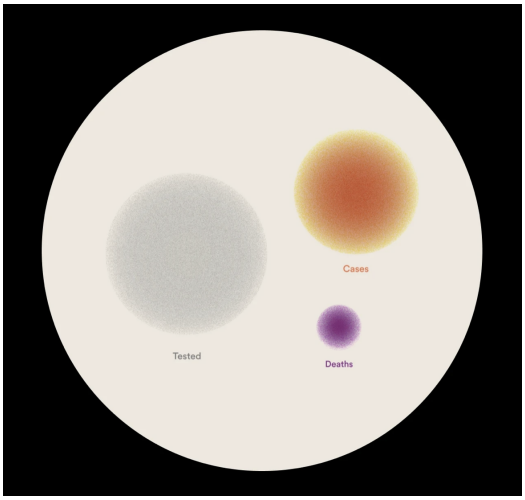


Figura 1.26: Rappresentazione grafica dei dati.[47]

Il video è composto da moduli che mostrano figure, mappe, grafici ed aree, in modo da poter rappresentare i dati analizzati secondo prospettive diverse (temporali, geografiche e di paragone con altri dati). La visualizzazione di questi dati è resa particolarmente efficace dal motion design. Per esempio, il modulo della Mappa permette di poter comprendere l'evoluzione della diffusione del virus negli Stati Uniti nelle ultime due settimane: il video guida lo spettatore giorno per giorno, mostrando l'evoluzione dell'intensità del contagio nei diversi stati nel tempo.

La creazione e la distribuzione del video sono processi automatici: ogni mattina l'infrastruttura di Algo crea autonomamente un video e lo carica direttamente nelle piattaforme di output scelte (il sito web Johns Hopkins Covid Resources, ed il canale Twitter e Youtube della Johns Hopkins University), il tutto senza alcun intervento umano.

Daily Covid Tracker è dunque un ottimo esempio di utilizzo della video automation per la data visualization: non solo permette di automatizzare il processo di produzione, così da fornire ogni giorno le informazioni aggiornate senza dispendio aggiuntivo di tempo o risorse, ma offre anche, tramite un design pensato, l'utilizzo della motion graphics ed una logica di personalizzazione curata, un livello di interpretazione dei dati aggiuntivo, che rafforza e rende l'assimilazione delle informazioni efficace e quasi immediata.

[47]Riferirsi alla nota a piè di pagina 43.

Capitolo 2

Processi per la Produzione Automatizzata di Video Personalizzabili

La creazione automatizzata di video personalizzabili può avvenire secondo diversi processi. Questi condividono, generalmente, alcuni step fondamentali:

- la creazione di un video-template generico
- la raccolta ed elaborazione dei dati sui quali avverrà la customizzazione
- la personalizzazione tramite scripting del template, in base ai dati di riferimento
- la renderizzazione del video risultante

Lo scopo di questo processo è quello di ottenere il risultato finale - un video adattato ai dati presi in considerazione - quanto più velocemente ed efficacemente possibile, senza dover modificare manualmente il template generale. È chiaro, dunque, come sia necessario strutturare dei processi di produzione specifici per raggiungere tale obiettivo.

Nell'ambito della tesi si fa riferimento all'esperienza di tirocinio svolta in Algo, dunque si analizzeranno i processi di produzione per la video automation utilizzati dall'azienda. I metodi analizzati si basano sull'utilizzo di After Effects[1], poiché è il software da loro impiegato per la creazione di video in motion graphics.

[1] After Effects: `https://www.adobe.com/products/aftereffects.html`

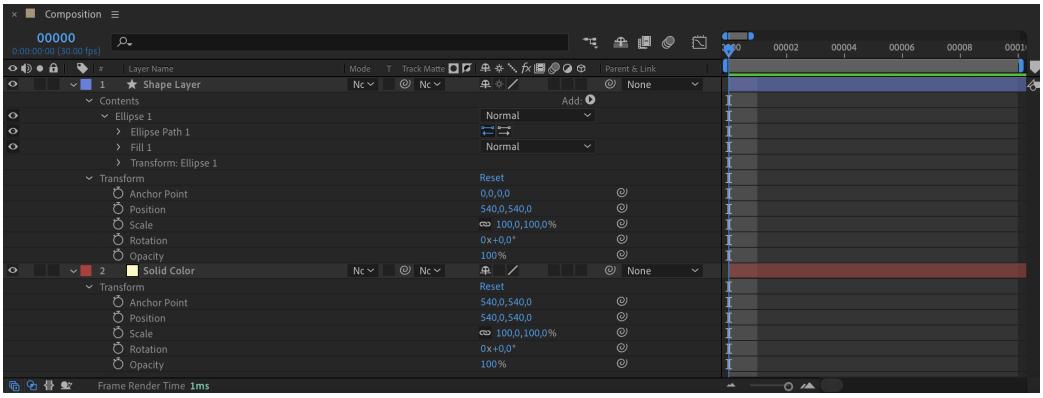
2.1 La struttura base di un progetto After Effects

Per trattare i processi di produzione presi in considerazione, è necessario conoscere la struttura base di un progetto After Effects.

Un progetto è definito come un insieme di *composition* (composizione), ognuna rappresentante un singolo video. Sulla composition si definiscono le impostazioni globali del video risultante, tra cui le dimensioni, il framerate e la durata. Le composition sono raccolte di *layer* (livelli). Le tipologie di layer sono molteplici, ognuna con scopi e caratteristiche differenti. Le principali sono:

- Audio, Video o Image Layer, ovvero livelli che contengono rispettivamente risorse audio, video o immagine.
- Solid Layer, ovvero livelli di colore.
- Livelli che contengono elementi grafici creati su After Effects: Shape Layer, i quali contengono forme o linee, e Text Layer, che contengono linee o paragrafi di testo.
- Precomposition Layer, o Precomp, ovvero composition esterne importate nella composition corrente come un unico livello.
- Livelli che non hanno render grafico ma servono come supporto, come gli Adjustment Layer e Null Layer.

L'aspetto e il comportamento dei layer è definito da una serie di proprietà. Alcune proprietà sono presenti in ogni layer, a prescindere dalla tipologia. Queste sono le proprietà di *transform*: la posizione del layer all'interno della composition, la sua rotazione, la sua scala, la sua opacità, e la posizione del suo *anchor point* (punto di ancoraggio).



(a) Composition: pannello dei layer

(b) Composition: resa grafica

Figura 2.1: Composition in After Effects formata da uno shape layer e un solid layer.

Altre proprietà sono invece specifiche per ogni tipologia di layer. Per esempio, gli Shape Layer presentano una serie di proprietà che permettono di cambiare l'aspetto delle forme che rappresentano, come il loro colore o dimensione, mentre i Text Layer presentano proprietà che permettono di cambiare l'aspetto del testo che rappresentano, come il font utilizzato o il font-size. Ad un livello possono poi essere applicati degli effetti. Gli effetti operano in maniera non distruttiva per modificare l'aspetto dei livelli, e agiscono in modo diverso in base alla configurazione delle proprietà che li definiscono.

After Effects viene utilizzato per creare video di motion graphics: le varie caratteristiche del video devono quindi poter cambiare nel tempo. In-

nanzitutto, è possibile cambiare la posizione del layer nella timeline della composizione, il suo *start time* (punto temporale di inizio) e la sua durata. Inoltre, la maggior parte delle proprietà dei livelli è animabile, ovvero se ne può definire il comportamento nel tempo tramite una serie di *keyframe* (fotogrammi chiave). Un keyframe indica che la proprietà deve assumere uno specifico valore in uno specifico istante di tempo (o in uno specifico frame). Inseriti due keyframe, uno di partenza e uno di arrivo, After Effects calcola automaticamente i valori che la proprietà deve assumere per ogni istante di tempo compreso tra i due keyframe. Il metodo di calcolo dei valori intermedi è influenzato dalla tipologia e dalle proprietà dei keyframe.

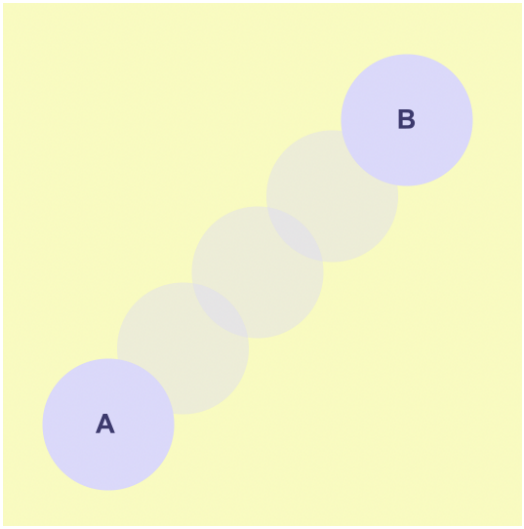


Figura 2.2: Posizioni intermedie occupate da uno shape layer animato dalla Posizione A alla Posizione B

Il render del video finale si ottiene renderizzando la composition corrispondente.

2.2 Produzione automatizzata di video: After Effects scripting in ExtendScript

Si analizza in questo paragrafo un processo di produzione di video personalizzati basato su operazioni di scripting supportate da After Effects.

2.2.1 L'automazione in After Effects: scripting in After Effects

After Effects offre nativamente servizi per automatizzare le operazioni di animazione: *expression* (espressioni) e supporto per gli script in ExtendScript.

Le expression sono porzioni di codice - in un linguaggio basato su JavaScript - che indicano ad una proprietà di un layer di eseguire una certa operazione. In questo modo, viene calcolato il valore di stato appropriato che deve assumere la proprietà - secondo quanto indicato dall'expression stessa - per ogni frame dell'animazione, senza l'inserimento esplicito di keyframe. Le expression permettono, evitando la creazione esplicita di keyframe, di automatizzare delle azioni e di risparmiare una notevole quantità di tempo. Con le expression è possibile realizzare una moltitudine di operazioni: collegare proprietà di diversi layer, cambiare una proprietà in base al tempo, creare loop, aggiungere casualità ad una proprietà, o governare una proprietà con delle funzioni matematiche. Si riportano in seguito alcuni esempi di expression.

- Esempio 1: loopOut().
  Definita un'animazione su una proprietà con due keyframe, uno di inizio e uno di fine, si può rendere tale animazione ciclica aggiungendo sulla proprietà l'expression loopOut().

- Esempio 2: wiggle(frequency, amplitude).
  Si può aggiungere un'animazione casuale su una proprietà con l'expression wiggle(frequency, amplitude). *Frequency* indica quante volte al secondo il valore della proprietà cambia, *amplitude* indica il massimo offset concesso - in positivo e in negativo - rispetto al valore della proprietà.

- Esempio 3: dipendenza da un'altra proprietà e valueAtTime (lookUpTime).
  È possibile collegare la proprietà di un layer ad un'altra proprietà (eventualmente di un altro layer), semplicemente inserendo come expression il riferimento alla proprietà da imitare. Il valore restituito da questo riferimento può a sua volta essere elaborato tramite semplici funzioni matematiche o tramite funzioni offerte da After Effects. Per esempio, è possibile applicare un delay tra la proprietà di riferimento e la proprietà target valutando la proprietà di riferimento su valueAtTime(time - delay), ovvero assegnando alla proprietà target, per ogni istante di tempo *time*, il valore che la proprietà di riferimento possiede nel tempo (time-delay).

Le expression, tuttavia, presentano dei limiti. Innanzitutto, poiché queste risiedono ed agiscono direttamente all'interno dell'applicazione, per poter variare una proprietà sulla base di un dato, è richiesta la presenza del dato - quindi del file che lo contiene - all'interno del progetto in After Effects. Inoltre, alcune operazioni più complesse non sono realizzabili tramite expression, operazioni quali la creazione, la rimozione, la modifica nel tempo e nel valore di keyframe, l'elaborazione dell'audio sulla sua forma d'onda, l'elaborazione avanzata di testi, il cambio di sorgente per immagini e video, calcoli complessi e la creazione modulare di video.

Per queste ragioni, è opportuno rivolgersi ad un altro supporto nativo di After Effects per l'automazione: il supporto per l'esecuzione di script. Gli script indicano all'applicazione un insieme di operazioni da eseguire, e tramite il loro utilizzo è possibile automatizzare operazioni eseguibili su After Effects, compiere calcoli complessi, sfruttare funzioni e variabili per personalizzare e modificare opportunamente le proprietà del progetto, ed anche eseguire operazioni non disponibili direttamente per interfaccia grafica. Gli script After Effects sono in estensione *.jsx* ed utilizzano una forma estesa di JavaScript, Adobe ExtendScript.

Dal punto di vista dello scripting, un progetto After Effects viene rappresentato con una struttura ad oggetti: gli elementi di interesse - come composition, layer, proprietà, immagini, video, maschere o effetti - sono codificati come oggetti ExtendScript, dotati di specifici metodi e proprietà. Sono disposti, e dunque accessibili, tramite una struttura gerarchica che emula la struttura presente nel progetto in After Effects. Se, per esempio, si vuole accedere ad una proprietà definita su un layer del progetto, è necessario prima recuperare la composition di riferimento, il livello di interesse nella composition, ed infine la proprietà all'interno del livello, secondo il seguente schema:

```
app.project.item(index).layer(index).property
```

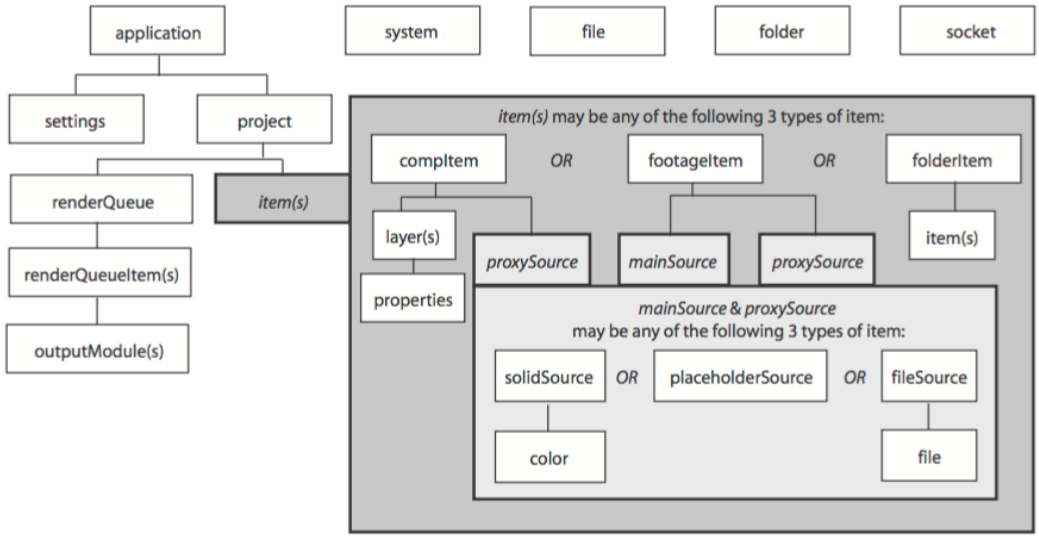


Figura 2.3: Struttura gerarchica a oggetti di un progetto After Effects. [2]

Una volta recuperato l'oggetto-proprietà di interesse si possono utilizzare i metodi forniti da ExtendScript per manipolare e modificare il comportamento di questo elemento all'interno del video.

Si riporta in seguito un esempio.

Si vuole assegnare come testo di un text layer un valore presente in un oggetto JSON, accessibile tramite la sua chiave *title*.

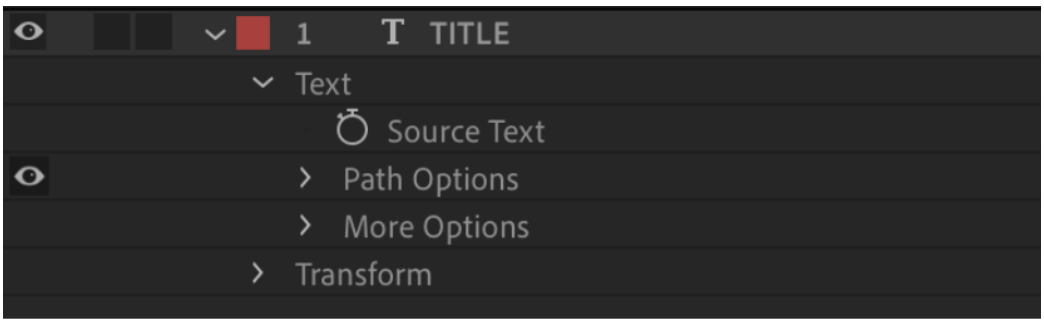


Figura 2.4: Text layer.

```
app.project.item(1).layer("TITLE").property("Text").
    property("Source Text").setValue(json.title);
```

[2]Diagramma gerarchico dei principali oggetti di scripting After Effects. [Online; ultimo accesso 12/02/2023]. URL: `https://ae-scripting.docsforadobe.dev/introduction/objectmodel.html`

Premesso che la composition sia il primo elemento nella repository del progetto, si accede ad essa tramite `app.project.item(1)`. Si accede al livello della composition tramite il suo nome, con `app.project.item(1).layer("TITLE")`. Per accedere alla proprietà Source Text, è necessario accedere alla proprietà che la contiene, ovvero la proprietà Text. Una volta recuperata Source Text con `app.project.item(1).layer("TITLE").property("Text").property("Source Text")`, è possibile assegnare ad essa un nuovo valore, tramite il metodo *setValue(value)*, dove value in questo caso è uno dei campi di un oggetto JSON (memorizzato nella variabile *json*).

Le informazioni sono strutturate in modo tale da essere il più simili possibili alla struttura del progetto in After Effects: è dunque chiaro che sia necessario conoscere adeguatamente After Effects, la sua interfaccia grafica, e il progetto creato, per poter eseguire operazioni di scripting.

La manipolazione di un progetto tramite script permette di automatizzare il processo di customizzazione di un video. Nello script JSX è possibile accedere alle proprietà di interesse del progetto e modificarle opportunamente in base a dei dati esterni, e questo processo di personalizzazione avviene semplicemente eseguendo lo script: il motion designer non ha la responsabilità di modificare manualmente il progetto ogni volta che un nuovo video deve essere realizzato.

La customizzazione tramite scripting richiede un processo di testing accurato ed esaustivo, e a meno che non si esegua ogni volta lo script, non è possibile vedere modifiche in real time sul template. In un contesto reale, dunque, si tende a sfruttare sia l'utilizzo di expression, sia lo scripting. Un esempio di una struttura ibrida che gode dei vantaggi dei due metodi consiste nel collegare tramite expression il comportamento di una proprietà di interesse ad una proprietà "di controllo", impostando in questa fase le eventuali elaborazioni matematiche necessarie. Si modifica poi tramite script il valore della proprietà di controllo coerentemente con i dati rilevati.

Si riporta un esempio: si considerano tre shape layer distinti, ed un valore di riferimento che viene fornito da un oggetto JSON di dati esterno al progetto. La scala del primo layer deve essere uguale a questo valore, la scala del secondo layer deve essere la metà, la scala del terzo layer un terzo. Si può creare un layer di tipo Null, chiamato Null Layer, ed aggiungere a questo layer un Expression Control di tipo Slider Control. Lo Slider Control è un effetto di After Effects che consiste di uno slider al quale si può assegnare un valore decimale. Si associano alle proprietà di scala degli shape layer le seguenti expression.

```
temp = thisComp.layer("Null Layer").effect("Slider Control")(
    "Slider");
[temp, temp]

temp = thisComp.layer("Null Layer").effect("Slider Control")(
    "Slider")/2;
[temp, temp]

temp = thisComp.layer("Null Layer").effect("Slider Control")(
    "Slider")/3;
[temp, temp]
```

Il primo layer ha scala pari al valore dello slider control, il secondo metà, il terzo un terzo. A questo punto, se varia il valore dello Slider Control, la scala dei layer varia nel modo desiderato. Lo script ha dunque il compito di modificare, in maniera appropriata e coerente con i dati letti, unicamente il valore dello slider control.

```
    app.project.item(1).layer("Null Layer").property("Effects
        ").property("Slider Control").property("Slider").
        setValue(json.value)
```

2.2.2 Pipeline di produzione

Il processo di produzione basato sullo scripting su After Effects si compone, essenzialmente, dei seguenti step:

- La creazione del template del video su After Effects
- La raccolta e l'elaborazione dei dati di personalizzazione
- L'esecuzione di uno script che personalizzi il template in base ai dati raccolti
- Il render del video risultante in una sequenza di PNG
- La conversione della sequenza di PNG ottenuta in un video MP4

Queste operazioni vengono eseguite su una macchina virtuale denominata *controller*. Si descrivono in maniera dettagliata le seguenti fasi.

Creazione del template

Il template After Effects rappresenta il video pre-customizzazione. Su questo template si definisce l'aspetto dei diversi elementi grafici e sonori, e le opportune expression sulle proprietà di interesse.

Raccolta ed elaborazione dei dati

La produzione automatizzata di video personalizzati ha come punto di partenza la raccolta di dati, poiché sono i dati che determinano l'aspetto e l'unicità del singolo video. Questa avviene in modo diverso in base alla natura del video. I dati possono essere raccolti da un database, tramite una chiamata ad una API, o possono essere inseriti esplicitamente tramite input di un utente; nel primo caso la raccolta dei dati è automatica e senza intervento dell'utente, nel secondo caso, invece, è necessario fornire un'interfaccia che permetta al cliente di poter customizzare e comporre il video come preferisca. È possibile prevedere anche una modalità ibrida, in cui il video viene personalizzato sia tramite i dati forniti da una API, sia tramite l'input di un utente.

In ogni caso rimane necessario raccogliere, all'inizio del processo, quei dati che definiranno in che modo verrà realizzato e presentato il video. I dati nativi raccolti devono essere normalmente sottoposti ad un processo di elaborazione, perché possano essere uniformati in una struttura coerente e regolare. Poichè ExtendScript è un'estensione di JavaScript, i dati vengono riorganizzati in un oggetto JSON, in modo da essere facilmente manipolabili da chi dovrà gestire la customizzazione del prodotto.

Esecuzione dello script JSX

Una volta ottenuti i dati di riferimento nel formato corretto, si può passare alla customizzazione del video-template. L'elemento che si occupa di definire la customizzazione del video è lo script JSX. All'interno di questo file viene letto l'oggetto JSON dei dati di riferimento e vengono definite opportune funzioni di customizzazione che agiscono sulla base dei dati letti. Il progetto After Effects, indicato nello script, viene aperto automaticamente sul controller affinché avvengano le modifiche opportune. L'applicazione viene aperta in modalità non-UI, multiprocessing, e script - dove lo script che indica quali operazioni compiere è lo script JSX.

Render ed elaborazione del video

Una volta aggiornato il progetto secondo le funzioni e le operazioni definite dallo script di customizzazione, si può renderizzare il video ottenuto.

Un singolo video corrisponde ad una composition all'interno del progetto. Il render di una composition avviene avvalendosi di una *render farm*: il progetto viene preso in carico da una rete di molteplici computer, detti motori di rendering, all'interno dei quali sono installate versioni di After Effects abilitate unicamente al rendering. Sfruttare una render farm permette di ridurre

drasticamente il tempo di render di una composition, poiché distribuisce il carico di lavoro su più nodi. Non è possibile utilizzare più computer per renderizzare un video, dunque la composition viene esportata come una sequenza di immagini PNG che, solo in un passo successivo, verranno convertite in un file MP4.

Per eseguire il rendering di rete della composition, il progetto After Effects - sul quale sono state apportate le opportune modifiche definite dallo script JSX - viene salvato come una copia in una cartella speciale, la *watch folder*, insieme ai file sorgente necessari. Quando un progetto viene inserito in una watch folder, le composition nella cartella principale (a *root level*) del progetto vengono automaticamente aggiunte alla *render queue*. La watch folder è visibile e controllata dalle macchine di render nella render farm: in questo modo le macchine sono in grado rispondere nel momento in cui un progetto viene inserito all'interno di essa.

Nel rendering di rete una composition viene renderizzata come una sequenza di immagini: ogni macchina prende in carico il render di un frame. Affinché il render avvenga correttamente, è quindi necessario tenere sempre conto del prossimo frame che deve essere renderizzato. All'interno della cartella designata di output vengono inizializzati tanti file PNG a 0 kB quanti sono i frame che devono essere renderizzati (il numero dei frame deve essere opportunamente indicato in un file di testo): ogni qual volta una macchina di render si libera, prende in carico il render del primo file PNG a 0 kB che trova nella cartella di output.

Una volta terminato il render, viene salvata la copia del progetto generata dalla customizzazione in una cartella apposita, mentre il progetto globale viene chiuso senza essere salvato, così da mantenere il template video sempre intatto.

Infine si converte la sequenza di immagini ottenute in un video. In questa fase di conversione è anche opportuno aggiungere - nel caso sia necessario - l'audio. È possibile che l'audio rimanga invariato rispetto al video. Tuttavia, spesso anche l'audio richiede di essere sottoposto ad un processo di customizzazione: è possibile, per esempio, avere diverse tracce audio disponibili e sceglierne una sola, può essere necessario adattare la lunghezza della traccia audio alla lunghezza del video, oppure spostarla in base alla comparsa di una determinata feature visiva. Quindi se deve essere gestito (e customizzato) in After Effects, è possibile aggiungere esplicitamente l'export dell'audio alla render queue, in modo tale da ottenere un file risultante che abbia le tempistiche, il placement e le caratteristiche corrette e adattate al video personalizzato. Come ultimo step, si fa l'overlay dell'audio file con il video ottenuto.

2.2.3 Analisi delle prestazioni: i limiti del processo JSX-based

Il metodo analizzato permette di automatizzare il processo di render di video customizzati su specifici dati. Tuttavia, i progetti di video automation possono differire molto per quanto riguarda la quantità e la frequenza di video da produrre, la reattività e la tempestività richiesti per il render, e il formato finale dell'output. Il processo di produzione JSX-based può non essere la soluzione più adatta per alcune tipologie di progetti. Per quanto, infatti, velocizzi ampiamente la produzione rispetto ad un update e render manuale del video, prevede tempistiche che, a seconda delle richieste del progetto, possono considerarsi lunghe. Ogni video prodotto, infatti, prevede l'apertura di After Effects e l'export esplicito del video, procedura che talvolta, a seconda della complessità del video, può essere più o meno lunga. Inoltre, è difficilmente scalabile: le macchine di render coinvolte nel processo devono essere esplicitamente instanziate, e questa procedura richiede diversi minuti per essere compiuta. Se, per esempio, il progetto richiede la raccolta periodica di dati e la creazione e conseguente pubblicazione dei video da essi derivanti, le tempistiche del processo JSX-based non sono particolarmente problematiche. Se, invece, il video personalizzato deve essere presentato ad un utente in seguito all'inserimento di alcuni input, il tempo di render non può essere eccessivamente lungo, altrimenti, abituato alla tempestività di risposta nel web landscape, l'utente potrebbe iniziare a pensare che il processo non sia andato a buon fine, o potrebbe semplicemente ritenerlo troppo lungo e abbandonarlo. Un video che viene prodotto con le tempistiche indotte dal processo JSX-based può essere più che accettabile per un progetto e problematico per un altro. È chiaro dunque che, se per alcune tipologie di progetto questo processo risulta adeguato, per altre è necessario cercare soluzioni alternative che possano sopperire ai limiti esaminati. In particolare, dunque, è necessario cercare un metodo che possa essere scalabile e tempestivo, e che permetta potenzialmente di poter produrre più video contemporaneamente allo stesso costo. Il secondo metodo utilizzato da Algo e analizzato nel paragrafo successivo nasce per raggiungere questo scopo.

2.3 Produzione automatizzata di video: Lottie

Il secondo processo di produzione utilizzato da Algo si basa sull'utilizzo di Lottie, una libreria per web e mobile che permette di rappresentare delle animazioni After Effects esportate e rappresentate con un oggetto JSON.

2.3.1 Lottie

Lottie è una libreria open-source, sviluppata da Airbnb e disponibile per Web, iOS e Android, che permette di leggere e renderizzare in real time animazioni in formato JSON in una pagina web o su mobile. L'esportazione di una composition After Effects in un oggetto JSON compatibile con Lottie avviene tramite uno specifico plug-in, Bodymovin[3].

Lottie nasce per poter facilitare l'implementazione di animazioni in ambito web e mobile. Normalmente, le animazioni possono essere ricreate appositamente tramite codice, incluse in formato GIF o come sequenza di PNG. Questi metodi tuttavia presentano dei limiti.

La creazione di animazioni tramite codice è estremamente dispendiosa in termini di tempo, poiché il design presentato dall'animatore viene utilizzato unicamente come reference da parte del developer, che deve ricreare l'animazione interamente. Inoltre risulta difficilmente scalabile e poco adatta ad un contesto multi-platform, poiché richiede che le animazioni vengano esplicitamente sviluppate per ognuno dei supporti previsti.

Le GIF sono spesso pesanti - pesano circa il doppio di un Lottie-JSON[4], e, poiché prevedono compressione di tipo lossy, possono presentare problemi di qualità. Inoltre, non sono ridimensionabili senza perdita di qualità, per cui non è possibile utilizzare lo stesso file GIF su schermi a diversa risoluzione.

Le sequenze di PNG presentano una qualità maggiore delle GIF, poiché prevedono compressione di tipo lossless, ma a scapito della dimensione: arrivano a pesare fino a 30-50 volte il peso di un Lottie-JSON[5]. Inoltre, non sono ridimensionabili senza perdita di qualità, quindi anche in questo caso non è possibile utilizzare un'unica sequenza di PNG per schermi a diversa risoluzione.

Lottie nasce per offrire un'alternativa che superi i limiti principali dei metodi standard analizzati. In particolar modo, le animazioni Lottie risultano essere:

- Leggere: poichè sono file JSON, pesano circa qualche decina di kB e riducono notevolmente i problemi legati alla dimensione e al dispendio di risorse.

- Ridimensionabili: sono vector-based e dunque possono essere ridimensionate infinitamente senza sgranamento di pixel e perdita di qualità. Per questo motivo è sufficiente avere un unico file JSON che rappresenti l'animazione.

[3]Bodymovin: `https://aescripts.com/bodymovin/`

[4]`https://airbnb.io/lottie/#/`

[5]Riferisi alla nota a piè di pagina 4.

- Scalabili: una volta inclusa la libreria di Lottie - e una volta istanziato il player fornito dalla libreria - il file Lottie-JSON è unico attraverso le diverse piattaforme: web, iOS e Android.

Nell'ambito della tesi si esplorerà l'utilizzo di Lottie unicamente in un contesto web. Per integrare un'animazione in una pagina web con Lottie, è necessario includere nella pagina HTML la libreria di Lottie, la quale contiene il player che renderà possibile la visualizzazione, e disporre dell'animazione in formato Lottie-JSON. Il caricamento e lo start dell'animazione avviene chiamando la funzione *lottie.loadAnimation()*. Questa funzione accetta un oggetto-parametro sul quale è necessario definire:

- la variabile che contiene l'oggetto JSON dell'animazione o il path all'oggetto JSON.
- se l'animazione debba essere in loop.
- se l'animazione debba partire automaticamente al caricarsi della pagina.
- quale tipo di renderer utilizzare (SVG, Canvas o HTML).
- l'elemento della pagina HTML sul quale renderizzare l'animazione.

Una volta chiamata questa funzione, l'animazione sarà renderizzata sulla pagina web. Lottie rende inoltre disponibili una serie di funzioni ed operazioni sul player dell'animazione, grazie alle quali è possibile, per esempio, interrompere o distruggere un'animazione, cambiarne la velocità e la direzione, restringerla in un frame-range specifico, ridimensionarla, etc. È anche possibile aggiungere degli EventListener su specifici eventi dell'animazione, come per esempio sul suo completamento o sul completamento di un loop.[6]

2.3.2 Pipeline di produzione

Il processo di produzione basato su Lottie prevede i seguenti step:

- La creazione del template del video su After Effects
- L'esportazione del template in un Lottie-JSON
- La raccolta e l'elaborazione dei dati di personalizzazione

[6]Per approfondire le operazioni supportate si rimanda alla documentazione ufficiale di Lottie: `https://github.com/airbnb/lottie-web`

- La personalizzazione del Lottie-JSON secondi i dati raccolti
- L'eventuale conversione da web-animation a video MP4

Rispetto al metodo di produzione analizzato precedentemente, due step rimangono invariati: la creazione del video-template su After Effects, e la raccolta ed elaborazione dei dati in un file JSON.

Creazione del template ed esportazione in Lottie

Il primo step da compiere, per quanto riguarda l'animazione, è la realizzazione del video-template su After Effects. Lottie non offre supporto esaustivo per tutte le funzionalità e feature di After Effects, dunque è necessario confrontarsi con la documentazione ufficiale affinché le animazioni nel template vengano realizzate con metodi compatibili con Lottie. La composition del template viene poi esportata - senza aver fatto alcuna operazione di customizzazione - tramite Bodymovin in un Lottie-JSON. Il JSON che si ottiene rappresenta e racchiude le informazioni del video-template, non ancora customizzato.

Raccolta ed elaborazione dei dati

Questo step rimane invariato rispetto al metodo JSX-based.

Personalizzazione del Lottie-JSON

Una volta ottenuti i dati di riferimento, è possibile passare allo step di customizzazione. La customizzazione, in questo caso, non avviene sul progetto After Effects del template, bensì sul Lottie-JSON che lo rappresenta. Poiché tutte le informazioni e le proprietà che costituiscono il template sono racchiuse nel JSON, per ottenere una variazione coerente con i dati di personalizzazione, è sufficiente modificare direttamente quei campi del JSON che rappresentano le proprietà di interesse. Poiché si ha a che fare con un oggetto JSON, reso visibile in una pagina web, le operazioni di customizzazione sono definite in uno script JavaScript.

Render ed elaborazione video

Il Lottie-JSON dell'animazione customizzata viene integrato in una pagina web ed eventualmente reso visibile come web-animation. Se la natura del progetto da realizzare richiede che il video debba essere scaricato o postato su altre piattaforme - è molto comune che i video vengano postati su differenti

social network, per esempio - allora è necessario convertire il risultato ottenuto da web-animation (generata dal Lottie-JSON) ad un video MP4. Una possibile soluzione - quella adottata da Algo - è quella di caricare il Lottie-JSON post-customizzazione su un Canvas HTML5 nascosto e, tramite un comando offerto dalla libreria di Lottie, recuperare i frame che compongono l'animazione in formato SVG. Dopodichè si convertono i frame SVG in file PNG, e da questa sequenza di immagini si ottiene il video MP4.

Per quanto riguarda l'audio, questo non è nativamente supportato da Lottie. Tuttavia, Lottie permette l'integrazione con librerie audio esterne o audio player custom. Per gestire l'audio è possibile dunque customizzarlo modificando opportunamente il JSON - per esempio cambiando la durata o la sorgente audio - ed includere una libreria esterna per poterlo attivare in fase di visualizzazione del video sulla pagina web. Nel caso in cui il video debba essere poi scaricato, l'audio viene gestito tramite FFMPEG[7], un framework multimedia che serve per filtrare, codificare e decodificare file audio e video. La customizzazione dell'audio viene replicata tramite FFMPEG, al quale si delega anche l'overlay dell'audio sul video.

2.3.3 Analisi delle prestazioni: differenze tra i processi di produzione

Il processo Lottie-based e il processo JSX-based presentano alcune importanti differenze. Una delle differenze principali è il momento e il luogo in cui avviene la customizzazione del video. Con il framework JSX-based, affinché si possa ottenere un video customizzato, si deve modificare tramite scripting il progetto After Effects del video-template. Questo comporta sempre l'apertura del progetto e l'esportazione della composition analizzata, ogni qual volta si voglia produrre un video. Il metodo Lottie-based, invece, prevede che l'export avvenga una volta sola: all'inizio del processo. La customizzazione avviene andando a modificare il JSON dell'animazione e la sua visualizzazione lato web è pressoché immediata: si riducono i tempi di export, ed è necessario aspettare solamente che il Lottie-JSON venga ricaricato (un tempo di attesa quasi irrisorio). Nel caso, poi, in cui il video debba essere scaricato, si aggiunge il tempo del download. Inoltre, non presenta i problemi di scalabilità legati all'istanziamento delle diverse macchine di render del processo JSX-based: l'elaborazione della customizzazione è delegata al browser in cui viene visualizzata la Lottie-animation. È chiaro che in progetti che richiedono una rapida e vasta produzione di video il metodo Lottie-based sembra essere particolarmente adatto, poiché prevede un'elaborazione piuttosto snella,

[7]FFMPEG: `https://ffmpeg.org`

tempi di renderizzazione minori, e costi di mantenimento dell'infrastruttura limitati.

Lottie, tuttavia, presenta dei limiti. Uno di questi è il supporto limitato alle features di After Effects: vi sono infatti alcune proprietà, expression ed effetti che non vengono renderizzati, o che causano problemi e comportamenti inaspettati. Questo è un aspetto da considerare fin dalle fasi embrionali del progetto, sia per quanto riguarda il design iniziale, sia per quanto riguarda la realizzazione del template su After Effects. È necessario consultare la documentazione ufficiale e testare il progetto, in modo tale da realizzare la composition affinché il risultato visualizzato tramite Lottie sia uguale al template su After Effects.

Un'altra difficoltà imposta da Lottie risiede nel processo di customizzazione. Benché modificare direttamente il JSON abbia vantaggi dal punto di vista del testing, delle tempistiche e del calcolo, questo processo non è sempre diretto. Il JSON organizza le informazioni in maniera gerarchica, e dunque, in base alla complessità del template, può diventare piuttosto intricato da analizzare. Inoltre, le informazioni non sono spesso facilmente leggibili, sia per quanto riguarda il modo in cui queste vengono codificate, sia per quanto riguarda le chiavi attraverso le quali si rendono accessibili (queste chiavi sono solitamente composte da una o due lettere, e difficilmente intuibili). L'accesso e la modifica di un parametro lato JSX risulta essere piuttosto semplice: è necessario semplicemente conoscere la struttura del progetto e le funzioni messe a disposizione da Adobe. La stessa operazione lato Lottie è invece decisamente più dispendiosa in termini di tempo e risorse.

Da questa problematica nasce dunque una necessità: quella di poter costruire una serie di funzioni JavaScript che facilitino la manipolazione del JSON e che possano replicare il metodo di accesso alle informazioni e customizzazione presente nel metodo JSX-based.

Capitolo 3

Lo Sviluppo di una Libreria di Personalizzazione per Animazioni in formato Lottie

Uno degli step più dispendiosi in termini di tempo e risorse nel processo di produzione Lottie-based è quello della customizzazione dell'oggetto JSON che rappresenta l'animazione. La sua lunghezza e la sua complessità dipendono sicuramente dalla complessità del template After Effects, ma anche per i progetti più semplici districarsi tra le informazioni non è un compito banale.

Innanzitutto, le proprietà di interesse possono essere annidate e difficili da raggiungere all'interno dell'oggetto. Talvolta è anche difficile capire qual è la proprietà desiderata all'interno del JSON, sia perché le chiavi utilizzate sono poco esplicative (iniziali o sigle), sia perchè, oltre alle proprietà presenti su After Effects, vi sono anche campi relativi a informazioni di controllo utili per il player Lottie.

Inoltre, in base alla proprietà e al comportamento definito su After Effects, non è sempre possibile cambiarne il valore direttamente nel JSON. Talvolta il valore associato alla proprietà è codificato in maniera tale da non corrispondere numericamente al valore inserito su After Effects - valori a cui il motion designer è abituato. In altri casi, può accadere che il valore di un campo dell'oggetto dipenda dal valore di altri campi, e se si andasse a modificare uno di questi senza rispettare la dipendenza, si otterrebbe un risultato visivo errato. Si riporta un esempio per spiegare meglio la situazione.

Si considera la proprietà di opacità su uno shape layer. Se questa proprietà non è animata, basta recuperare la porzione dell'oggetto JSON relativa all'opacità del layer e cambiare opportunamente il valore.

```
"o": {
       "a": 0,
       "k": 100,
       "ix": 11
     }
```

```
"o": {
       "a": 0,
       "k": 10,
       "ix": 11
     }
```

Se, invece, l'opacità è animata e dunque presenta dei keyframe, cambiare le informazioni di questi keyframe non è così semplice e diretto.

```
"o": {
            "a": 1,
            "k": [
                {
                    "i": {
                        "x": [0.4],
                        "y": [1.083]
                    },
                    "o": {
                        "x": [0.3],
                        "y": [0]
                    },
                    "t": 0,
                    "s": [100]
                },
                {
                    "t": 15,
                    "s": [10]
                }
            ],
            "ix": 11
    }
```

Innanzitutto, a prima vista, non è subito chiaro a cosa si riferiscano i diversi campi di un keyframe, così come non è chiaro in che modo le informazioni riguardanti i keyframe impostate su After Effects vengano codificate nel JSON. I keyframe presentano poi un caso di dipendenza di informazioni: come si analizzerà in seguito, il valore del campo y dipende dal valore degli altri campi presenti. Ciò significa che se si volesse cambiare il valore di un keyframe non sarebbe sufficiente, come nel primo caso, identificare e modificare il valore d'interesse poiché il rapporto tra i campi cambierebbe, ottenendo un risultato visivo diverso da quello desiderato ed impostato in fase di progetto. Trovare la proprietà desiderata e modificarla non è dunque un'operazione banale.

Inoltre, vi sono una serie di operazioni che risultano essere difficili e laboriose da gestire sul JSON, come per esempio azioni di aggiunta o rimozione di keyframe.

Infine, avere un metodo simile a quello JSX-based di chiamata di funzioni di customizzazione risulterebbe in ogni caso molto pratico, poiché non richiederebbe di analizzare e conoscere il Lottie-JSON del singolo progetto: non sarebbero necessarie competenze approfondite riguardo Lottie e basterebbe conoscere la struttura della composition After Effects per poter chiamare le funzioni opportune.

È dunque per queste ragioni che è stata sviluppata, in sede di tirocinio, una libreria di funzioni di customizzazione per animazioni in formato Lottie-JSON.

Lottie supporta una serie vasta di proprietà di After Effects. Lo sviluppo della libreria è stato svolto nell'ambito di un tirocinio di tre mesi, dunque è stato necessario condurre un'analisi per scegliere quali proprietà avessero la priorità di essere gestite e supportate. La libreria è stata pensata per essere utilizzata specificatamente nella produzione di due progetti Lottie-based realizzati in sede di tirocinio ed analizzati nel capitolo successivo, e più generalmente nello sviluppo dei progetti tipicamente presi in carico da Algo, per cui sono state trattare le proprietà funzionali a questo scopo.

3.1 Analisi della struttura di un oggetto Lottie-JSON

Il primo step affrontato per la costruzione della libreria di customizzazione è stato lo studio e l'analisi della struttura generale di un Lottie-JSON, per comprendere in che modo fossero organizzate e codificate le informazioni.

L'oggetto JSON globale rappresenta una composition After Effects e mantiene la sua struttura gerarchica. Al suo interno sono presenti oggetti che rappresentano i livelli, e all'interno di questi oggetti che rappresentano le proprietà e gli effetti. Ognuno di questi oggetti presenta coppie chiave-valore che codificano le proprietà dell'elemento di interesse.

3.1.1 Composition

L'oggetto JSON globale corrisponde alla composition. Questo contiene le sue proprietà globali (quelle che su After Effects corrispondono alle *Composition Settings*), e gli elementi che risiedono al suo interno.

```
{
    "v": "5.9.0",
    "fr": 25,
    "ip": 0,
    "op": 1000,
    "w": 1080,
    "h": 1080,
    "nm": "Comp",
    "ddd": 0,
    "assets": [
            {...}
       ],
    "fonts": {
       "list": [
            {...}
       ]
    },
    "layers": [
            {...}
       ],
    "markers": []
}
```

I campi presenti che caratterizzano una composition sono:

- v: la versione utilizzata di Bodymovin.
- fr: framerate.
- ip: in-point della composition, ovvero l'inizio della composition, espresso in frame.
- op: out-point della composition, ovvero la fine della composition, espresso in frame.
- w: larghezza della composition, in pixel.
- h: altezza della composition, in pixel.
- nm: nome della composition.
- ddd: indica se all'interno della composition sono presenti livelli 3D.
- assets: array di oggetti che contengono informazioni riguardo risorse immagini, risorse audio o precomposition.
- fonts: oggetto che contiene un array dei font utilizzati nel progetto.
- layers: array di oggetti che rappresentano i livelli della composition.

- markers: array che contiene i markers della composition (elementi inseriti esplicitamente in After Effects che memorizzano metadati della composition).
- chars: array che contiene oggetti che rappresentano i singoli caratteri presenti in tutti i layer di testo della composition. Questo array è presente solamente se, in fase di export con Bodymovin, si sceglie di codificare il testo come una serie di glifi invece che come stringhe di testo. Poiché il testo è spesso customizzabile, solitamente si disabilita questa opzione, e l'array *chars* non è presente.

Si analizzano nello specifico alcuni elementi.

Assets

L'array *assets* contiene le informazioni riguardanti risorse immagini, audio e precomposition.

Risorse Immagini

```
    "assets": [
            {
                "id": "profile.jpg",
                "w": 500,
                "h": 500,
                "u": "images/",
                "p": "profile.jpg",
                "e": 0
            }]
```

- id: identificativo unico della risorsa immagine.
- w: larghezza dell'immagine, in pixel.
- h: altezza dell'immagine, in pixel.
- u: indica il path della cartella che contiene l'immagine, se inserita come file.
- p: indica il nome del file o l'url dell'immagine.
- e: indica se l'immagine è stata incorporata direttamente o meno nel JSON.

Tra i setting di export di Bodymovin, è possibile scegliere di incorporare le risorse direttamente nel JSON, ovvero di avere le immagini *embedded* nel Lottie (questa opzione è di default deselezionata). Nel caso l'immagine sia embedded, il flag *e* è ad 1 e cambiano di significato i campi *u* e *p*: il primo rimane vuoto, il secondo contiene i dati che codificano l'immagine.
Risorse Audio. Vengono codificate come le risorse immagini. Sono assenti i campi *w* e *h*.
Precomposition (o precomp). Le precomposition sono composition che vengono inserite come un unico livello nella composition corrente. In quanto tali queste riprendono la struttura dell'oggetto JSON globale, apportando però delle differenze.

```
"assets": [
    {
    "id": "comp_0",
    "nm": "PreComp",
    "fr": 25,
    "layers": [...]
    ]
```

Una precomp contiene le seguenti informazioni:

- id: identificativo unico della precomposition.
- nm: nome della precomposition.
- fr: framerate.
- layers: array di oggetti che rappresentano i layer all'interno della precomposition.

È importante notare che l'oggetto che identifica la precomp all'interno dell'array *assets* serve per memorizzare le informazioni di questo elemento come composition. La precomp, poi, viene inserita in un'altra composition, e in questo contesto diventa un livello. In quanto livello, il suo comportamento viene definito e memorizzato in un oggetto specifico nell'array di layer della composition globale. Il campo *id*, un identificativo assegnato da Bodymovin, serve per collegare il livello della precomposition al corrispondente oggetto presente in *assets*.

È possibile avere precomposition annidate, oppure immagini o audio all'interno di una precomposition. Tuttavia, esiste un unico array globale di risorse, definito sulla composition: tutte le informazioni che riguardano risorse annidate vengono definite allo stesso livello in *assets*.

Fonts

Fonts è un oggetto che contiene un array di chiave *list*. Questo contiene a sua volta oggetti che rappresentano tutti i font utilizzati all'interno della composition (e nelle precomp).

```
"fonts": {
    "list": [
        {
            "origin": 0,
            "fPath": "fonts/Circular Air Pro-Light.otf",
            "fClass": "",
            "fFamily": "Circular Air Pro",
            "fWeight": "",
            "fStyle": "Light",
            "fName": "CircularAirPro-Light",
            "ascent": 72.198486328125
        }
    ]
}
```

In ogni oggetto presente in *list* sono definite le caratteristiche che identificano un font:

- fPath: indica il path o l'url del font utilizzato. Di default questo campo è vuoto, ma può essere definito in fase di export su Bodymovin.
- fClass: classe CSS che si può assegnare al font. Di default questo campo è vuoto, ma può essere definito in fase di export su Bodymovin.
- fFamily: famiglia del font.
- fWeight: spessore del font. Di default questo campo è vuoto, ma può essere definito in fase di export su Bodymovin.
- fStyle: stile del font.
- fName: identificativo del font assegnato da Bodymovin. Serve per assegnare il font corretto ai livelli di testo della composition.
- ascent: ascent del font, ovvero la distanza tra la baseline e l'elemento più alto del font. Viene calcolato in fase di export.

Affinché il font venga caricato, è necessario fornire o la font-family o il nome della classe CSS da includere nell'elemento del DOM di visualizzazione.

3.1.2 Layer

Un layer di After Effects viene rappresentato nel JSON come un elemento dell'array *layers*, definito all'interno della composition (o di una precomposition in *assets*). I campi presenti all'interno di un livello possono variare molto in base alla sua tipologia, e in base agli effetti applicati su di esso. Vi sono tuttavia delle proprietà sempre presenti.

```
{
  "ddd": 0,
  "ind": 10,
  "ty": 3,
  "nm": "Layer",
  "td": 1,
  "sr": 1,
  "ks": {...},
  "ao": 0,
  "ip": 0,
  "op": 1026,
  "st": 0,
  "bm": 0
}
```

- ddd: indica se il livello è un livello 3D.
- ind: indice del layer in After Effects (questo valore parte da 1, associato al layer più in alto nello stack). È l'indice che viene utilizzato per imparentare un layer ad un altro. L'operazione di *parenting* (imparentamento) permette ad un livello di seguire automaticamente il movivimento definito nel *parent layer*.
- ty: indica la tipologia del layer, secondo il seguente schema:
  - 0 - Precomp Layer
  - 1 - Solid Layer
  - 2 - Image Layer
  - 3 - Null Layer
  - 4 - Shape Layer
  - 5 - Text Layer
  - 6 - Audio Layer

  Su After Effects sono presenti ulteriori tipologie di livello, tuttavia questi non sono supportati da Lottie.

- nm: nome del layer.
- sr: *time stretching* (dilatazione o contrazione del tempo) del livello, di default ad 1.
- ks: oggetto che contiene le informazioni riguardo le *transform*, ovvero le proprietà di posizione, rotazione, scala, opacità ed anchor point. Ogni oggetto transform contiene almeno i seguenti tre campi:
  - a: indica se la proprietà è stata animata.
  - k: valore della proprietà.
  - ix: indice della proprietà. Viene utilizzato nelle expression per indicarne il riferimento.

  Questa struttura è spesso presente nel Lottie-JSON, e d'ora in poi ci si riferirà ad essa come *struttura a-k-ix*.

Si indicano in seguito gli oggetti relativi ad ogni transform, considerandole nella loro forma statica, priva di keyframe.

Opacità

```
"o": {
   "a": 0,
   "k": 100,
   "ix": 11
  }
```

Il valore *k* è un numero da 0 a 100.

Rotazione

```
"r": {
   "a": 0,
   "k": 0,
   "ix": 10
 }
```

Il valore *k* è un numero. Su After Effects la rotazione si indica tramite un angolo compreso tra 0 e 359°, e il numero di cicli di rotazione. Nel Lottie si indica direttamente l'angolo totale.

Scala

```
"s": {
      "a": 0,
      "k": [100, 100, 100],
      "ix": 6,
      "l": 2
}
```

Il valore k è un array tridimensionale. Anche se il livello è 2D, l'array memorizza tutte e tre le dimensioni (lasciando nel caso la terza dimensione pari a 100).

Anchor Point

```
"a": {
      "a": 0,
      "k": [
          -387.018,
          -419.057,
          0
      ],
      "ix": 1,
      "l": 2
}
```

Il valore k è un array tridimensionale. Anche se il livello è 2D, l'array memorizza tutte e tre le dimensioni (lasciando nel caso la terza dimensione uguale a 0).

Posizione

La posizione su After Effects può essere gestita come unica proprietà, sulla quale si può modificare la coordinata sui tre (o due) assi, oppure come proprietà distinte: *X Position*, *Y Position* e *Z Position*. Nel JSON si ha una struttura differente in base alla gestione su After Effects.

```
"p": {
      "s": true,
      "x": {
          "a": 0,
          "k": -387.012,
          "ix": 3
      },
      "y": {
          "a": 0,
          "k": -419.085,
          "ix": 4
      }}
```

Il campo s indica se la posizione è stata separata. In caso di gestione degli assi distinta, si aggiungono i campi x e y (eventualmente anche z, se il livello è tridimensionale), ed ognuno di questi campi riprende la struttura classica della transform a-k-ix, nella quale k è un valore numerico.

```
"p": {
        "a": 0,
        "k": [
            540,
            540,
            0
        ],
        "ix": 2,
        "l": 2
    }
```

Nel caso in cui invece la posizione non sia stata separata, la transform presenta la struttura classica e k risulta essere un array tridimensionale. Anche se il livello è 2D, l'array memorizza tutte e tre le dimensioni (lasciando nel caso la terza dimensione uguale a 0).

Se su una delle transform viene applicata una expression, questa viene memorizzata ed indicata sotto un campo x.

La gestione delle transform - e più genericamente delle proprietà - nel caso in cui siano animate è complessa, e verrà trattata in seguito in un paragrafo dedicato.

- ao: indica se è stata applicata l'opzione di auto-orient lungo un path (quando si definisce un'animazione della posizione di un layer, è possibile aggiungere la proprietà di auto-orient e il layer si orienterà lungo il path dell'animazione).
- ip: in-point del layer, ovvero il momento in cui il livello diventa visibile, espresso in frames.
- op: out-point del layer, ovvero il momento in cui il livello non è più visibile, espresso in frames.
- st: start time del layer, ovvero l'inizio del livello, espresso in frames. Per un livello, in-point e start time rappresentano due informazioni diverse. Lo start time rappresenta l'inizio effettivo del livello, l'in-point rappresenta il momento in cui il livello diventa visibile. Questi due valori spesso coincidono, tuttavia se il livello viene tagliato all'inizio, allora questi due valori risultano differenti.
- bm: indica il *blending mode* (modalità di fusione) del livello, secondo il seguente schema:

  - 0: Normal (Normale)
  - 1: Multiply (Moltiplica)
  - 2: Screen (Scolora)
  - 3: Overlay (Sovrapponi)
  - 4: Darken (Scurisci)
  - 5: Lighten (Schiarisci)
  - 6: Color Burn (Colore Brucia)
  - 7: Soft Light (Luce Soffusa)
  - 8: Hard Light (Luce Intensa)
  - 9: Difference (Differenza)
  - 10: Exclusion (Esclusione)
  - 11: Hue (Tonalità)
  - 12: Saturation (Saturazione)
  - 13: Color (Colore)
  - 14: Luminosity (Luminosità)

Possono anche essere presenti altri campi come:

- parent: indica l'indice del parent layer.
- ef: oggetto che contiene le informazioni riguardanti gli effetti applicati sul livello.
- cl: nome del layer utilizzato come classe HTML nel renderer di tipo SVG o HTML.
- ln: nome del layer utilizzato come id HTML nel renderer di tipo SVG o HTML.
- td e tt: sono i campi che gestiscono i track matte. I track matte layer (come immagini, testi, shapes o video) vengono utilizzati come maschere di trasparenza per un altro layer. I track matte hanno diverse configurazioni:
  - Alpha Matte: il layer su cui si definisce il track matte utilizza come maschera il canale alpha del layer di riferimento.
  - Alpha Inverted Matte: il layer su cui si definisce il track matte utilizza come maschera lo spazio negativo del canale alfa del layer di riferimento.

  - Luma Matte: il layer su cui si definisce il track matte utilizza come maschera la luminosità del layer di riferimento.
  - Luma Inverted Matte: il layer su cui si definisce il track matte utilizza come maschera l'inverso della luminosità del layer di riferimento.

  Nel JSON *tt* è un campo presente nel livello su cui si applica il track matte (e che viene dunque mascherato). Questo può assumere due valori, 2 se il track matte è di tipo inverted, 1 altrimenti. Il campo *td*, invece, è definito sul layer di riferimento (ovvero quello che fornisce le informazioni per la maschera). Anche questo parametro può assumere due valori, 1 se il track matte è di tipo alpha, 2 se è di tipo luma.

- hasMask: valore booleano che indica se il livello presenta una maschera.
- maskProperties: contiene un array di oggetti che definiscono le maschere applicate al livello.
- hd: valore booleano che indica se il livello è nascosto. Se il livello è visibile, questo campo è di default assente.

Si descrivono in seguito le diverse tipologie di layer e in che modo differiscono dallo schema generale.

Null Layer

Un null layer non ha alcuna resa grafica, ma viene utilizzato come supporto ad altri layer (per esempio tramite parenting di livelli). Lo schema di un livello di tipo Null è uguale a quello generale.

Solid Layer

Un solid layer è un livello di colore uniforme. Lo schema di un livello di tipo Solid aggiunge alle proprietà già presenti nello schema generale i seguenti campi:

- sc: colore del Solid in formato HEX.
- sw: larghezza del Solid.
- sh: altezza del Solid.

Image Layer

Lo schema di un livello di tipo Image aggiunge alle proprietà già presenti nello schema generale un solo campo: *refId*. Questo serve per poter identificare nell'array di assets la corretta risorsa immagine associata al livello (si sceglie quell'oggetto di *assets* il cui campo *id* è uguale a *refId*). Il valore viene associato da Bodymovin in fase di export.

Precomp Layer

Lo schema di un livello di tipo Precomp aggiunge alle proprietà già presenti nello schema generale i seguenti campi:

- refId: analogamente all'Image Layer, *refId* serve per poter identificare la precomposition corretta nell'array *assets*.
- tm: indica un eventuale *time remapping* (rimappatura del tempo) sulla precomp.

Audio Layer

Lo schema di un livello Audio è leggermente diverso da quello generale, poiché sono assenti i campi relativi alla dimensione spaziale di un livello: *ks* (le proprietà di transform) e *ao* (auto-orient). Non possono essere applicate maschere e track matte. Inoltre non è possibile abilitare e disabilitare il livello tramite il valore di *hd*. Così come nei Precomp Layer e Image Layer, anche gli Audio Layer presentano un campo *refId* per recuperare la sorgente audio corretta dall'array *assets*.

Shape Layer

Uno shape layer contiene al suo interno una o una serie di forme vettoriali, ovvero una serie di *shape* (forme). Lo schema di un livello di tipo Shape espande quello generale. Per comprendere meglio quali sono le informazioni nel JSON e come vengono organizzate è utile osservare da cosa è costituito uno Shape Layer su After Effects.

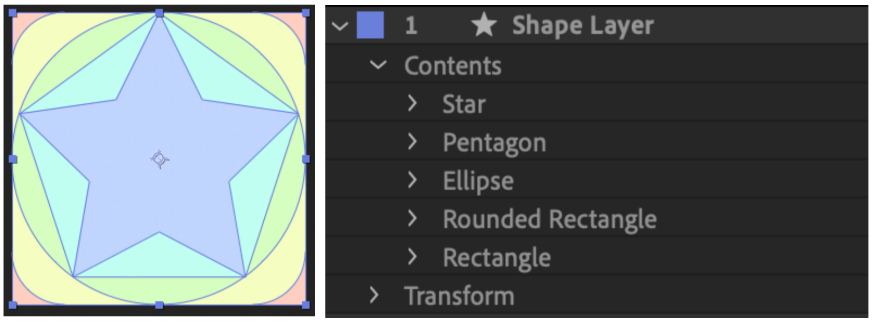
Le forme che After Effects mette di default a disposizione sono:

- Rectangle (Rettangolo)
- Rounded Rectangle (Rettangolo arrotondato)
- Ellipse (Ellisse)

- Polygon (Poligono)
- Star (Stella)

Ognuna di queste forme è caratterizzata da specifiche proprietà che ne definiscono l'aspetto, come il raggio per shape ellittiche o le dimensioni per shape rettangolari. In realtà, le tipologie il Rettangolo e Rettangolo Arrotondato rappresentano la stessa forma con valori diversi in una proprietà (la *roundness*, la percentuale di arrotondamento degli angoli) e si possono quindi considerare come un'unica forma. Lo stesso avviene per le tipologie Poligono e Stella: dalla una shape di tipo Stella è possibile ottenere anche un qualunque poligono regolare. Si possono quindi considerare tre forme globali: Rettangolo, Ellisse, e più generalmente Stella/Poligono. In alternativa è anche possibile definire shape *custom*, disegnandone i punti e definendo eventualmente maniglie di Bézier su di essi. Con questo metodo è possibile creare non solo shape chiuse, ma anche forme aperte.

Uno shape layer può contenere più forme all'interno di esso.



(a) Resa Grafica (b) Layer

Figura 3.1: Shape layer contenente diverse forme.

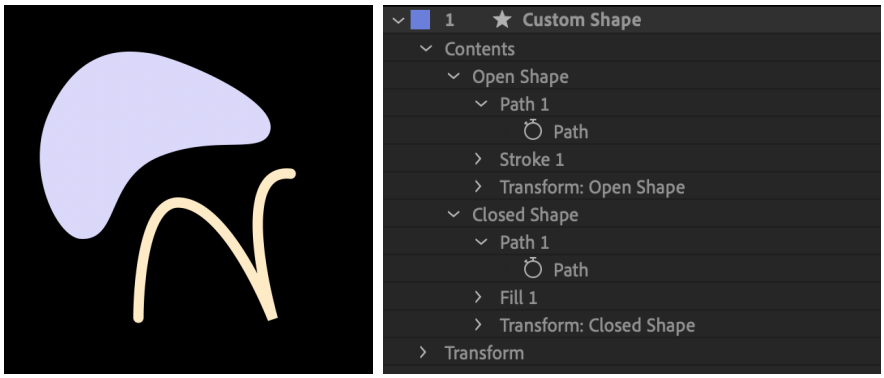
La struttura di uno shape layer si suddivide in due macrosezioni: *Transform*, che definisce le proprietà transform globali del livello, e *Contents*, che contiene le shape definite all'interno di esso. Ogni shape è definita da una serie di proprietà:

- Le proprietà di *path* (tracciato), definiscono la forma della shape, e sono differenti in base alla tipologia.
- Le proprietà di *fill* (riempimento), definiscono il colore interno della shape. Il fill di una shape può essere un colore unico, un gradiente o vuoto (assente). Se ne può anche definire l'opacità e la blending mode.

- Le proprietà di *stroke* (tratto), definiscono il contorno della shape. Lo stroke è ampiamente personalizzabile: se ne può definire lo spessore, il colore, l'opacità, la forma degli angoli e delle estremità, ed applicare su di esso stili particolari come il tratteggio.

- Le proprietà di *transform*, definiscono posizione, opacità, scala, rotazione, anchor point, *skew* (inclinazione) e *skew axis* (rotazione dell'asse di inclinazione) della shape. Queste influenzano solo la singola forma su cui sono definite, non il livello globale.

Un'attenzione particolare viene riservata alle shape custom, definite arbitrariamente disegnandone i punti. Queste mantengono inalterate le sezioni relative a fill, stroke e transform. La sezione del path non contiene proprietà se non *Path*, una proprietà animabile e non customizzabile da menu che racchiude, ma non rende visibili, le informazioni riguardanti i punti della shape.



(a) Resa Grafica (b) Layer

Figura 3.2: Shape layer di tipo custom contenente una forma aperta e una chiusa.

L'oggetto che rappresenta uno shape layer nel Lottie-JSON riprende la struttura gerarchica del rispettivo layer definito in After Effects. Oltre ai campi già analizzati per un generico layer, presenta un campo *shapes*. *Shapes* è un array che contiene oggetti che rappresentano le singole forme contenute nello shape layer. Questi oggetti hanno il seguente schema:

```
{
  "ty": "gr",
  "it": [{...}, {...},...],
  "nm": "Rectangle",
  "np": 3,
  "cix": 2,
  "bm": 0,
  "ix": 5,
  "mn": "ADBE Vector Group",
  "hd": false
}
```

- it: array di oggetti che contiene le informazioni di path, stroke, fill e transform della shape.
- nm: nome della shape.
- np: numero delle proprietà definite nella shape (questo campo viene utilizzato per le expression).
- bm: blending mode.
- ix: indice utilizzato per le expression.
- mn: *match name* di After Effects, utilizzato per le expression.
- hd: valore booleano che indica se la shape è nascosta o visibile. Se la shape è visibile, il valore è *false*. A differenza del corrispondente campo negli oggetti layer, qui questo valore è sempre presente.

Si analizza il campo *it*. Questo è un array che contiene, generalmente, gli oggetti che definiscono una shape in After Effects: path, stroke, fill e transform. Sul progetto After Effects è possibile eliminare le proprietà di stroke e fill (per eliminare il contorno o il riempimento di una shape), in questo caso gli oggetti relativi risultano assenti nel Lottie-JSON.

Si descrivono in seguito gli oggetti - potenzialmente - contenuti all'interno del campo it.

Path

L'array *it* presenta sempre un oggetto relativo alla proprietà di path della shape. La struttura di questo oggetto varia sensibilmente in base alla tipologia di shape scelta. Considerando le shape di default di After Effects, i campi sempre presenti sono:

- ty: definisce la tipologia di shape, secondo il seguente schema:

  - sr: può indicare sia una forma di tipo Stella, sia una forma di tipo Poligono. La distinzione tra le due avviene tramite un ulteriore campo, *sy*, presente solo in questi due casi: se *sy* è uguale ad 1 si tratta di una Stella, se è uguale a 2 un Poligono.
  - el: Ellisse.
  - rc: Rettangolo o Rettangolo Arrotondato, i quali hanno la stessa struttura Lottie (con valori differenti nella proprietà di arrotondamento degli angoli).

- d: indica la direzione del path che definisce la shape.
- nm: indica il nome del path.
- mn: *match name* di After Effects, usato per le expression.
- hd: indica se il path è nascosto o meno.

Dopodichè la struttura presenta diversi campi in base alla tipologia della shape. Questi campi corrispondono alle proprietà che definiscono la shape su After Effects, presenti nello stesso ordine. Sono sempre rappresentati da una struttura a-k-ix. Per esempio, una shape di tipo Rettangolo o Rettangolo Arrotondato ha un oggetto path di questo tipo:

```
{
    "ty": "rc",
    "d": 1,
    "s": {
        "a": 0,
        "k": [720,720],
        "ix": 2
    },
    "p": {
        "a": 0,
        "k": [0,0],
        "ix": 3
    },
    "r": {
        "a": 0,
        "k": 120,
        "ix": 4
    },
    "nm": "Rectangle Path 1",
    "mn": "ADBE Vector Shape - Rect",
    "hd": false
}
```

Dove i campi *s*, *p* e *r* corrispondono alle proprietà *Size* (dimensioni), *Position* (posizione) e *Roundness* (percentuale di arrotondamento degli angoli) che definiscono una shape rettangolare.

Un discorso a parte merita la shape di tipo custom, poiché il suo path non è definito attraverso proprietà esplicite, ma tramite i punti inseriti via interfaccia grafica su After Effects. L'oggetto path di una custom shape ha il seguente schema:

```
    {
        "ind": 0,
        "ty": "sh",
        "ix": 1,
        "ks": {
            "a": 0,
            "k": {
                "i": [[...], [...]],
                "o": [[...], [...]],
                "v":[[...], [...]],
                "c": false
            },
            "ix": 2
        },
        "nm": "Path 1",
        "mn": "ADBE Vector Shape - Group",
        "hd": false
}
```

Si può notare che vi sono alcuni campi in comune con le shape precedentemente analizzate, come il nome, il match name, il flag di visibilità, l'indice per le expression, e la tipologia di shape (la stringa corrispondente ad una shape custom è *sh*). La proprietà che definisce il path si trova sotto il campo *ks*, e benché questa presenti una struttura a-k-ix, è necessario analizzare nel dettaglio in che modo sono organizzare le informazioni sotto il campo *k*. L'elemento *k* è un oggetto con quattro campi:

- v: è un array che contiene al suo interno le coordinate dei punti che definiscono la custom shape, memorizzati come array bidimensionali $[x, y]$. È importante notare che le coordinate presenti non sono relative al sistema di riferimento globale della composition, ma sono calcolate in base all'anchor point della shape. Normalmente, questi sono valori numerici non visibili su After Effects, dove i punti vengono definiti e modificati spostandoli manualmente nello spazio della composition. È possibile definire esplicitamente le coordinate dei punti solo via expression.

- i: è un array che contiene al suo interno le coordinate degli in-point delle maniglie di Bezier definite sui punti, memorizzati come array bidimen-

sionali $[x, y]$. Per in-point si intende il punto nello spazio identificato dalla posizione della maniglia di Bézier di entrata definita su un punto. Nel caso non siano state definite maniglie di Bézier su un punto, l'in-point corrispondente risulta $[0, 0]$. Le coordinate sono memorizzate come offset orizzontale e verticale rispetto al punto a cui si riferisce l'in-point.

- o: array analogo ad *i*, riferito agli out-point. Per out-point si intende il punto nello spazio identificato dalla posizione della maniglia di Bézier di uscita definita su un punto.
- c: valore booleano che indica se la shape è chiusa o meno.

Stroke
La struttura varia leggermente se lo stroke è definito come un unico colore o come un gradiente di colori.

Si considera il caso in cui lo stroke sia definito da un unico colore.

```
{
    "ty": "st",
    "c": {...}
    "o": {
        "a": 0,
        "k": 100,
        "ix": 4
    },
    "w": {
        "a": 0,
        "k": 13,
        "ix": 5
    },
    "lc": 1,
    "lj": 1,
    "ml": 4,
    "bm": 0,
    "nm": "Stroke 1",
    "mn": "ADBE Vector Graphic - Stroke",
    "hd": false
}
```

- ty: costante uguale a *st*, serve per identificare l'oggetto come quello relativo allo stroke definito con un unico colore.
- c: se lo stroke ha un colore unico, *c* contiene le informazioni di colore. Ha una struttura a-k-ix. L'oggetto *k* contiene le informazioni di colore

ed ha una struttura $[r, g, b, a]$, dove r, g e b sono in scala 0-1 e dove a, che rappresenta la trasparenza, è sempre uguale a 1 (anche se si cambiasse, il colore apparirebbe sempre a trasparenza piena, poiché Lottie non supporta valori del canale alpha diversi da 1).

Nell'esempio sottostante, il colore indicato corrisponde alla stringa HEX #15B8EB.

```
        "c": {
                "a": 0,
                "k": [
                    0.082315998451,
                    0.720004990522,
                    0.924858003504,
                    1
                ],
                "ix": 3
            }
```

- o: indica l'opacità dello stroke, ed ha una struttura a-k-ix.
- w: indica lo spessore dello stroke, ed ha una struttura a-k-ix.
- lc: *line cap*, indica la modalità di chiusura dei punti all'estremità di un path aperto. Sono presenti tre diverse modalità.
- lj: *line join*, indica la modalità di render degli angoli formati dallo stroke. Sono presenti tre diverse modalità.
- ml: indica il *miter limit*, un parametro utile per definire il comportamento di una tipologia particolare di line join.
- nm: nome dello stroke.
- mn: *match name* dato da After Effects per le expression.
- hd: indica se è lo stroke è nascosto o meno.

Si considera il caso in cui lo stroke sia definito da un gradiente di colori.

```
{
    "ty": "gs",
    "o": {
        "a": 0,
        "k": 100,
        "ix": 9
    },
    "w": {
        "a": 0,
        "k": 66,
        "ix": 10
    },
    "g": {...}
    "s": {
        "a": 0,
        "k": [-274.641,0],
        "ix": 4
    },
    "e": {
        "a": 0,
        "k": [268.947,0],
        "ix": 5
    },
    "t": 1,
    "lc": 1,
    "lj": 1,
    "ml": 4,
    "ml2": {
        "a": 0,
        "k": 4,
        "ix": 13
    },
    "bm": 0,
    "nm": "Gradient Stroke 1",
    "mn": "ADBE Vector Graphic - G-Stroke",
    "hd": false
}
```

I campi differenti rispetto allo schema precedente sono:

- ty: costante uguale a *gs*, serve per identificare l'oggetto come quello relativo allo stroke definito con un gradiente di colori.
- g: racchiude le informazioni riguardanti i colori del gradient.

```
{
    "p": 3,
    "k": {
        "a": 0,
        "k": [0,0.983,1,0,
              0.5, 0.56,0.874,0.437,
              1,0.137,0.749,0.874],
        "ix": 8
    }
}
```

  - p: indica il numero di punti che definisce il gradiente.
  - k: oggetto a struttura a-k-ix che contiene i colori del gradiente. Se il gradiente è in bianco e nero, i punti che lo definiscono sono due, nei restanti casi sono invece tre: il colore di partenza, il colore di arrivo e il *midpoint*, ovvero il colore di transizione centrale. Ogni colore viene identificato nell'array k - all'interno dell'oggetto k - come un quartetto di valori in scala 0-1: *location*, canale r, canale g, canale b. Di default, la location dello start color è zero, del midpoint è 0.5, dell'end point 1. Un valore di location diverso sullo start point restringe il gradient verso l'end point e viceversa. Nell'esempio riportato, i due colori del gradient sono #FAFF00 ($r = 0.983$, $g = 1$, $b = 0$) e #23BFDF ($r = 0.137$, g=0.749, $b = 0.874$). Il colore del midpoint viene calcolato automaticamente da Bodymovin in fase di export.

- s: oggetto a-k-ix che indica lo start point del gradiente, ovvero il punto nello spazio della composition dal quale il colore inizia a cambiare.
- e: oggetto a-k-ix che indica l'end point del gradiente, ovvero il punto nello spazio della composition dal quale il colore termina di cambiare.
- t: indica la tipologia del gradiente. Questa può essere lineare - la transizione di colore avviene lungo una linea - o radiale - la transizione avviene secondo un pattern circolare. Se è lineare, allore è uguale a 1, se radiale a 2.

Lo schema analizzato è quello di un gradiente lineare. Un gradiente radiale presenta inoltre i seguenti campi:

- h: indica l'*highlight length* e ha struttura a-k-ix. Considerata la circonferenza il cui raggio è dato dalla differenza tra l'end point e lo start point, highlight length indica dove si dispone il punto più saturo del

primo colore rispetto al diametro della circonferenza (100% si dispone sull'end point, 0% si dispone sullo speculare dell'end point).

- a: indica l'*highlight angle* e ha struttura a-k-ix. Considerata la circonferenza il cui raggio è dato dalla differenza tra l'end point e lo start point, highlight angle indica l'angolo di rotazione del punto più saturo del primo colore all'interno di questa circonferenza. A differenza di come solitamente vengono memorizzate le informazioni radiali nel Lottie, il valore dell'angolo è moltiplicato per un fattore 360.

Fill
Anche il fill può essere definito con un unico colore o con un gradiente di colori, ed ha una struttura leggermente diversa nei due casi.

Si considera il caso in cui il fill sia definito tramite un unico colore.

```
{
        "ty": "fl",
        "c": {
            "a": 0,
            "k": [
                0.082315998451,
                0.720004990522,
                0.924858003504,
                1
            ],
            "ix": 4
        },
        "o": {
            "a": 0,
            "k": 100,
            "ix": 5
        },
        "r": 1,
        "bm": 0,
        "nm": "Fill 1",
        "mn": "ADBE Vector Graphic - Fill",
        "hd": false
    }
```

- ty: costante uguale a *fl*, serve per identificare l'oggetto come quello relativo al fill con colore unico.

- c: indica il colore e presenta la stessa struttura analizzata precedentemente per l'oggetto relativo allo stroke.

- o: indica l'opacità, ha struttura a-k-ix.
- r: indica la *fill rule*. Ha valore pari ad 1 se per il riempimento della shape viene presa in considerazione la direzione del path, ed ha valore pari a 2 altrimenti.
- bm: blending mode.
- nm: nome del fill.
- mn: *match name* dato da After Effects per le expression.
- hd: indica se è il fill nascosto o meno.

Si considera il caso in cui il fill sia definito da un gradiente di colori. Si hanno le seguenti differenze:

- ty è uguale a *gf*, per indicare che è un fill di tipo *gradient*.
- al posto del campo c sono presenti i campi visti ed analizzati precedenti nel caso di *gradient stroke*: g, per indicare i colori del gradiente, s ed e per indicare start point ed end point, t per indicare la tipologia del gradiente ed eventualmente h ed a per indicare highlight lenght e highlight angle.

Transform
Ogni shape all'interno di uno shape layer presenta le proprie proprietà di transform. Le transform riprendono la stessa struttura delle transform già analizzate per i layer, con alcune differenze:

- La posizione è sempre unita, poiché non è possibile separarla in After Effects.
- I valori di scala, posizione e anchor point sono bidimensionali, e non tridimensionali.

Su After Effects, è possibile aggiungere degli "effetti" specifici agli Shape Layer, non raggiungibili attraverso il panel degli effetti globali. Non vengono analizzati in questa sede poiché non sono stati gestiti dalla libreria di customizzazione (spesso infatti il comportamento di questi effetti viene definito su After Effects e non deve essere modificato).

Text Layer

Lo schema di un livello di tipo Text espande quello generale. Per comprendere meglio quali sono le informazioni nel JSON e come vengono organizzate è utile osservare da cosa è costituito un text layer su After Effects.

La struttura di un text layer si suddivide in due macrosezioni: *Transform*, che definisce le proprietà globali di transform del livello, e *Text*, che contiene alcune delle informazioni riguardanti il testo del livello. Le informazioni che riguardano l'aspetto del testo, come il font utilizzato, la grandezza o il colore, sono racchiuse nella proprietà *Source Text* ma non sono esposte. Queste proprietà si trovano e si possono modificare in due sezioni separate: *Character* e *Paragraph*. Anche il contenuto del testo è incluso in Source Text e non esposto, ed è modificabile direttamente via interfaccia grafica su After Effects.

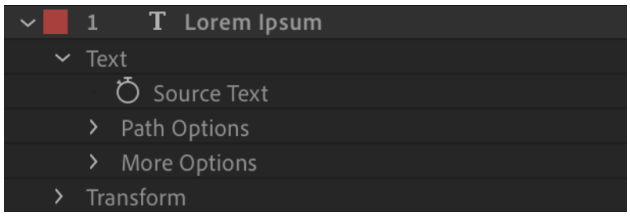


Figura 3.3: Text layer.

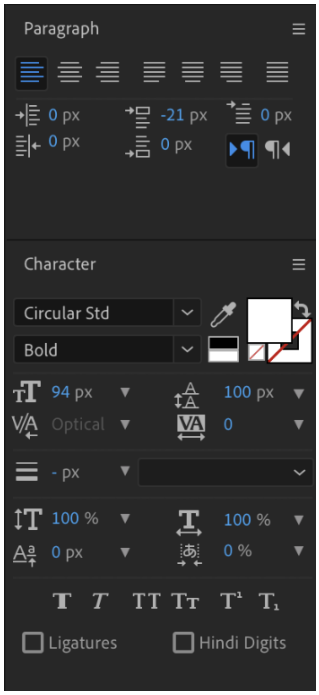


Figura 3.4: Sezioni Character e Paragraph.

Nel Lottie-JSON, le informazioni testuali del livello sono contenute in un unico campo *t*.

```
    "t": {
                "d": {
                    "k": [{
                            "s": {
                                "s": 72,
                                "f": "CircularAirPro-Light",
                                "t": "Lorem ipsum dolor sit
                                   amet"
                                "ca": 0,
                                "j": 0,
                                "tr": 0,
                                "lh": 100,
                                "ls": 0,
                                "fc": [0.05, 0.05,0.05]
                            },
                            "t": 0
                        }]
                },
                "p": {
                },
                "m": {
                    "g": 1,
                    "a": {
                        "a": 0,
                        "k": [0,0],
                        "ix": 2
                    }
                }
    }
```

I campi *p* e *m* corrispondono alle sezioni *Path* e *More Options* di un text layer. Allo scopo di spiegare la libreria realizzata, si analizza in modo più approfondito come viene codificata la proprietà Source Text, racchiusa nel campo *d*, all'interno dell'oggetto *t*. L'oggetto *d* a sua volta contiene l'array *k*. Gli elementi di questo array sono quelli che definiscono l'aspetto e il comportamento del testo. Sono oggetti composti di due campi: *s* e *t*. Il campo *s* racchiude tutte le proprietà supportate da Lottie e inglobate in Source Text. Il campo *t* indica un riferimento temporale associato all'oggetto. Per ogni elemento di *k*, il testo assume l'aspetto definito da *s*, al tempo *t*. Se Source Text non presenta keyframe, allora l'aspetto del testo rimane costante, e il Lottie è strutturato come nell'esempio fornito (la lunghezza di *k* è unitaria). Se invece viene animata, allora *k* presenta più elementi, tanti quanti sono i keyframe definiti su Source Text.

L'aspetto del testo è definito da *s*. I campi di questo oggetto sono:

- s: grandezza del font.

- f: font utilizzato. La stringa presente in questo campo viene comparata al campo *fName* della lista globale di font nell'oggetto JSON, in modo tale da recuperare le corrette informazioni del font per il render.
- t: la stringa di testo del layer.
- ca: indica se è stata selezionata l'opzione *All Caps* (testo in maiuscolo) su After Effects.
- j: indica la giustificazione del testo secondo il seguente schema:
  - 0: margine a sinistra
  - 1: margine a destra
  - 2: centrato
  - 3: giustificato a sinistra
  - 4: giustificato a destra
  - 5: giustificato al centro
  - 6: giustificato
- tr: indica il *tracking* (ovvero la distanza tra i caratteri) del testo, tuttavia questa feature non è supportata dal player di Lottie e quindi non ha alcun effetto grafico.
- lh: *line height*, ovvero la distanza tra le linee del testo.
- ls: *line shift*, ovvero l'offset positivo o negativo applicato sulla baseline del testo.
- fc: indica il colore del testo, memorizzato come un array tridimensionale $[r, g, b]$ in scala 0-1.

Il testo su After Effects può essere di tipo *paragraph*: viene definito un *bounding box* (rettangolo di delimitazione) e il testo si dispone in modo da rispettarne i limiti. In questo caso sono presenti due ulteriori attributi in *s*:

- sz: array bidimensionale che indica la dimensione del bounding box.
- ps: array bidimensionale che indica la posizione del bounding box.

Su un layer di testo è possibile aggiungere un *animator*, un elemento che permette di animare delle proprietà di un testo in base ai suoi caratteri, parole o linee. Un animator è composto da uno o più *selector*, che governano una o una serie di proprietà. I selector funzionano come delle "maschere" ed indicano su quali porzioni del testo e con quale intensità deve applicarsi la proprietà collegata. Possono agire in base ai caratteri, ai caratteri esclusi gli spazi, alle parole o alle linee del testo.

Esistono tre tipologie di selector:

- Expression Selector: la regola di selezione è definita in base ad una expression che deve essere esplicitamente inserita dal motion designer.
- Wiggly Selector: la regola di selezione è definita da un valore randomico e variabile nel tempo, che indica con quale intensità l'elemento di riferimento (carattere, parola,...) viene influenzato dall'animator. Il modo in cui il selector agisce è governabile da specifiche proprietà, per esempio è possibile definire la percentuale e la frequenza di variazione del valore randomico.
- Range Selector: la regola di selezione è definita da un range specifico, identificato da uno start, un end, ed eventualmente un offset. Gli elementi che ricadono all'interno del range subiscono gli effetti dell'animator. Le proprietà di start, end e offset possono essere indicate come percentuale o come indice dell'elemento all'interno del testo. Il range selector è customizzabile anche su ulteriori parametri più avanzati (per esempio è possibile definire in che modo avviene la transizione da un elemento - carattere, parola,... - all'altro).

Le proprietà animabili tramite un animator sono molteplici: proprietà di transform, proprietà di colore, e proprietà uniche del testo come tracking o character offset.

Per quanto riguarda Lottie, non tutti i selector e non tutte le proprietà sono supportati. Innanzitutto, è supportato unicamente il Range Selector, e non è possibile definire più selector per un unico animator. Inoltre, non sono rese disponibili tutte le proprietà, ma solamente quelle riferite alle transform e al colore (sono escluse dunque le proprietà di tracking, line anchor, line spacing, character offset, character value, e blur).

Le informazioni sugli animator sono incluse in un array presente sotto il campo a all'interno dell'oggetto t del testo (a compare solo se sono stati definiti animator sul livello in After Effects).

```
    "a": [
        {
            "nm": "Animator 1",
            "s": {...},
            "a": {...}
        }
    ]
```

L'array *a* contiene tanti oggetti quanti sono gli animator definiti sul text layer. Ogni oggetto rappresenta un animator ed è composto dai seguenti campi:

- nm: nome dell'animator.
- s: oggetto che rappresenta il range selector.
- a: oggetto che rappresenta le proprietà governate dall'animator.

Si analizzano *a* ed *s*.
L'oggetto a contiene una lista di oggetti a struttura a-k-ix rappresentanti le proprietà dell'animator.

L'oggetto s ha la seguente struttura:

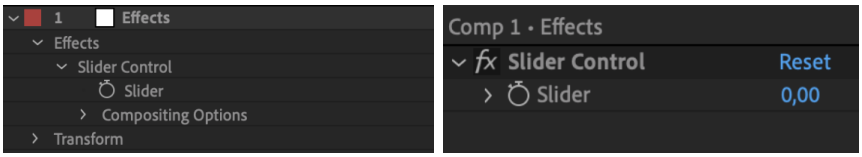
```
    "s": {
            "t": 0,
            "xe": {
                "a": 0,
                "k": 0,
                "ix": 7
            },
            "ne": {
                "a": 0,
                "k": 0,
                "ix": 8
            },
            "a": {
                "a": 0,
                "k": 100,
                "ix": 4
            },
            "b": 1,
            "rn": 0,
            "sh": 1,
            "sm": {
                "a": 0,
                "k": 100,
                "ix": 6
            },
            "s": {
                "a": 0,
                "k": 0,
                "ix": 4
            },
            "e": {
                "a": 0,
                "k": 8,
                "ix": 5
            },
            "o": {
                "a": 0,
                "k": 0,
                "ix": 6
            },
            "r": 2
       }
```

- t: tipologia del selector (0 rappresenta il range selector, l'unico supportato da Lottie).

- s: start del selector, rappresentato con un oggetto a-k-ix. Lo start indica l'inizio del range di azione dell'animator.
- e: end del selector, rappresentato con un oggetto a-k-ix. L'end indica la fine del range di azione dell'animator.
- o: offset del selector, rappresentato con un oggetto a-k-ix. L'offset sposta il range di azione dell'animator.
- xe e ne: *ease-high* e *ease-low* del selector, rappresentati con oggetti a-k-ix. Determinano la velocità della transizione della selezione.
- a: *amount*, rappresentato con un oggetto a-k-ix. Indica quanto gli elementi del testo vengono influenzati dalla proprietà dell'animator.
- b: parametro sul quale il selector si basa, secondo il seguente schema:
  - 1: Caratteri
  - 2: Caratteri esclusi gli spazi
  - 3: Parole
  - 4: Linee
- rn: indica se è selezionata l'opzione *randomize* (nel caso questo campo è uguale a 1). Questa opzione aggiunge un fattore random all'ordine nel quale l'animazione viene applicata agli elementi del testo.
- sh: shape del selector. Controlla in che modo - secondo quale forma - avviene la transizione tra gli elementi della selezione. Sono disponibili sei diverse modalità.
- sm: smoothness del selector, rappresentato con un oggetto a-k-ix. Determina la quantità di tempo che impiega la transizione dell'animazione da un elemento all'altro quando si utilizza la shape Square.
- r: indica "l'unità di misura" dei parametri del selector. Se è uguale a 1, allora i parametri sono da leggersi in percentuale, se è uguale a 2 invece sono da leggersi come l'indice dell'elemento di riferimento all'interno del testo.

3.1.3 Effetti

È possibile aggiungere o modificare delle caratteristiche dei livelli tramite l'applicazione di effetti. Benché After Effects ne metta a disposizione un numero estremamente vasto, Lottie ne supporta un insieme limitato: nella documentazione ufficiale sono indicati *Drop shadow* (Ombra), *Fill* (Riempimento colore), *Levels* (Livelli automatici), *Tint* (Tinta), *Tritone* (Tre tonalità), *Stroke* (Tratto) e *Gaussian Blur* (Controllo sfocatura). Nella sezione degli effetti rientrano anche gli *Expression Control*. Questi non modificano l'aspetto di un layer, ma vengono utilizzati per fornire un valore ed essere utilizzati all'interno delle expression, in modo da renderle scalabili. Sono supportati in Lottie.



(a) Effetti: Panel dei layer (b) Effetti: Panel degli effetti

Figura 3.5: Effetto Slider Control

Nel Lottie-JSON gli effetti sono memorizzati in un array sotto il campo *ef*, all'interno dell'oggetto del relativo layer. Si prende d'esempio un livello a cui sia stato applicato un'effetto di tipo Slider Control (Expression Control che espone un valore numerico decimale).

L'array degli effetti ha la seguente struttura:

```
"ef": [
        {
            "ty": 5,
            "nm": "Slider Control",
            "np": 3,
            "mn": "ADBE Slider Control",
            "ix": 7,
            "en": 1,
            "ef": [
                {
                    "ty": 0,
                    "nm": "Slider",
                    "mn": "ADBE Slider Control-0001",
                    "ix": 1,
                    "v": {
                        "a": 0,
                        "k": 0,
                        "ix": 1
                    }
                }
             ]
            }
        ]
```

Gli effetti sono rappresentati dagli oggetti in *ef*. A prescindere dall'effetto, i seguenti campi sono sempre presenti:

- ty: tipo dell'effetto.
- nm: nome dell'effetto.
- mn: *match name* dato da After Effects, usato per le expression.
- ix: indice usato per le expression.
- en: indica se l'effetto è *enabled* (attivo) o meno .
- ef: questo è l'array che contiene i valori editabili dell'effetto considerato. Contiene tanti elementi - oggetti - quanti sono i parametri dell'effetto. Per esempio, Slider Control ha un unico parametro, dunque *ef* ha lunghezza unitaria, mentre Gaussian Blur presenta tre parametri (*Blurriness*, *Blur Dimensions* e *Repeat Edge Pixels*), dunque *ef* ha lunghezza pari a 3. Un elemento di *ef* ha la seguente struttura:
  - ty: tipo del parametro
  - nm: nome del parametro

- mn: *match name* dato da After Effects, utilizzato per le expression
- ix: indice del parametro, utilizzato per le expression
- v: valore del parametro. Ha una struttura diversa in base all'effetto. Se è animabile ha struttura a-k-ix.

3.1.4 Animazione e gestione dei keyframe

Le proprietà analizzate finora sono sempre state considerate nella loro forma statica, non animata. Tuttavia, poiché si ha a che fare con video di motion graphics, è necessario comprendere in che modo il Lottie-JSON elabori e memorizzi le informazioni di animazione. In After Effects una proprietà può essere animata secondo due modalità: tramite expression e tramite keyframe. Nel primo caso, sul Lottie-JSON, l'expression viene indicata sotto la chiave x. Il valore del campo x è l'expression opportunamente convertita in Javascript. Benché infatti il linguaggio delle expression si basi su Javascript, in esse sono incluse una serie di metodi ed oggetti propri ad After Effects - come Layer, Comp, etc - che devono essere opportunamente gestiti e tradotti. È importante sottolineare che non tutte le expression sono supportate ed è necessario testarle opportunamente in Lottie.

Per comprendere in che modo vengono codificati i keyframe, invece, è necessario affrontare un discorso più approfondito. Nel Lottie-JSON, una proprietà sulla quale si possono definire dei keyframe ha sempre la struttura a-k-ix, dove a è il flag che indica se la proprietà è animata, e k indica il valore della proprietà. Una proprietà è animata se cambia il suo valore nel tempo, ovvero se su di essa viene definito più di un keyframe. In questo caso l'oggetto a-k-ix che descrive la proprietà varia leggermente: il flag a diventa uguale ad 1, e k non contiene più il singolo valore della proprietà, bensì le informazioni riguardanti i keyframe.

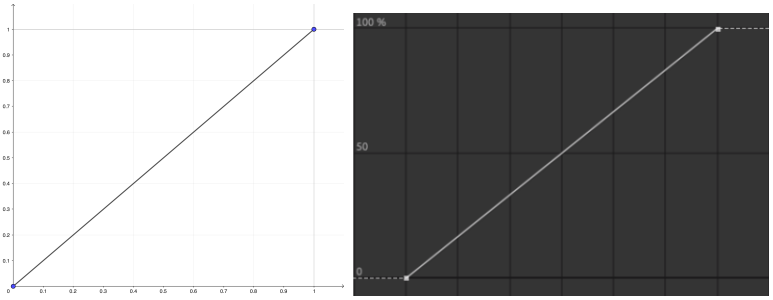
Un keyframe indica che la proprietà deve assumere uno specifico valore in uno specifico istante di tempo (o frame). Deve essere quindi definito almeno da due parametri: l'istante temporale in cui viene inserito nella timeline e il valore che la proprietà deve assumere in quell'istante. Il motion designer inserisce i keyframe su specifici istanti temporali, e la proprietà deve variare in modo tale da assumere i valori indicati dai keyframe negli istanti di tempo ad essi associati. After Effects ha il compito di calcolare i valori per ogni istante di tempo compreso tra due keyframe, tramite un processo detto *interpolazione temporale*.

È possibile definire diversi metodi di interpolazione e variarne le caratteristiche, cambiando quindi il modo in cui i valori tra i keyframe vengono calcolati. Per osservare in che modo agisce l'interpolazione sul valore cal-

colato, si può osservare la sua *curva di interpolazione*, la quale rappresenta la variazione del valore nel tempo. Tipi di interpolazione diversi presentano curve diverse. In After Effects la curva di interpolazione è visibile come *value graph*, ed indica proprio quali valori assume la proprietà nel tempo.

Per ogni keyframe è possibile definire il tipo di interpolazione temporale ed eventualmente i parametri che la definiscono. Le modalità di interpolazione sono:

- Interpolazione Lineare: la curva di interpolazione è una linea retta che congiunge i valori noti alle estremità. Il valore cambia dunque in maniera costante dal valore di partenza a quello di arrivo.



(a) Curva di interpolazione (b) Value graph

Figura 3.6: Curva di interpolazione lineare e value graph di una proprietà animata con due keyframe lineari.

- Interpolazione di Bézier: la curva di interpolazione è una curva di Bézier cubica, ovvero una curva parametrica definita da quattro punti nel piano o nello spazio - P_0, P_1, P_2, P_3 - secondo la seguente formula:

$$B(t) = P_0(1-t)^3 + 3P_1t(1-t)^2 + 3P_2t^2(1-t) + P_3t^3, t \in [0,1]$$

  La curva passa sempre per i punti d'estremità P_0 e P_3. La posizione dei punti intermedi P_1 e P_2 serve invece per definirne la forma e la direzione.

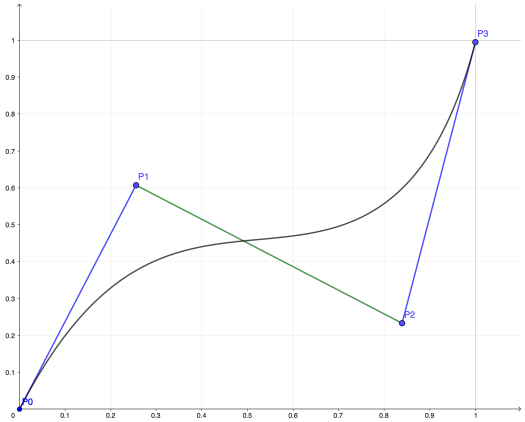
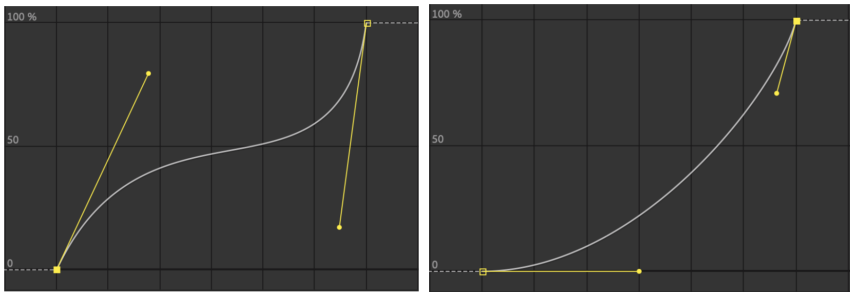


Figura 3.7: Esempio di una curva di Bézier.

Su After Effects, lo spazio bidimensionale su cui è definito il value graph è un piano valore-tempo. Si considerano due keyframe con interpolazione di Bézier e la curva parametrica che li collega: i punti d'estremità P_0 e P_3 corrispondono rispettivamente al keyframe di partenza e di arrivo. I punti P_1 e P_2 sono invece definibili nel grafico tramite delle apposite maniglie, per P_1 associata al keyframe iniziale, per P_2 a quello finale.



(a) Value graph (b) Value graph

Figura 3.8: Value graph di una proprietà animata con due keyframe di Bézier con diverse configurazioni.

È evidente quindi che i keyframe di Bézier offrano all'animatore maggiore controllo, poiché permettono di manipolare direttamente la curva di interpolazione. Ad un generico keyframe di Bézier sono associate due maniglie: una per la curva in entrata, una per la curva in uscita.

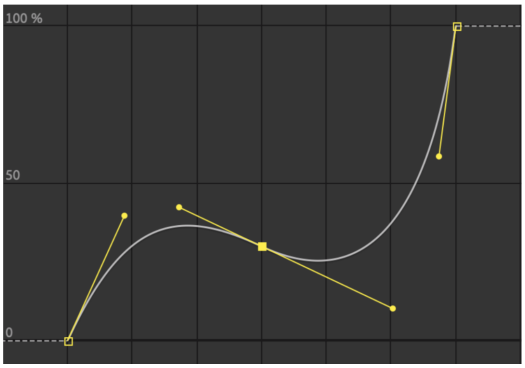


Figura 3.9: Value graph di una proprietà animata con tre keyframe di Bézier.

La posizione delle maniglie può essere definita direttamente sul value graph o impostata tramite due parametri numerici:

- Influence: corrisponde alla posizione orizzontale della maniglia. È indicata in termini percentuali rispetto alla distanza temporale tra i keyframe: per esempio, influence pari a 50% indica che il punto della maniglia si posiziona - sull'asse orizzontale - a metà tra i due keyframe.
- Velocity: determina l'inclinazione della maniglia, ed è definita come il rate di variazione della proprietà al secondo. La posizione verticale della maniglia, dunque, corrisponde alla retta identificata dalla velocity valutata nella coordinata orizzontale influence.

Si considerano dunque due keyframe di coordinate nel value graph (t_0, v_0) e (t_3, v_3) - dove t è il valore temporale espresso in frame, e v il valore della proprietà - con influence e velocity, rispettivamente (inf_0, vel_0) e (inf_3, vel_3). Le coordinate dei punti intermedi della curva di Bézier P_1 e P_2 sono date rispettivamente da (t_1, v_1) e (t_2, v_2) con:

$$(t_1, v_1) = \left(t_0 + \frac{inf_0(t_3 - t_0)}{100}, v_0 + \frac{inf_0(t_3 - t_0)}{100} \cdot \frac{vel_0}{framerate} \right)$$

$$(t_2, v_2) = \left(t_3 - \frac{inf_3(t_3 - t_0)}{100}, v_3 - \frac{inf_3(t_3 - t_0)}{100} \cdot \frac{vel_0}{framerate} \right)$$

Si nota che il parametro di velocity, definito come rate di variazione al secondo, deve essere normalizzato sul framerate della composition se il

tempo viene espresso in termini di frame. Per ogni keyframe di Bézier sono definite due coppie di influence-velocity, una per la maniglia in entrata e una per la maniglia in uscita. Ci si riferisce a questi parametri, rispettivamente, come informazioni di ease in e di ease out.

L'interpolazione lineare può essere considerata come un'interpolazione di Bézier le cui maniglie sono disposte in modo da formare una linea retta.

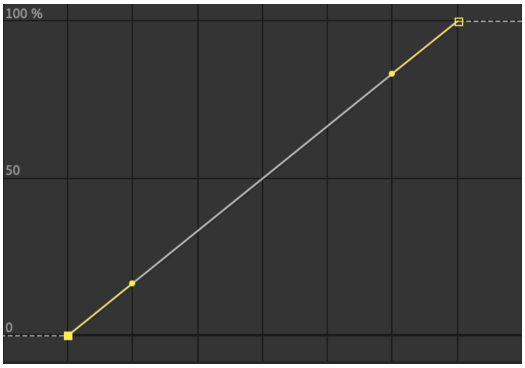


Figura 3.10: Value graph di una proprietà animata con keyframe lineari.

Su After Effects sono anche disponibili keyframe di tipo Auto Bézier e Continuous Bézier. I primi calcolano automaticamente le informazioni di ease in ed ease out in modo tale da assicurare che la transizione tra i keyframe avvenga in maniera uniforme. Influence e velocity cambiano e si adattano opportunamente al cambio di valore (temporale o della proprietà) del keyframe, in modo tale da garantire che le maniglie di entrata e di uscita siano sempre allineate sulla stessa retta. I secondi sono simili ai keyframe di tipo Auto Bézier - mantengono una transizione uniforme e le maniglie allineate - ma non si adattano al cambio di valore del keyframe (le maniglie devono essere modificate manualmente).

- Interpolazione Hold: viene utilizzata per creare transizioni "a scatto", non graduali. Se vengono definiti due keyframe di tipo hold con tempo e valore, rispettivamente, (t_0, v_0) e (t_1, v_1), la proprietà assumerà il valore v_0 per tutti i frame compresi nell'intervallo $[t_0, t_1)$ e v_1 per i frame compresi in $[t_1, +\infty)$.

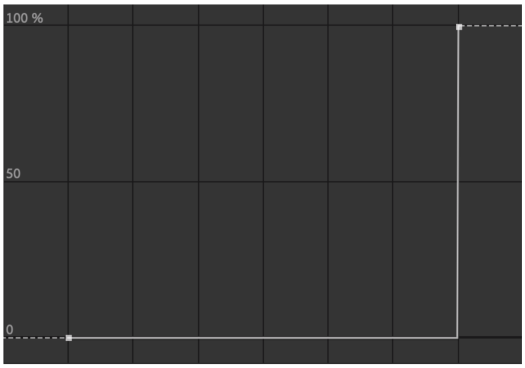
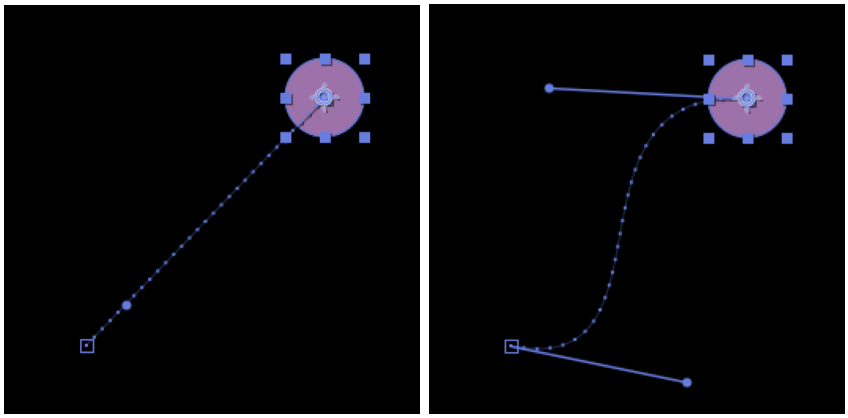


Figura 3.11: Value graph di una proprietà animata con keyframe hold.

Per le proprietà che variano nello spazio multidimensionale (come la posizione o l'anchor point), il processo di interpolazione dei valori avviene anche sulla dimensione spaziale. Si parla dunque di *interpolazione spaziale*. Se, per esempio, si considera un layer che varia da una posizione iniziale (x_0, y_0) ad una posizione finale (x_1, y_1), il percorso che il layer compie non è unico. Questo viene definito dal motion designer agendo sulla tipologia - e sulla configurazione - dell'interpolazione spaziale. Questa presenta le modalità Lineare, Bézier, Auto Bézier e Continuous Bézier. A differenza dell'interpolazione temporale, la curva di interpolazione è modificabile solo per interfaccia grafica, spostando le maniglie dei punti.



(a) Interpolazione lineare. (b) Interpolazione di Bézier.

Figura 3.12: Differenti percorsi di animazione definiti dall'interpolazione spaziale.

Un keyframe è dunque definito dalle seguenti informazioni: tempo, valore, modalità di interpolazione (ed eventualmente informazioni di ease in ed ease

out) temporale e/o spaziale.

È necessario analizzare come vengono elaborate e memorizzate queste informazioni nel Lottie-JSON. Innanzitutto, il modo in cui le informazioni del keyframe sono organizzate varia leggermente in base alla natura della proprietà sulla quale sono definiti. Si descrive di seguito il caso più generico e comune, e si osserveranno successivamente dei casi particolari. Si prende come esempio la proprietà di scala di uno shape layer sulla quale sono stati definiti tre keyframe lineari.

```
    "k": [
            {
                "i": {
                    "x": [0.833,0.833,0.833],
                    "y": [0.833,0.833,0.833]
                },
                "o": {
                    "x": [0.167,0.167,0.167],
                    "y": [0.167,0.167,0.167]
                },
                "t": 0,
                "s": [100,100,100]
            },
            {
                "i": {
                    "x": [0.833,0.833,0.833],
                    "y": [0.833,0.833,0.833]
                },
                "o": {
                    "x": [0.167,0.167,0.167],
                    "y": [0.167,0.167,0.167]
                },
                "t": 15,
                "s": [50,50,100]
            },
            {
                "t": 30,
                "s": [120,120,100]
            }
            ],
            "ix": 6,
            "l": 2
        }
```

Nell'array *k* della struttura a-k-ix della proprietà sono memorizzate le informazioni riguardanti i keyframe. La lunghezza di questo array corrisponde al numero di keyframe inseriti sulla proprietà. Tuttavia, si può notare che non vi è una corrispondenza diretta tra *n*-esimo oggetto e *n*-esimo keyfra-

me. Infatti, non tutti gli oggetti hanno la stessa struttura: l'ultimo presenta sempre e solo due campi, t ed s.

Preso un n-esimo oggetto dell'array k, questo è composto dalle seguenti proprietà:

- i: è un oggetto che contiene le informazioni di ease in dell'$(n+1)$-esimo keyframe.
  - x: è un array che contiene le informazioni riguardanti l'influence di ease in. La sua lunghezza è pari alla dimensione della proprietà animata.
  - y: è un array che contiene le informazioni riguardanti la velocity di ease in. La sua lunghezza è pari alla dimensione della proprietà animata.
- o: è un oggetto che contiene le informazioni di ease out dell'n-esimo keyframe.
  - x: è un array che contiene le informazioni riguardanti l'influence di ease out. La sua lunghezza è pari alla dimensione della proprietà animata.
  - y: è un array che contiene le informazioni riguardanti la velocity di ease out. La sua lunghezza è pari alla dimensione della proprietà animata.
- t: riferimento temporale del keyframe, espresso in frame.
- s: è un array la cui lunghezza è uguale alla dimensione della proprietà che contiene il valore del keyframe, ovvero il valore che la proprietà assume al frame t. Se la proprietà è monodimensionale, rimane un array di lunghezza unitaria.

Non vi è dunque una corrispondenza univoca tra keyframe e elementi di k dello stesso indice: l'n-esimo elemento di k contiene alcune informazioni riguardanti l'n-esimo keyframe - ease out, tempo e valore - e alcune informazioni riguardanti l'$(n+1)$-esimo keyframe - ease in. Questa spiega la struttura differente dell'ultimo elemento dell'array. Ai fini del render corretto dell'animazione, non è necessario immagazzinare le informazioni di ease out dell'ultimo keyframe definito su After Effects, poiché non vi è alcuna curva di interpolazione sulla quale possano agire. Per questa ragione, il campo o è assente dall'ultimo elemento di k. Le informazioni di ease in invece sono contenute - come da regola - nell'elemento precedente, il penultimo oggetto.

Rimangono dunque solo le informazioni di tempo e valore. Analogamente accade per il primo keyframe, per il quale non è necessario memorizzare le informazioni di ease in. Il campo i presente, infatti, non si riferisce alle informazioni di ease in del primo keyframe, ma del secondo. Si evince dunque che per ogni keyframe si può risalire alle informazioni di ease in ed ease out, fatta eccezione per il primo (ease in assente) ed ultimo (ease out assente). Le informazioni di ease in ed ease out vengono opportunamente elaborate. I campi x e y, infatti, non corrispondono direttamente ai valori di influence e velocity inseriti su After Effects.

Si considerano, per esempio, una composition con framerate pari a 30 fps e due keyframe di Bézier, K_1 e K_2, applicati sulla proprietà di opacità di uno shape layer, definiti dai seguenti parametri:

	K_1	K_2
Valore(%)	10	100
Tempo (frame)	5	30
Ease In Influence (%)	-	20
Ease In Velocity (%/sec)	-	300
Ease Out Influence (%)	40	-
Ease Out Influence (%/sec)	120	-

Il value graph in After Effects ha il seguente aspetto:

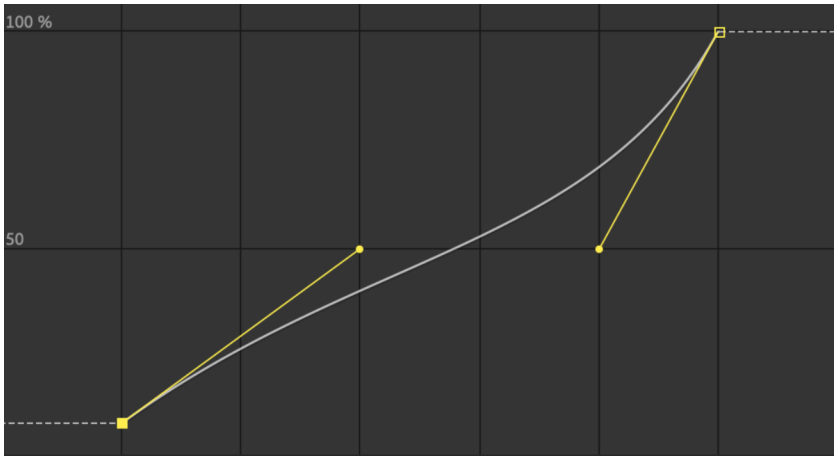


Figura 3.13: Value graph della proprietà di opacità animata con K_1 e K_2.

Nel Lottie-JSON i keyframe vengono memorizzati nella seguente struttura:

```
    "k": [
            {
                "i": {
                    "x": [0.8],
                    "y": [0.444]
                },
                "o": {
                    "x": [0.4],
                    "y": [0.444]
                },
                "t": 5,
                "s": [10]
            },
            {
                "t": 30,
                "s": [100]
            }
       ]
```

Si prende in considerazione l'ease out del primo keyframe, definito su After Effects con influence pari a 40% e velocity pari a 120 %/sec. Su Lottie le informazioni di ease out sono contenute nell'oggetto *o* del primo elemento: il campo x contiene le informazioni di influence, il campo y di velocity. Nello specifico, i valori x ed y rappresentano la posizione della maniglia di ease out nel value graph normalizzato sull'asse orizzontale e verticale, e traslato in modo tale da avere come origine le coordinate del keyframe sul quale è definito l'ease out. Ci si riferisce a questo grafico come *grafico xy*.

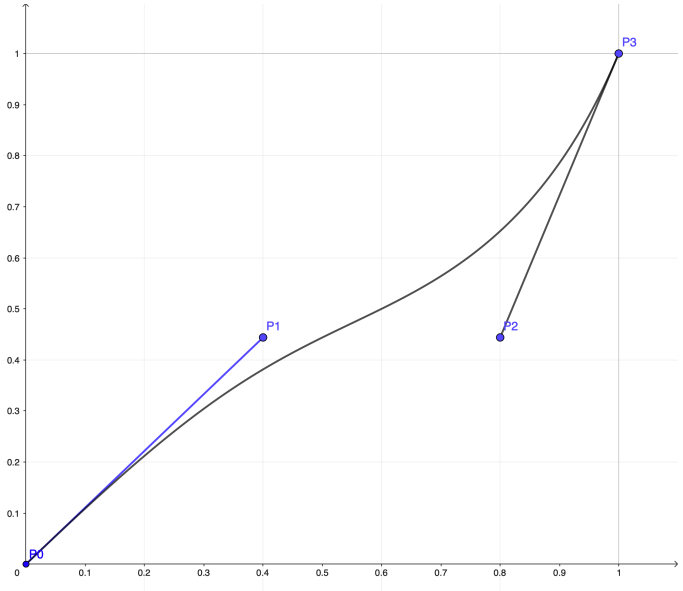


Figura 3.14: Curva di interpolazione dell'animazione definita da K_1 e K_2 sul grafico xy.

Per passare dal value graph di After Effects al grafico xy è necessario applicare due operazioni:

- Traslazione, così da portare il punto identificato dal primo keyframe all'origine del sistema di riferimento.
- Normalizzazione, così da ottenere valori temporali e spaziali compresi tra 0 e 1.

Per comprendere come si ottengono i valori di x e di y, è necessario dunque comprendere il rapporto tra influence e velocity, e le coordinate della maniglia corrispondente sul value graph. Come è stato precedentemente spiegato, presi due keyframe di coordinate - nel value graph - (t_0, v_0) e (t_1, v_1), le coordinate della maniglia di ease-out sono date da:

$$(t_{out}, v_{out}) = \left(t_0 + \frac{inf_o(t_1 - t_0)}{100} \ , \ v_0 + \frac{inf_o}{100} \cdot \frac{vel_o(t_1 - t_0)}{framerate} \right)$$

Per ottenere x è necessario traslare e normalizzare la coordinata orizzontale, ovvero:

- sottrarre la coordinata orizzontale del primo keyframe t_0
- dividere per la differenza temporale tra i keyframe $(t_1 - t_0)$

Si ottiene dunque:

$$x = \frac{inf_o}{100}$$

Per ottenere y è necessario traslare e normalizzare la coordinata verticale, ovvero:

- sottrarre la coordinata verticale del primo keyframe v_0
- dividere per la differenza tra i valori dei keyframe $(v_1 - v_0)$

Si ottiene dunque:

$$y = \frac{vel_o(t_1 - t_0)}{framerate(v_1 - v_0)} \cdot \frac{inf_o}{100}$$

Se la proprietà è multidimensionale, i valori di x e y vengono calcolati su ognuna delle dimensioni.

Considerando una proprietà di dimensione d, le informazioni di ease out del j-esimo keyframe sono contenute nei valori di x ed y dell'oggetto o del j-esimo elemento dell'array k, $k[j].o$, definiti nel seguente modo:

$$x_o[n] = \frac{inf_o[n]}{100}$$

$$y_o[n] = \frac{vel_o[n](k[j+1].t - k[j].t)}{framerate(k[j+1].s[n] - k[j].s[n])} \cdot \frac{inf_o[n]}{100}$$

per ogni $n \in [0, d]$, e dove $inf_o[n]$ e $vel_o[n]$ sono i valori di outgoing influence e velocity della n-esima dimensione impostati su After Effects.

Lo stesso sistema di assegnazione avviene per i campi x ed y di ease in. Presi i due keyframe di posizione - nel value graph - (t_0, v_0) e (t_1, v_1), la posizione della maniglia di ease-in è data da:

$$(t_{in}, v_{in}) = \left(t_1 - \frac{inf_i(t_1 - t_0)}{100} \ , \ v_1 - \frac{inf_i}{100} \cdot \frac{vel_i(t_1 - t_0)}{framerate} \right)$$

Per ottenere x è necessario traslare e normalizzare la coordinata orizzontale, ovvero:

- sottrarre la posizione del primo keyframe t_0
- dividere per la differenza temporale tra i keyframe $(t_1 - t_0)$

Si può riscrivere t_{in} nel seguente modo:

$$t_{in} = t_1 - \frac{inf_i(t_1 - t_0)}{100} = t_1 - \frac{inf_i(t_1 - t_0)}{100} + t_0 - t_0 = t_0 + \left(1 - \frac{inf_i}{100}\right) \cdot (t_1 - t_0)$$

Si ottiene dunque:

$$x = 1 - \frac{inf_i}{100}$$

Per ottenere y è necessario traslare e normalizzare la coordinata verticale, ovvero:

- sottrarre la posizione del primo keyframe v_0
- dividere per la differenza tra i valori dei keyframe $(v_1 - v_0)$

Si ottiene:

$$y = \frac{\left(v_1 - \frac{inf_i}{100} \cdot \frac{vel_i(t_1 - t_0)}{framerate} - v_0\right)}{(v_1 - v_0} = \frac{(v_1 - v_0)}{(v_1 - v_0)} - \frac{vel_i(t_1 - t_0)}{framerate(v_1 - v_0)} \cdot \frac{inf_i}{100}$$

e quindi:

$$y = 1 - \frac{vel_i(t_1 - t_0)}{framerate(v_1 - v_0)} \cdot \frac{inf_i}{100}$$

Considerando una proprietà di dimensione d, le informazioni di ease in del j-esimo keyframe sono contenute nei valori di x ed y dell'oggetto i del $(j-1)$-esimo elemento dell'array k, $(k[j].i)$, definiti nel seguente modo:

$$x_i[n] = 1 - \frac{inf_i[n]}{100}$$

$$y_i[n] = 1 - \frac{vel_i[n](k[j].t - k[j-1].t)}{framerate(k[j].s[n] - k[j-1].s[n])} \cdot \frac{inf_i[n]}{100}$$

per ogni $n \in [0, d]$, e dove $inf_i[n]$ e $vel_i[n]$ sono i valori di incoming influence e velocity della n-esima dimensione impostati su After Effects.

Da questa analisi si nota una particolarità che è risultata di fondamentale importanza nello sviluppo della libreria di customizzazione: il valore y, che contiene le informazioni riguardo la velocità del keyframe, dipende dalla influence (e quindi da x), dalla distanza temporale dei keyframe, e dalla differenza tra i valori dei keyframe, che verrà definita d'ora in poi *distanza*

spaziale. Se deve essere customizzato uno di questi valori è necessario cambiare opportunamente il valore di y, al fine di mantenere la velocity risultante uguale a quella definita su After Effects.

Lottie non distingue esplicitamente l'interpolazione di Bézier da quella lineare, così come non etichetta in maniera particolare keyframe di tipo Auto Bézier o Continuous Bézier. I keyframe di tipo lineare vengono memorizzati con valori costanti di x e y. Se l'ease in è di tipo lineare, x e y dell'oggetto i sono sempre uguali a 0.833. Se l'ease out è di tipo lineare, x e y dell'oggetto o sono sempre uguali a 0.167. Questi valori, in realtà, non si discostano dalla regola generale precedentemente analizzata. L'interpolazione lineare può considerarsi come un'interpolazione di Bézier la cui curva è la linea retta che congiunge il valore di partenza ed il valore di arrivo: sul grafico normalizzato xy questa retta è la bisettrice.

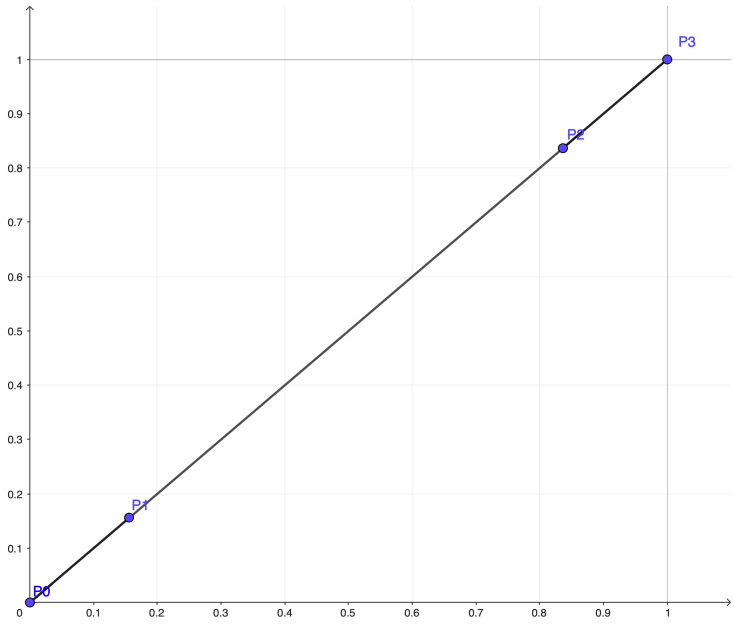


Figura 3.15: Curva di interpolazione lineare sul grafico xy.

Si considera la maniglia di ease out: se come valore di x si ha 0.167, il valore di y che rende l'ease out lineare è quello che identifica la retta con pendenza unitaria, ovvero 0.167. Analogamente accade per la maniglia di ease in: x ed y, entrambi pari a 0.833, mantengono un rapporto unitario. La scelta del valore di x - e quindi dell'influence - è potenzialmente arbitraria: la linearità si ottiene preservando il rapporto unitario tra x ed y. Si suppone che la scelta sia ricaduta sui valori 0.167-0.833 perchè su After Effects, un

keyframe di tipo Bézier con configurazione lineare presenta un valore di influence pari a 16.66667%. Se sia l'influence di ease out sia l'influence di ease in sono pari a questo valore, allora la x di ease out risulta essere effettivamente uguale a 0.167, mentre la x di ease in a 0.833.

L'unica interpolazione che viene esplicitamente etichettata come tale è quella di tipo hold. Un keyframe di tipo hold mantiene la proprietà al valore indicato fino al keyframe successivo, senza che vi sia transizione graduale di valori. Per questa ragione, nell'elemento corrispettivo dell'array k sono assenti sia le informazioni di ease in sia le informazioni di ease out. È presente però un campo aggiuntivo h pari ad 1, un flag per indicare che si tratta di un'interpolazione di tipo hold.

```
"p": {
        "a": 1,
        "k": [
            {
                "t": 0,
                "s": [739,477,0],
                "h": 1
            },
            {
                "t": 75,
                "s": [349, 233,0],
                "h": 1
            },
            {
                "t": 151,
                "s": [231,515,0],
                "h": 1
            }
        ],
        "ix": 2,
        "l": 2
    }
```

Alcune proprietà presentano una struttura dell'array k relativo ai keyframe diversa dallo schema generale spiegato finora.

Posizione e Anchor Point

Generalmente, se una proprietà è multidimensionale, è possibile inserire per ogni dimensione diversi valori di influence e velocity: x ed y risultano essere array di lunghezza uguale alla dimensione. Tuttavia, vi sono alcune proprietà in cui i valori delle diverse dimensioni vengono governati dalla stessa influence

e velocity, come avviene nel caso, per esempio, della posizione - quando non viene separata in X Position e Y Position - e dell'anchor point.

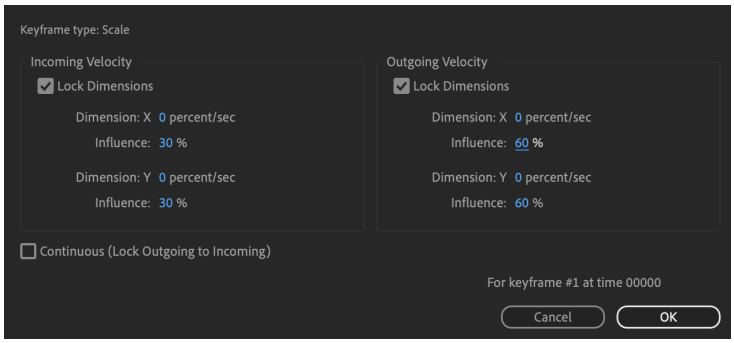


Figura 3.16: Keyframe settings della proprietà di scala.

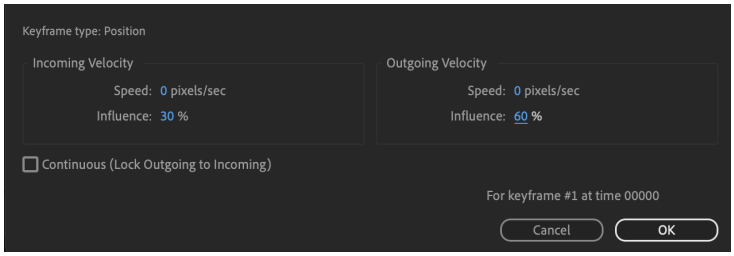


Figura 3.17: Keyframe settings della proprietà di posizione.

In questo caso, le informazioni vengono strutturate nel Lottie-JSON in maniera leggermente diversa rispetto alla regola standard.

```
"k": [
        {
            "i": {
                "x": 0.1,
                "y": 1
            },
            "o": {
                "x": 0.3,
                "y": 0
            },
            "t": 0,
            "s": [230,860,0],
            "to": [ 34.333,-271.667,0],
            "ti": [-276.333,-14.333,0]
        },
        {
            "t": 30,
            "s": [880,190,0]
        }
    ]
```

La prima differenza che si riscontra è che i valori legati all'influence e alla velocity - ovvero i campi x ed y - poichè sono unici per ogni dimensione, vengono memorizzati come valori singoli. Nella struttura analizzata precedentemente, invece, risultavano sempre essere degli array, anche in caso di proprietà monodimensionale (in quel caso erano array di lunghezza unitaria). Il calcolo del campo x e del campo y non subisce variazioni dal punto di vista logico. Il campo x dipende unicamente dall'influence inserita su After Effects, e si ottiene sempre tramite la formula analizzata precedentemente.

$$x_{out} = \frac{inf_o}{100} \ , \ x_{in} = 1 - \frac{inf_1}{100}$$

Il campo y dipende invece da più fattori: velocity, influence, distanza temporale e distanza spaziale. In questo nuovo contesto è necessario tuttavia ridefinire il concetto di distanza spaziale. Nella struttura generale, il campo y era un array: ogni n-esimo elemento dell'array y considerava come distanza spaziale la differenza tra l'n-esimo elemento di $k[j].s$ e l'n-esimo elemento di $k[j+1].s$ (o $k[j].s$ e $k[j-1].s$ nel caso di ease in). In questo caso, invece, s continua ad essere un array, mentre y è un valore unico. La distanza spaziale diventa la distanza nello spazio bidimensionale (o tridimensionale) tra i valori dei keyframe, ovvero la lunghezza della curva di spostamento definita dall'animazione. Per comprendere quale sia questa lunghezza, è necessario analizzare la seconda differenza riscontrata rispetto alla struttura generica dell'array k. Proprietà come la posizione e l'anchor point vengono animate

su uno spazio bidimensionale (o tridimensionale) e prevedono dunque anche interpolazione spaziale.

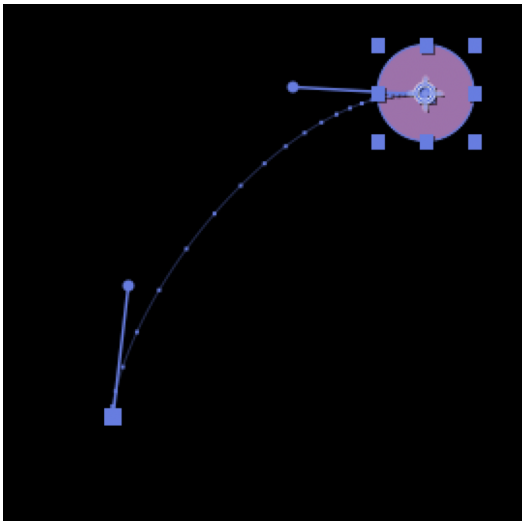


Figura 3.18: Animazione della posizione con interpolazione spaziale di Bézier.

Gli elementi dell'array k, dunque presentano due ulteriori campi che contengono le informazioni di interpolazione spaziale: *to* e *ti*, rispettivamente *out tangents* e *in tangents*. Questi sono array di lunghezza pari alla dimensione della proprietà e permettono di ottenere la posizione nello spazio delle maniglie di ease out e ease in. In particolar modo *to* e *ti* contengono l'offset da sommare, rispettivamente, alla posizione di partenza e alla posizione di arrivo definite dai keyframe. Analogamente a quanto succede per l'interpolazione temporale, le informazioni di ease out dell'n-esimo keyframe sono contenute nell'n-esimo elemento di k, mentre le informazioni di ease in dell'n-esimo keyframe sono contenute nell'$(n-1)$-esimo elemento di k.

I valori dei keyframe insieme alla posizione delle maniglie identificano la curva che l'oggetto compie per spostarsi dalla posizione di partenza alla posizione di arrivo. La distanza spaziale è proprio la lunghezza di questa curva.

Il campo y si calcola dunque secondo la seguente formula:

$$y_{out} = \frac{vel_o(k[j+1].t - k[j].t)}{framerate \cdot distance_o} \cdot \frac{inf_o}{100} \ , \ y_{in} = 1 - \frac{vel_i(k[j].t - k[j-1].t)}{framerate \cdot distance_i} \cdot \frac{inf_i}{100}$$

dove $distance_o$ è la lunghezza della curva identificata dai quattro punti di posizione $k[j].s,\ k[j].s + k[j].to,\ k[j].ti + k[j+1].s,\ k[j+1].s$.

e $distance_i$ è la lunghezza della curva identificata dai quattro punti di posizione $k[j-1].s,\ k[j-1].s + k[j-1].to,\ k[j-1].ti + k[j].s,\ k[j].s$.

Queste distanze vengono calcolate per approssimazione, valutando la curva di Bézier in diversi punti e sommando i segmenti tra i punti individuati.

Così come avviene per l'interpolazione temporale, non vi è una distinzione netta tra interpolazione lineare o di Bézier. L'interpolazione spaziale lineare può essere indicata in due modi:

- *to* e *ti* identificano due maniglie tali per cui la curva di interpolazione risultante è una retta. In questo caso non vi sono valori fissi di riferimento come nell'interpolazione temporale, poichè, essendo in un sistema di riferimento non normalizzato, i valori delle posizioni delle maniglie dipendono sempre dai valori dei keyframe.
- *to* e *ti* sono array di zeri

Colore

Il colore è una proprietà multidimensionale, poiché composta dalle informazioni del canale R, G, B e alpha. La struttura dell'array *k* riprende quella standard.

```
 "k": [
{
    "i": {
        "x": [0.4],
        "y": [2.405]
    },
    "o": {
        "x": [0.3],
        "y": [0]
    },
    "t": 0,
    "s": [0.61505503935,0.447642008464,0.667292995079,1]
},
{
    "t": 30,
    "s": [0.447641998529,0.667293012142,0.544288694859,1]
}
 ]
```

Tuttavia, anche in questo caso è necessario analizzare il concetto di distanza spaziale. Su After Effects si inserisce un unico valore di influence e di velocity per definire l'animazione di un colore: i campi x ed y sono array monodimensionali. I valori di colore vengono però definiti nello spazio RGBa, e sono dunque rappresentati con array di lunghezza pari a quattro. Si potrebbe pensare che la distanza spaziale sia - come nel caso della posizione e

dell'anchor point - la lunghezza della curva definita nello spazio tridimensionale dai diversi valori di R,G, e B (alpha rimane sempre uguale a 1), ma non è questo il caso. Il modo in cui vengono calcolati x ed y riprende esattamente quello della regola standard. In questo caso viene considerata come distanza spaziale la differenza tra i valori del canale R dei due colori.

$$y_o[0] = \frac{vel_o(k[j+1].t - k[j].t)}{framerate(k[j+1].s[0] - k[j].s[0])} \cdot \frac{inf_o}{100}$$

$$y_i[0] = 1 - \frac{vel_i(k[j].t - k[j-1].t)}{framerate(k[j].s[0] - k[j-1].s[0])} \cdot \frac{inf_i}{100}$$

Ne consegue, dunque, che non è possibile animare una transizione tra due colori che hanno lo stesso canale di R con una velocity diversa da zero, poiché si otterrebbe y pari a *NaN*.

Source Text

Source Text è una proprietà animabile su After Effects, ma presenta un'elaborazione dei keyframe differente rispetto alla regola generale. Innanzitutto, l'oggetto che contiene le informazioni della Source Text non ha una struttura a-k-ix: è presente solamente l'array k. Questo perché la proprietà viene sempre considerata come "animata", anche quando è statica. L'array k contiene i keyframe della proprietà: se questa non è stata animata immagazzina le informazioni statiche della Source Text in un unico keyframe, altrimenti contiene tanti elementi quanti sono i keyframe aggiunti. Inoltre, per sua natura, Source Text può accettare *solamente* keyframe di tipo hold. Per questa ragione, il flag di hold h è assente - poiché non sono previste alternative a questa tipologia di interpolazione. Il campo s, poi, non è un array di valori, bensì un oggetto, che contiene tutte le proprietà racchiuse dalla Source Text.

3.2 Sviluppo di una libreria JavaScript di funzioni per la customizzazione di animazioni in formato Lottie

La struttura di un Lottie-JSON può essere lunga, complessa, e articolata. L'idea di sviluppare una libreria di funzioni JavaScript per la manipolazione del JSON nasce per poter automatizzare l'accesso e la modifica delle informazioni in esso contenute, così da rendere il processo di customizzazione più rapido e snello. Prima dello sviluppo è stato necessario affrontare due

questioni principali: in che modo strutturare la libreria e quali funzioni e proprietà supportare.

I progetti di Algo sono progetti di video automation realizzati con due processi diversi, uno JSX-based - la customizzazione avviene tramite scripting su After Effects - e uno Lottie-based - la customizzazione avviene, finora, modificando manualmente il JSON dell'animazione. Il processo JSX-based è ampiamente consolidato: il motion engineer - ovvero la figura che nell'azienda si occupa di gestire l'automation e customization del video - ha dimestichezza e familiarità con la gestione delle funzioni in ExtendScript per l'accesso e la modifica delle varie proprietà di una composition. Si è pensato dunque di avvicinare, almeno nella fase di personalizzazione dell'animazione, questi due processi.

Un progetto After Effects viene rappresentato in ExtendScript con una struttura gerarchica ad oggetti. L'oggetto globale è il progetto. Al suo interno sono presenti gli oggetti rappresentanti le composition, all'interno di questi gli oggetti rappresentanti i layer, e all'interno di questi gli oggetti rappresentanti le proprietà di un livello: transform, maschere, effetti e proprietà proprie alla tipologia del layer. Ognuno di questi oggetti contiene i metodi appropriati per l'accesso e la modifica dei parametri che li definiscono. L'idea alla base della configurazione della libreria è stata dunque di replicare la struttura gerarchica con la quale viene rappresentato un progetto After Effects in ExtendScript, e con essa i metodi di customizzazione delle proprietà. La libreria dunque presenta una struttura a classi. Benché JavaScript non sia un linguaggio propriamente ad oggetti, mette a disposizione un costrutto *class* che serve per creare oggetti e definire su di essi metodi ed attributi. Si mantiene la stessa struttura gerarchica presente in JSX - che altri non è che la struttura gerarchica di un progetto in After Effects - con una differenza principale: il Lottie-JSON non rappresenta un progetto After Effects ma una composition. L'oggetto globale è dunque la composition del Lottie, al suo interno vi sono oggetti-layer, e all'interno di questi vi sono a sua volta oggetti-proprietà, sui quali sono stati definiti metodi custom di personalizzazione.

La libreria è stata dunque sviluppata in modo che la chiamata delle funzioni di customizzazione riprenda la stessa struttura della chiamata delle funzioni nel metodo JSX-based. Tuttavia, si è scelto di non replicare interamente questi metodi ed oggetti, ma di apportare, dove si ritenesse necessario, delle modifiche. La replica della struttura ad oggetti è stata pensata primariamente per mantenere una logica che fosse familiare al motion engineer. Si è analizzato se fosse il caso di riprodurre integralmente le funzioni JSX, in modo tale da facilitare il porting di un progetto JSX-based ad uno Lottie-based (e viceversa) utilizzando eventualmente uno script di persona-

lizzazione simile su entrambi i processi. Tuttavia, si è scelto di non seguire questa strada, poiché difficilmente è richiesto che uno stesso progetto debba essere realizzato con entrambe le metodologie. I due processi danno vita a campagne video con caratteristiche differenti: la natura stessa del progetto porta a preferire un metodo piuttosto che un altro. Inoltre, vi sono alcune operazioni che, tramite JSX, risultano piuttosto laboriose. Poiché è la libreria che definisce in che modo compiere tali operazioni su Lottie, si è pensato di snellire certi processi e renderli più semplici. Per esempio, un'operazione comune nella customizzazione di video, che tuttavia richiede diversi step tramite scripting JSX, è l'aggiunta di un keyframe su una proprietà. Una volta recuperata la proprietà di riferimento, si devono eseguire separatamente le diverse operazioni:

- creare due oggetti Ease che incapsulino le informazioni di ease in ed ease out del keyframe.
- aggiungere alla proprietà un keyframe, indicandone solo il placement temporale.
- accedere al keyframe aggiunto e definirne il valore.
- accedere al keyframe aggiunto e definirne il tipo di interpolazione.
- accedere al keyframe aggiunto e definirne l'ease in e l'ease out tramite gli oggetti Ease creati precedentemente.

```
easeIn = new KeyframeEase(speed, influence)
easeOut = new KeyframeEase(speed, influence)
keyIndex = app.project.item(index).layer(index).
   propertySpec.addKey(time)
app.project.item(index).layer(index).propertySpec.
   setValueAtKey(keyIndex)
app.project.item(index).layer(index).propertySpec.
   setInterpolationTypeAtKey(keyIndex, easeIn, easeOut)
```

Sono richieste cinque differenti operazioni per aggiungere un unico keyframe ad una proprietà. Per il motion engineer sarebbe più comodo e pratico, per esempio, definire un unico oggetto keyframe tramite il suo valore, il suo placement temporale, e le sue informazioni di ease in ed ease out, ad aggiungere questo oggetto-keyframe alla proprietà.

```
var easeIn = new Ease (influence, velocity)
var easeOut = new Ease (influence, velocity)
var key = new Keyframe (time, value, easeIn, easeOut)
composition.layer(index).propertySpec.addKey(key)
```

Come anticipato ed analizzato precedentemente, le proprietà customizzabili di un generico JSON sono molteplici: la scelta su quali supportare e su quali operazioni inserire nella libreria è stata fatta sulla base dei tipici progetti Algo e dei progetti realizzati nell'ambito del tirocinio. Ci si è concentrati in particolar modo - poiché si tratta di motion graphics e video animati - sul supporto quanto più approfondito possibile ai keyframe, e sulle proprietà base o più comuni delle diverse tipologie di layer supportati da Lottie.

Nell'ambito del tirocinio è stata realizzata una documentazione dettagliata pensata come *user guide*, che spiega come applicare le funzioni di customizzazione e quali metodi sono supportati dalla libreria, in modo tale che la personalizzazione del video possa essere sviluppata anche da persone che non abbiano familiarità né con Lottie né con la struttura della libreria. Si riporta tale documentazione nella sezione Appendice.

Si descrivono in seguito le classi principali della libreria.

3.2.1 Classi ausiliari per la gestione di keyframe: Ease, Keyframe e KeyframeProperty

Una delle feature primarie offerta dalla libreria è la personalizzazione di proprietà animabili mediante keyframe, di fondamentale importanza poiché questa è la modalità principale con la quale vengono definite le animazioni nei video di motion graphics. È stato necessario pensare dunque a delle classi che modellassero da una parte un oggetto keyframe e dall'altra le proprietà animabili mediante keyframe.

L'oggetto keyframe viene modellato con l'aiuto di due classi: Ease e Keyframe. Un oggetto Ease viene costruito attraverso i due parametri che definiscono un ease in o un ease out su After Effects: influence e velocity.

```
const ease = new Ease (influence, velocity)
```

Si è scelto di rappresentare l'ease lineare impostando influence pari a 0.

Un oggetto Keyframe viene costruito, almeno, tramite il suo placement temporale, ed il valore ad esso assegnato:

```
const key = new Keyframe (time, value)
```

Il tempo *time* è espresso in secondi ed il valore *value* dipende dalla proprietà sulla quale sarà definito questo keyframe: se è una proprietà di colore value sarà una stringa HEX, se è una posizione un array multidimensionale, se è una proprietà source text sarà una stringa di testo, etc. Il costruttore di un Keyframe accetta in realtà diversi parametri.

```
const key = new Keyframe (time, value, easeIn, easeOut,
   inTangents, outTangents, h)
```

Si possono inserire le informazioni di ease in ed ease out mediante opportuni oggetti Ease. Se easeIn ed easeOut non sono specificati, il keyframe risultante è lineare. Se invece viene inserito solo easeIn, easeOut viene impostato uguale ad easeIn, come avviene in ExtendScript. Il costruttore accetta eventuali informazioni di in-tangents ed out-tangents, corrispondenti ai campi *ti* e *to* del Lottie-JSON, ovvero array di offset per che permettono di identificare le rispettive maniglie di interpolazione nello spazio. Se inTangents e outTangents non sono definiti, o sono definiti come array di zeri, l'interpolazione spaziale risultante sarà lineare. Se viene definito solo inTangents, outTangents viene impostato uguale ad inTangents. Questi valori verranno presi in considerazione solo da quelle proprietà che prevedono interpolazione spaziale. I valori di influence e velocity riprendono la logica di assegnazione di After Effects, poiché è estremamente familiare al motion engineer, mentre i valori di in-tangents e out-tangents riprendono una logica di assegnazione più simile all'elaborazione di Lottie, poiché la posizione delle maniglie di interpolazione spaziale solitamente viene definita direttamente per interfaccia grafica e non tramite assegnazione esplicita delle coordinate. Si è scelta una linea di definizione del keyframe più vicina a Lottie che allo scripting in JSX, meno restrittiva e più pratica: non deve essere esplicitamente indicato il tipo di interpolazione - temporale o spaziale - ma questo viene dedotto dai valori assegnati ad easeIn, easeOut, inTangents e outTangents.

L'unica tipologia di interpolazione che nel Lottie-JSON viene esplicitamente etichettata è quella di tipo hold. Un oggetto Keyframe può essere indicato come hold ponendo il parametro *h* del costruttore pari a 1, oppure tramite l'unico metodo incluso in questa classe, *setHold(bool)*.

La maggior parte delle proprietà animabili è rappresentata nel Lottie-JSON con la struttura a-k-ix. Il campo *a* è il flag che indica se la proprietà è stata animata o meno, il campo *k* è l'array che contiene il valore della proprietà - se statica - o le informazioni riguardanti i suoi keyframe - se animata, mentre il campo *ix* non è di interesse in questo contesto (serve solo come indice per le expression). Per questa ragione è stata definita una classe che permetta di poter rappresentare una proprietà generica animabile: KeyframeProperty.

La classe KeyframeProperty è stata costruita in modo tale da avere come attributo interno un array di oggetti Keyframe, *keyframes*, inizialmente calcolato a partire dall'array *k* che contiene le informazioni dei valori della proprietà nel Lottie-JSON (ovvero l'array *k* della struttura a-k-ix). Se la proprietà non è stata animata, *keyframes* è vuoto, altrimenti contiene le informazioni sui keyframe che sono stati definiti sulla proprietà in After Effects. La classe offre diversi metodi di customizzazione dei keyframe, i quali agiscono secondo uno schema comune:

- viene modificato *keyframes* secondo il metodo chiamato.
- si ottiene dal nuovo valore di *keyframes* l'array k corrispondente.
- si sostituisce nel Lottie-JSON il campo k con l'array k calcolato precedentemente.

Sono necessari dunque tre metodi interni:

- un metodo di conversione da array k ad un array di oggetti Keyframe, per ottenere l'attributo *keyframes* iniziale.
- un metodo di conversione da un array di oggetti Keyframe ad un array k, per ottenere l'array k aggiornato post-customizzazione.
- un metodo di sostituzione del campo k della struttura a-k-ix nel Lottie-JSON con il nuovo array k calcolato.

È stato scelto questo approccio, invece della modifica diretta dell'array k sul Lottie-JSON, perché le informazioni che rappresentano un keyframe su Lottie sono strettamente interconnesse. In particolar modo, come è stato analizzato precedentemente, il campo y di un elemento dell'array k, è strettamente dipendente dal valore di altri campi. Considerando una generica proprietà di dimensione d, il j-esimo elemento dell'array k, *k[j]*, e gli oggetti di ease in ed ease out definiti al suo interno $k[j].i$ e $k[j].o$ si ha:

$$k[j].o.y[n] = \frac{vel_o[n](k[j+1].t - k[j].t)}{framerate(k[j+1].s[n] - k[j].s[n])} \cdot \frac{inf_o[n]}{100}$$

$$k[j].i.y[n] = 1 - \frac{vel_i[n](k[j].t - k[j-1].t)}{framerate(k[j].s[n] - k[j-1].s[n])} \cdot \frac{inf_i[n]}{100}$$

con $n \in [0, d)$. Il campo y di ease out dipende dal campo x di ease out, e dai campi t ed s del j-esimo e $(j+1)$-esimo elemento di k. Il campo y di ease in dipende dal campo x di ease in, e dai campi t ed s del j-esimo e $(j-1)$-esimo elemento di k.

Ciò significa che, se si vuole mantenere lo stesso valore di velocity impostato su After Effects, ogni qual volta si modifica uno di questi valori è necessario modificare opportunamente anche il valore di y di ease in per il j-esimo e $(j+1)$-esimo elemento di k, ed il valore di y di ease out per j-esimo e $(j-1)$-esimo elemento di k. Mantenere i rapporti costanti e corretti è dunque un compito fondamentale. Per questa ragione si è scelto uno schema di modifica che permettesse di separare il processo di customizzazione

su due dimensioni, quella dei keyframe, e quella dell'array *k*. Ogni metodo di customizzazione definito modifica l'oggetto Keyframe nell'array *keyframes* nel modo opportuno, cambiando per esempio il suo valore o il suo placement temporale. Dopodiché chiama un metodo interno di update dell'array *k*, in modo che venga calcolato sulla base della versione aggiornata e customizzata di *keyframes*. È la funzione di update che si occupa di costruire il nuovo array *k* e che quindi, automaticamente, calcola il corretto rapporto tra i campi per mantenere le informazioni definite nei keyframe.

I metodi di customizzazione della classe KeyframeProperty permettono di:

- cambiare il placement temporale di un keyframe.
- cambiare il valore di un keyframe.
- cambiare l'ease in e/o l'ease out temporale di un keyframe.
- rendere l'interpolazione temporale di un keyframe di tipo hold.
- cambiare le in-tangents e/o out-tangents dell'interpolazione spaziale di un keyframe.
- rendere l'interpolazione spaziale di un keyframe lineare in entrata e/o in uscita.
- aggiungere un keyframe.
- rimuovere un keyframe.
- cambiare la durata di un'animazione (ovvero scalare automaticamente il placement temporale dei keyframe in modo da allungare o accorciare l'animazione definita).

Gli attributi della classe permettono di capire se la proprietà sia animata - tramite un flag specifico - e di visualizzare l'array di keyframe della proprietà, se questa è animata, o il suo valore, se questa è statica.

La classe KeyframeProperty gestisce la proprietà anche nel caso questa non sia animata. In questo caso l'unica operazione che si può compiere sulla proprietà è il cambio di valore, operazione che avviene tramite il metodo *setValue(value)*. Una proprietà statica può comunque essere animata in fase di customization con l'aggiunta di keyframe.

KeyframeProperty modella dunque le proprietà generiche animabili. Com'è stato visto in fase di analisi del Lottie-JSON, però, non tutte presentano la

stessa struttura ed elaborazione dei keyframe. Le proprietà di Position e Anchor Point, pur avendo una struttura diversa, vengono comunque modellate da questa classe (che è stata costruita per gestire questo tipo di eccezione), altre proprietà invece richiedono una gestione dei keyframe leggermente differente o dei metodi aggiuntivi specifici ad essa, e sono dunque istanze di *un'estensione* di KeyframeProperty (come nel caso di proprietà di colore o Source Text).

3.2.2 Classi ausiliari per rappresentare i colori: Color e Gradient

La classe Color permette di istanziare un oggetto che rappresenta una proprietà animabile di colore. È un'estensione di KeyframeProperty. Non aggiunge metodi specifici per la customizzazione, ma è stata creata per poter gestire le differenze tra la struttura JSON di un colore e di una proprietà generica.

La classe Gradient permette di istanziare un oggetto che rappresenta un gradiente di colori. Non è una estensione di KeyframeProperty. La struttura di memorizzazione dei keyframe su Lottie-JSON diverge dalla regola standard e avrebbe richiesto una gestione dedicata. Poiché raramente risulta necessario, si è scelto di non includere il supporto per la customizzazione di keyframe su un gradiente di colori. Tuttavia, è possibile definire dei keyframe su un gradiente in After Effects, e modificarne il valore dei colori tramite la libreria.

Gli attributi di questa classe sono composti da tutte le proprietà che definiscono un gradient: la tipologia, start ed end point, start ed end location, start ed end color. Per i gradienti di tipo *radial* vi sono anche highlight length e highlight angle. Gli attributi di start ed end point sono istanze di KeyframeProperty. I metodi offerti dalla classe permettono di:

- impostare i due colori del gradient, globalmente o su uno specifico keyframe.
- impostare start ed end location del gradiente.

Le modifiche su start point ed end point invece avvengono tramite i metodi di KeyframeProperty.

3.2.3 Classe ausiliaria per rappresentare proprietà di tipo dropdown: Dropdown

La classe Dropdown permette di istanziare una proprietà con struttura a *dropdown menu* (menù a tendina), che sia animabile. È un'estensione di KeyframeProperty. Questa classe non ha metodi specifici per la customizzazione, ma è stata creata per poter gestire le differenze tra una proprietà con struttura a drodpwn menu ed una generica proprietà animabile. Queste proprietà, infatti, prevedono unicamente keyframe di tipo hold, e sono caratterizzate da una numerazione dei valori differente dalla regola standard: solitamente, in una lista o array di valori la numerazione parte da 0, mentre in questo caso la numerazione parte da 1.

3.2.4 Composition

L'oggetto globale che rappresenta il Lottie-JSON è un oggetto della classe Composition. Per poter customizzare il JSON, il primo step da compiere è costruire il seguente oggetto:

```
const comp = new Composition (lottieJSON)
```

dove *lottieJSON* è la variabile che contiene il JSON dell'animazione. D'ora in poi, l'oggetto che rappresenta l'animazione è istanziato, e tutte le funzioni di customizzazione che vengono chiamate andranno a modificare i campi opportuni del JSON contenuto nella variabile lottieJSON.

La classe Composition permette di accedere agli attributi che definiscono una composition. Inoltre, mette a disposizione dei metodi per modificare alcune delle sue proprietà generali: dimensioni (larghezza e altezza), in-point, out-point, e durata.

L'oggetto Composition contiene poi le informazioni riguardanti i layer della composition, organizzate in un array di oggetti della classe Layer. L'accesso ad uno specifico oggetto Layer avviene tramite un apposito metodo di chiamata che accetta come parametro il nome del layer o la sua posizione nell'array di layer (ovvero la posizione nello stack di layer nella composition del progetto).

```
const layer = comp.layer(layerID)
```

3.2.5 Layer

La classe Layer rappresenta un layer di After Effects all'interno di una composition. La struttura di un oggetto rappresentante un layer all'interno del

Lottie-JSON è variabile, poiché dipende dalla tipologia del livello e dagli effetti ad esso applicati. Per ogni tipologia di layer è possibile accedere agli attributi comuni: nome, in-point, out-point, durata e start time, time stretch, visibilità, blending mode, flag 3D e tipologia di livello. Questi attributi - fatta eccezione per il nome, la tipologia, e il flag 3D - sono poi modificabili tramite appositi metodi. Oltre a queste proprietà, sono comuni a tutte le tipologie anche le proprietà di transform. Queste sono accessibili tramite un metodo di chiamata che accetta come parametro il nome della transform.

```
const transform = comp.layer(layerId).transform(
   transformName)
```

Le transform sono proprietà animabili e in quanto tali istanze di KeyframeProperty.

Ci sono poi, invece, una serie di proprietà diverse a seconda della tipologia del livello. È presente una funzione interna alla classe che si occupa di istanziare le opportune proprietà in base al tipo di livello, e renderle accessibili. L'accesso alle proprietà tipiche del livello avviene dunque nel seguente modo:

```
const text = comp.layer(layerId).text
const shape = comp.layer(layerId).shape(shapeName)
const shape = comp.layer(layerId).shape(shapePosition)
const precomp = comp.layer(layerId).preComp
const solid = comp.layer(layerId).solid
const image = comp.layer(layerId).image
const audio = comp.layer(layerId).audio
```

dove gli oggetti restituiti sono rispettivamente istanze delle classi Text, Shape, PreComp, Solid, ImageObject o Audio. Le caratteristiche tipiche di ogni tipologia di livello sono dunque definite e gestite in apposite classi separate.

Ad un layer possono anche essere applicati una serie di effetti. Si accede ad un effetto tramite un metodo di chiamata apposito che prende come parametro il nome dell'effetto:

```
const effect = comp.layer(layerId).effect(effectName)
```

L'oggetto restituito è un'istanza della specifica classe dedicata all'effetto. La libreria supporta solo un insieme limitato di effetti: gli Expression Control e il Gaussian Blur.

3.2.6 Text

La classe Text permette di istanziare un oggetto che contiene e rappresenta le caratteristiche proprie di un layer di tipo Text. In particolar modo un

oggetto Text contiene gli attributi *sourceText* (un oggetto istanza della classe SourceText) e un array *animators* che rappresenta gli animator applicati sul livello di testo tramite oggetti della classe Animator. L'accesso all'oggetto SourceText avviene mediante attributo, mentre gli animator sono accessibili tramite un apposito metodo di chiamata che prende come parametro il loro nome o la loro posizione nell'array *animators* (ovvero nello stack di animators definiti sul livello in After Effects).

```
const sourceText = comp.layer(layerId).text.sourceText
const animator = comp.layer(layerId).text.animator(
    animatorID)
```

3.2.7 SourceText

La classe SourceText permette di istanziare un oggetto che contiene e rappresenta le caratteristiche della proprietà di Source Text di un livello testuale: stringa di testo, font, font size, colore, etc. È un'estensione di KeyframeProperty: Source Text è una proprietà animabile, ma l'elaborazione dei keyframes sul Lottie-JSON devia dalla regola standard e richiede una gestione dedicata. In quanto estensione di KeyframeProperty, tuttavia, eredita i suoi metodi e attributi. Il valore di un oggetto Keyframe che debba essere riferito ad una proprietà Source Text è rappresentato dal contenuto di testo - ovvero la stringa - del keyframe. La libreria si occupa di istanziare automaticamente gli altri elementi che definiscono un keyframe di testo sul Lottie-JSON - font, font size, colore, etc - secondo un template sempre aggiornato. Questi parametri possono poi essere modificati attraverso metodi specifici della classe SourceText.

```
const key = new Keyframe (time, "value")
comp.layer(layerId).text.sourceText.addKey(key)
```

La proprietà Source Text accetta solamente keyframe di tipo hold. Tuttavia non è necessario definire esplicitamente la tipologia di interpolazione, poiché se ne occupa automaticamente la libreria nella definizione della classe SourceText.

In aggiunta ai metodi di KeyframeProperty, la classe rende possibile:

- cambiare la spaziatura tra le linee del testo.
- cambiare il colore del testo, globalmente o su un keyframe.
- cambiare la stringa di testo, globalmente o su un keyframe.
- cambiare il font size del testo, globalmente o su un keyframe.

- cambiare il font del testo, globalmente o su un keyframe.

Il font può essere cambiato solo tra quelli già utilizzati ed inclusi nel progetto After Effects, ovvero quelli inclusi nella lista di font globale del Lottie-JSON.

Il testo è una di quelle proprietà che più vengono coinvolte nel processo di customizzazione di un video. È estremamente comune, per esempio, cambiare la stringa di un testo. Tuttavia, bisogna assicurarsi che questo - a prescindere dal suo contenuto e quindi dalla lunghezza della stringa - sia sempre visibile, e non sfori i limiti spaziali del video. È necessario dunque gestire l'adattabilità del testo, a fronte di modifiche che possano variarne la dimensione.

Sono disponibili due metodi per definire i confini entro il quale deve disporsi un testo: *setMaxWidth(maxWidth)* e *setMaxHeight(maxHeight)*. Sono possibili diverse configurazioni:

- Sul testo viene definito un limite orizzontale. Il testo può adattarsi a questo limite secondo due modalità. La prima - *Responsive Font Size* - prevede che il testo si mantenga su un'unica riga e il font size diminuisca fino a che la larghezza del testo non rientri entro il limite imposto. La seconda - *Fixed Font Size* - mantiene invece inalterato il font size e suddivide il testo in più linee, in modo che la larghezza del testo risultante rientri entro il limite imposto. La modalità di adattamento del testo viene scelta tramite un apposito metodo *setCustomizationMode(mode)*. La modalità selezionata di default è Responsive Font Size.

- Sul testo viene definito un limite orizzontale e un limite verticale. In questo caso il testo è limitato all'interno di un box. Quando una linea di testo oltrepassa il limite orizzontale, si va a capo e si dispone il testo rimanente su una nuova linea. Quando l'altezza del testo - determinata dal numero di linee - supera il limite verticale, si riduce il font size del testo. Con il nuovo font size definito si ripete il processo: si inizia disponendo il testo su un'unica linea e, quando questo supera il limite orizzontale, si dispone il testo rimanente su una nuova linea. Quando questo supera il limite verticale si riduce nuovamente il font size. Si applica ricorsivamente questo processo fino a che l'intero testo non rientra all'interno del box definito.

Se il limite orizzontale non viene definito, il testo non è responsive. La gestione dell'adattabilità del testo è delegata ad un metodo interno specifico della classe, *adaptText()*, che viene chiamato ogni qualvolta si applica sull'oggetto SourceText un metodo che può variare lo spazio occupato dal testo: cambio di line height, font, font size e contenuto del testo. Per sviluppare

adaptText() è stato necessario capire in che modo poter misurare lo spazio occupato dal testo. L'altezza di un blocco di testo altro non è che il prodotto della line height del testo per il numero di righe su cui è disposto. La larghezza del testo è invece più difficile da ottenere, poiché dipende fortemente dalle caratteristiche del font utilizzato. Per poterlo ottenere, si è deciso di appoggiarsi ad Opentype.js, una libreria Javascript che permette di accedere a specifiche informazioni di font TrueType e OpenType. Per ottenere queste informazioni, tuttavia, è necessario fare un loading esplicito del font utilizzato, e dunque gestire una struttura di Promises. Per questa ragione alcuni dei metodi che permettono di ottenere le informazioni riguardanti il testo - come la sua larghezza, il suo contenuto, e il suo font size - devono essere chiamati tramite Promises o pattern async/await. Di un oggetto SourceText si può ottenere - tramite dei metodi e non tramite chiamata di attributi - la larghezza, l'altezza, il colore, il font, il font size, il leading, ed il contenuto, globale o per ogni keyframe.

3.2.8 Animator

La classe Animator permette di istanziare un oggetto che contiene e rappresenta le caratteristiche di un animator applicato ad un livello di testo. Un oggetto Animator ha due attributi fondamentali: *selector*, un oggetto della classe RangeSelector, e *properties*, una Mappa che contiene le diverse proprietà governate dall'animator. Una specifica proprietà è accessibile tramite un metodo di chiamata che prende come parametro il nome della proprietà stessa. Le proprietà supportate dalla libreria sono: opacità, scala, rotazione, skew, skew axis, posizione (o X Position ed Y Position), anchor point, fill color, stroke color e stroke width. Ogni proprietà è un oggetto della classe KeyframeProperty, eccetto le proprietà di colore fill color e stroke color, oggetti della classe Color.

```
comp.layer(layerId).text.animator.selector
comp.layer(layerId).text.property(propertyName)
```

3.2.9 RangeSelector

La classe RangeSelector permette di istanziare un oggetto che contiene e rappresenta le caratteristiche di un selector di tipo Range.

```
comp.layer(layerId).text.animator.selector
```

La classe permette di accedere ai seguenti attributi di un Range Selector:

- start, end, offset, ease high, ease low, smoothness, amount: sono istanze di KeyframeProperty, e dunque modificabili ed animabili tramite i suoi metodi
- shape, based on, units, randomize order: sono valori singoli, non animabili. Il loro valore può essere modificato tramite specifici metodi della classe.

3.2.10 Shape

La classe Shape permette di istanziare un oggetto che contiene e rappresenta le caratteristiche proprie di una shape all'interno di uno Shape Layer.

```
comp.layer(layerId).shape(shapeID)
```

Un oggetto Shape presenta sempre un attributo *path*, un oggetto che rappresenta la proprietà Path della shape. Questo oggetto ha diversi campi in base alla tipologia di shape, e alle proprietà che la definiscono. Tutte queste proprietà sono istanze di KeyframeProperty. Fa eccezione la shape di tipo custom: per queste il path non è definito da delle proprietà specifiche, ma da una serie di punti. In questo caso, l'attributo *path* è un oggetto istanza della classe Path.

```
comp.layer(layerId).shape(shapeID).path
```

Inoltre possono anche essere presenti gli attributi *fill* e *stroke*. L'attributo fill è un oggetto che rappresenta la proprietà Fill della shape. In base alla tipologia di Fill scelta su After Effects, le informazioni riguardanti il colore del fill sono contenute nel campo:

- color, istanza della classe Color, se il fill è un colore unico.
- gradient, istanza della classe Gradient, se il fill è un gradiente di colori.

Sono poi presenti anche gli altri campi che definiscono la proprietà Fill su After Effects.

```
comp.layer(layerId).shape(shapeID).fill
```

L'attributo stroke è un oggetto che rappresenta la proprietà Stroke della shape. In base alla tipologia di Stroke scelta su After Effects, le informazioni riguardanti il colore dello stroke sono contenute nel campo:

- color, istanza della classe Color, se lo stroke è un colore unico.
- gradient, istanza della classe Gradient, se lo stroke è un gradiente di colori.

Sono poi presenti anche gli altri campi che definiscono la proprietà Stroke su After Effects.

```
comp.layer(layerId).shape(shapeID).stroke
```

3.2.11 Path

La classe Path permette di istanziare un oggetto che contiene e rappresenta le caratteristiche della proprietà Path di una shape di tipo custom.

```
comp.layer(layerId).shape(shapeID).path
```

Path estende KeyframeProperty, dunque rende disponibili tutti i suoi metodi. Inoltre sono stati definiti metodi per:

- aggiungere un punto al path della shape, eventualmente su uno specifico keyframe (tenendo conto che non è possibile avere un numero discorde di punti tra i diversi keyframe).
- rimuovere un punto al path della shape, eventualmente su uno specifico keyframe (tenendo conto che non è possibile avere un numero discorde di punti tra i diversi keyframe).
- cambiare la maniglia di ease in di un punto.
- cambiare la maniglia di ease out di un punto.
- ottenere il numero di punti del path.

Inoltre è presente l'attributo *isClosed* per sapere se il path della shape è chiuso o meno.

3.2.12 Solid

La classe Solid permette di istanziare un oggetto che contiene e rappresenta le caratteristiche proprie di un layer di tipo Solid.

```
comp.layer(layerId).solid
```

È possibile accedere, tramite attributi, e modificare, tramite metodi, le seguenti proprietà:

- Colore del Solid. Il colore viene rappresentato con una semplice stringa HEX e non è un'istanza della classe Color, poiché non è animabile su After Effects.
- Larghezza del Solid.
- Altezza del Solid.

3.2.13 ImageObject

La classe ImageObject permette di istanziare un oggetto che contiene e rappresenta le caratteristiche proprie di un layer di tipo Image.

```
comp.layer(layerId).image
```

Di un image layer è possibile conoscere la larghezza, l'altezza, il path e il nome della risorsa immagine. È anche presente un flag che indica se l'immagine sia stata incorporata nel Lottie-JSON o meno. In caso affermativo, l'attributo di path perde di significato, poichè non è presente alcun file immagine esterno di riferimento. L'attributo name contiene il dato incorporato nel JSON. È presente un unico metodo il quale permette di sostituire la risorsa immagine del livello, fornendo il path o l'url all'immagine scelta o la variabile che contiene i dati dell'immagine incorporata. Le immagini devono spesso essere adattate nel processo di customizzazione. Per esempio, se si definisce un image layer come sfondo di un video, ci si deve assicurare che - nonostante le dimensioni della nuova immagine scelta - questa riempia sempre interamente lo sfondo. Il metodo di sostituzione, per questa ragione, assegna di default alla nuova risorsa immagine le dimensioni di quella originale: se l'immagine originale è larga w pixel e alta h pixel, la nuova immagine viene riscalata e tagliata in modo tale da mantenere le dimensioni $w \times h$, senza essere deformata. Tuttavia, è possibile cambiare le dimensioni della risorsa immagine a piacere, inserendo esplicitamente quelle desiderate come parametro del metodo di sotituzione.

3.2.14 Audio

La classe Audio permette di istanziare un oggetto che contiene e rappresenta le caratteristiche proprie di un layer di tipo Audio.

```
comp.layer(layerId).audio
```

Di un Audio Layer è possibile conoscere il path e il nome della risorsa audio. È anche presente un flag che indica se l'audio sia stato incorporato nel Lottie-JSON o meno. In caso affermativo, l'attributo di path perde di significato, poiché non è presente alcun file audio esterno di riferimento. L'attributo name contiene il dato incorporato nel JSON. È presente un unico metodo, il quale permette di sostituire la risorsa audio del livello, sia nel caso questa venga indicata con un path ad una risorsa, sia nel caso questa venga incorporata nell'oggetto.

3.2.15 PreComp

La classe PreComp permette di istanziare un oggetto che contiene e rappresenta le caratteristiche proprie di un layer di tipo Precomp.

```
comp.layer(layerId).preComp
```

Un oggetto PreComp presenta al suo interno un array che contiene gli oggetti di classe Layer che rappresentano i layer definiti all'interno della precomposizione. È possibile accedere ad uno di questi layer tramite un metodo di chiamata specifico, che accetta come argomento il nome del layer o il suo indice di posizione nell'array (o nello stack di layer nella precomposizione su After Effects):

```
comp.layer(layerId).preComp.layer(layerID)
```

Sull'oggetto restituito da questo metodo possono essere applicate tutte le funzioni di customizzazione della classe Layer.

3.2.16 Expression Controls

Sono state realizzate specifiche classi per rappresentare i diversi effetti di tipo Expression Control:

- SliderControl, PointControl, DDDPointControl, AngleControl, CheckboxControl estendono KeyframeProperty.
- ColorControl estende Color.
- DropdownMenuControl estende Dropdown.

3.2.17 BlurEffect

La classe BlurEffect permette di istanziare un oggetto che rappresenta l'effetto Gaussian Blur. Un oggetto BlurEffect è caratterizzato da attributi che rappresentano le proprietà che definiscono l'effetto:

- blurriness, istanza di KeyframeProperty. Indica la quantità di sfocatura da applicare.
- direction, istanza di Dropdown. Indica se la sfocatura debba essere applicata sulla direzione orizzontale e/o verticale.
- repeatEdgePixels, istanza di KeyframeProperty. Indica se i contorni della forma su cui si applica la sfocatura debbano essere mantenuti netti o meno.

BlurEffect non è dotata di metodi propri poiché le proprietà dell'effetto vengono customizzate tramite i metodi di KeyframeProperty.

Capitolo 4

L'Applicazione della Libreria di Personalizzazione in Progetti di Video Automation

Nel percorso di tirocinio in Algo è stato richiesto il testing ed utilizzo della libreria di customizzazione nella realizzazione di due progetti di video automation Lottie-based. Questi progetti sono stati pensati per essere inseriti nella sezione Demo del sito web dell'azienda: una sezione che permette ai visitatori del sito di testare e capire cosa sia la video automation. L'utente può scegliere diverse demo da testare. Viene richiesto uno o una serie di input, in base ai quali viene realizzato e visualizzato - nel minor tempo possibile - il relativo video data-driven. I video vengono visualizzati nel browser, e possono poi essere scaricati tramite l'inserimento della propria mail su un form apposito. Per questa ragione i progetti Demo sono Lottie-based: i video risiedono su un sito web e possono essere dunque mostrati come web-animation. Inoltre è bene che siano renderizzati e visualizzati in un tempo breve, per poter enfatizzare le potenzialità della video automation. Queste demo hanno rappresentato dunque l'occasione perfetta per testare, migliorare, ed utilizzare la libreria di customizzazione realizzata.

L'idea alla base della realizzazione dei progetti è stata quella di fornire un sostegno visivo per l'anteprima di un ipotetico podcast. In quanto demo ed anteprima, questi video hanno una durata limitata e piuttosto breve, non oltre i 30 secondi: non sono stati creati per supportare un intero episodio di un podcast, quanto più come *plus* visivo per una preview. Entrambi i progetti sono stati realizzati sia in formato 1:1 sia in formato 9:16. Si descrivono in seguito i due progetti e i dettagli della loro realizzazione, sia per quanto riguarda il template creato su After Effects, sia per quanto riguarda la Lottie-customization.

4.1 Progetto Waveform

Il primo progetto realizzato è stato il progetto Waveform. Questa demo permette di caricare un file audio ed ottenere un video con una "forma d'onda" animata in base alla traccia audio inserita.

La waveform è la feature principale del video. Sono presenti tuttavia ulteriori input di personalizzazione. In particolar modo all'utente viene richiesto:

- La traccia audio della preview del podcast.
- Il nome dello speaker.
- Il titolo del podcast, o dell'episodio a cui l'anteprima si riferisce.
- Un'immagine di copertina rappresentativa del podcast.
- Un link al sito del podcast.

Sono questi campi quelli che determinano la customizzazione del video risultante. Di questi input, sono tutti obbligatoriamente richiesti, fatta eccezione per il link al podcast. Se questo non viene inserito, verrà visualizzato il link alla sezione Demo del sito web di Algo.

Il video può essere suddiviso in quattro macro-momenti di animazione differenti:

- Intro, l'animazione di comparsa dell'immagine, e l'animazione di entrata dei testi.
- Waveform, l'animazione della forma d'onda.
- Outro, l'animazione di scomparsa dell'immagine, e l'animazione di uscita dei testi.
- CTA, la comparsa della *call to action* finale, ovvero del link al podcast.

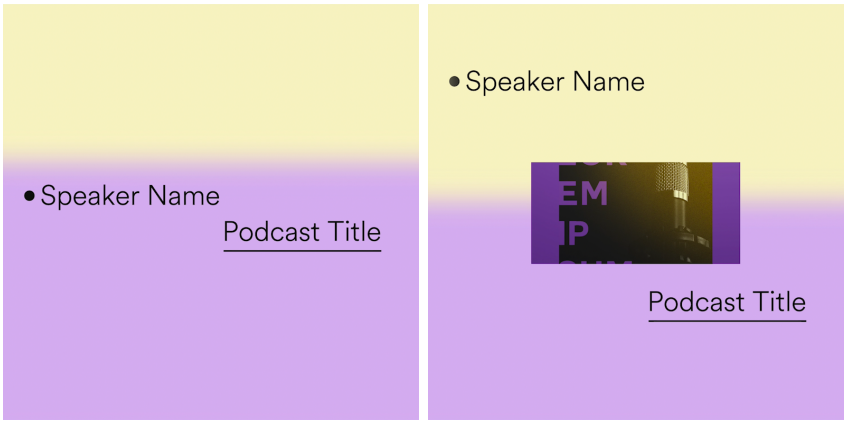


Figura 4.1: Intro

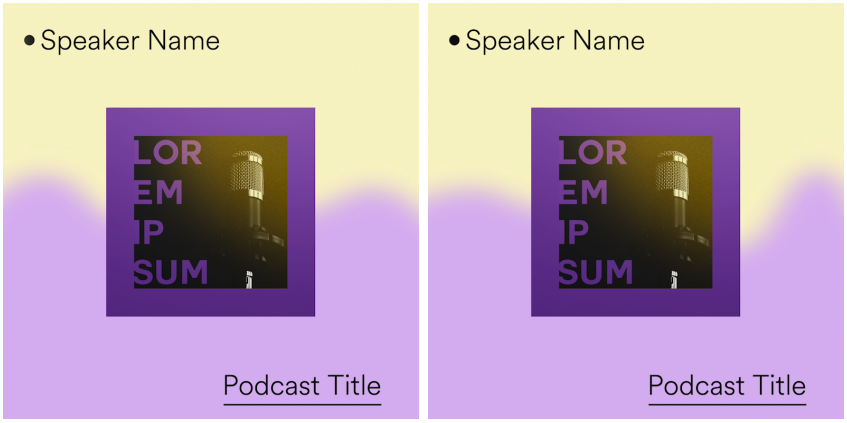


Figura 4.2: Waveform

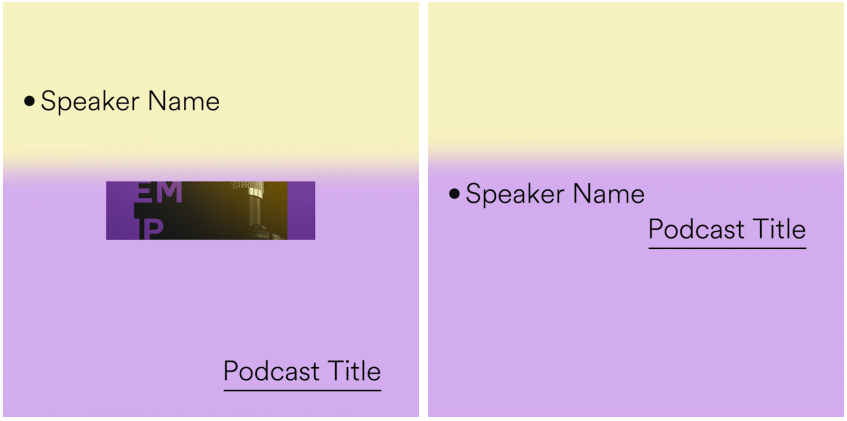


Figura 4.3: Outro

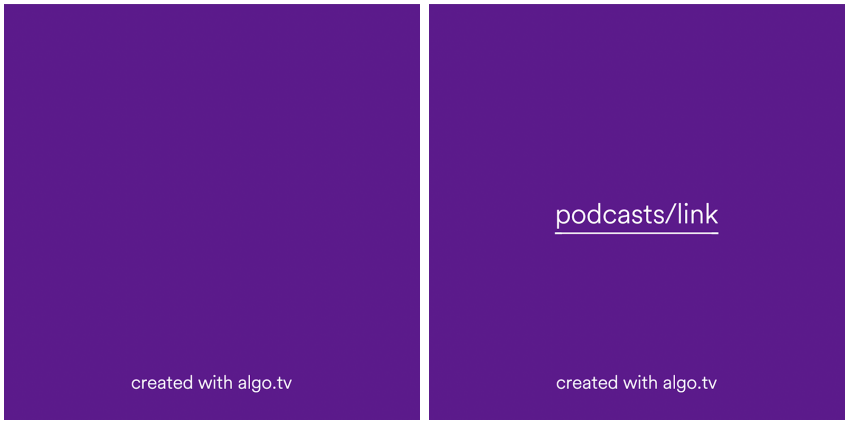


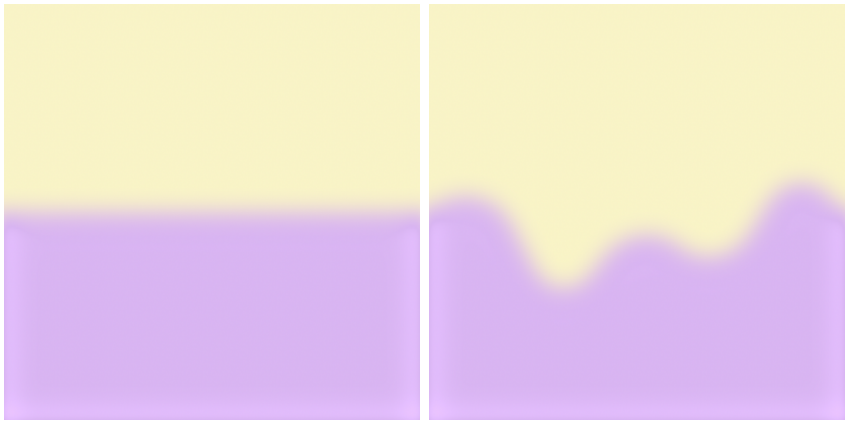
Figura 4.4: CTA

4.1.1 Sviluppo: After Effects

Si descrive in questo paragrafo la realizzazione del template video del progetto su After Effects. Non verranno descritte tutte le features del video, ma solo quelle che saranno soggette a customizzazione.

Waveform

La waveform è stata realizzata tramite uno shape layer di tipo custom, su cui sono stati definiti una serie di punti "mobili". È il movimento di questi punti che crea l'effetto della forma d'onda.

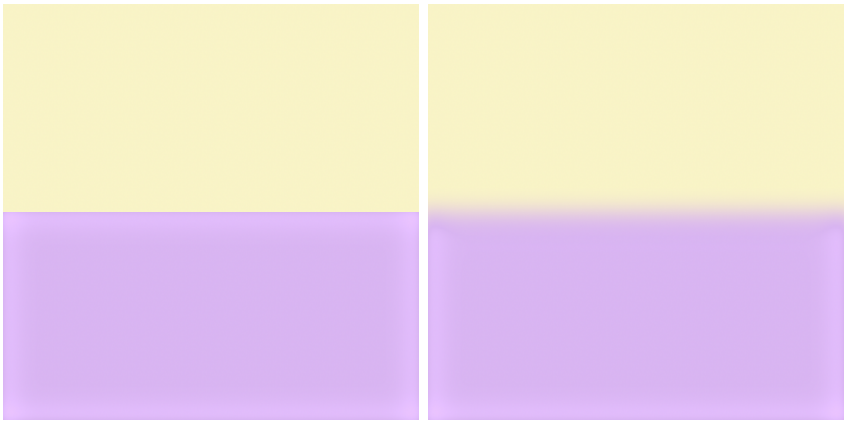


(a) Waveform in quiete (b) Waveform animata

Figura 4.5: Elemento Waveform

I punti della shape non sono stati inseriti manualmente, ma creati tramite expression. Nello specifico, ogni punto è legato ad un null layer, in modo tale da condividerne la posizione nella composition: se si muove il null layer, si muove il punto corrispondente. Per animare la waveform è necessario dunque animare la posizione dei null layer. La waveform si anima sulla base della traccia audio inserita dall'utente. Nello specifico, per ogni frame del video si deve definire la posizione dei null layer sulla base del "valore dell'audio" preso in considerazione in quel frame. Benché questo processo possa avvenire in maniera completa solamente in fase di customizzazione - una volta che si hanno a disposizione i dati di input - è possibile definire già in fase di progetto in che modo varia la posizione dei diversi null layer al cambiare di un valore di riferimento - quello che sarà successivamente il valore fornito dai dati audio. Si adotta dunque un approccio ibrido tra expression e scripting di customizzazione. Il valore di riferimento, che sarà poi popolato dai valori della traccia audio, è rappresentato da un effetto di tipo Slider Control, il quale espone un valore decimale che può essere utilizzato e inserito all'interno di una expression. Ogni null layer presenta una expression sulla proprietà di Y Position, la quale lega la sua posizione verticale al valore dello slider control. In particolar modo, viene calcolato sulla base del valore dello slider control un offset verticale da applicare alla posizione. Alternativamente, ogni null layer somma o sottrae questo offset alla propria posizione di quiete. Inoltre, tra un null layer e l'altro viene introdotto un *delay*, in modo tale da evitare che tutti i punti si muovano con lo stesso offset nello stesso istante. Se un valore n dello slider control provoca un offset o al tempo t nell'i-esimo null layer, questo stesso offset sarà applicato al j-esimo null layer al tempo $(t + delay)$, dove *delay* è un valore che dipende dall'indice del null layer. È la presenza del delay tra i punti che aiuta a dare la sensazione di movimento ed evoluzione nel tempo nella forma d'onda. Cambiando il valore dello slider control cambia la posizione dei null layer e con essi la posizione dei punti relativi. Il movimento della waveform sarà dunque creato animando il valore dello slider control sulla base dei dati ottenuti dalla traccia audio.

Una delle principali caratteristiche grafiche della waveform - e *fil rouge* dell'estetica progetto - è che il suo aspetto e i suoi contorni risultino sfumati. Questa resa grafica è stata ottenuta mediante l'applicazione dell'effetto Gaussian Blur sullo shape layer della waveform, deselezionando l'opzione Repeat Edge Pixels, la quale opera come in modo tale da mantenere i contorni della forma netti.



(a) Repeat Edge Pixels: on (b) Repeat Edge Pixels: off

Figura 4.6: Differenti configurazioni dell'effetto Gaussian Blur

Su questo effetto si è riscontrata una piccola problematica di supporto su Lottie, relativa alla resa grafica su Safari. In questo caso l'effetto dava lo stesso risultato ottenuto con l'opzione di Repeat Edge Pixels selezionata: il blur non veniva applicato ai limiti dello shape layer, risultando in confini molto netti, ed in contrasto con le linee guida di grafica richieste. La problematica risiedeva nella zona di confine dello shape layer. La soluzione adottata si è basata dunque sul ridefinire questa zona: nello stesso shape layer della waveform è stata inserita una seconda shape rettangolare, con le stesse dimensioni della composition, resa poi non visibile. Di questo modo i confini globali dello shape layer non coincidono più con i confini della waveform, ma con quelli della shape ausiliaria, che si estende oltre la forma d'onda. Questa stessa soluzione è stata adottata per gli altri elementi su cui è stato applicato l'effetto Gaussian Blur.

Immagine

L'immagine è costituita da un image layer ed uno shape layer rettangolare utilizzato come maschera (track matte layer). È lo shape layer che viene animato per creare l'animazione di comparsa dell'immagine.

Testo - Speaker

Il nome dello speaker è rappresentato da una struttura formata da un text layer, che contiene il campo testuale - il *name layer* - ed uno shape layer circolare - il *dot layer* - imparentato al text layer.

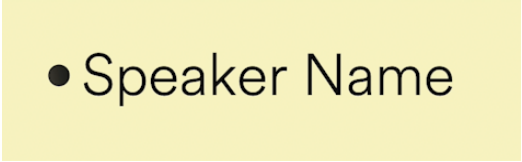


Figura 4.7: Text layer e dot layer.

È stata definita un'animazione di entrata e di uscita sul name layer, sulla sua proprietà di Y Position. Il dot layer presenta come parent il name layer, e dunque segue l'animazione definita.

Poiché l'unico elemento mobile nella scena della waveform è la waveform stessa, sono state definite delle animazioni sui testi, così da spezzare la monotonia del video. L'animazione viene creata mascherando gradualmente e progressivamente una porzione del testo.

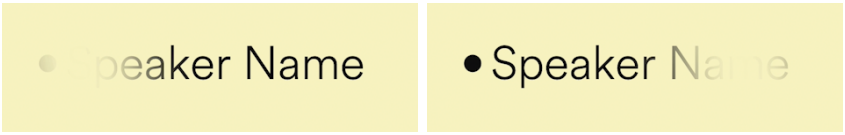


Figura 4.8: Animazione sul text layer e dot layer.

La maschera viene realizzata tramite uno shape layer che viene impostato come track matte sia del name layer sia del dot layer. Ci si riferisce a questo livello come *animation layer*. L'effetto di sfumatura si ottiene applicando su di esso un effetto di Gaussian Blur.

Su After Effects un unico livello può essere impostato come track matte di diversi livelli. Su Lottie, tuttavia, questo non è possibile: un livello può essere track matte di un solo altro livello. Nel Lottie-JSON, infatti, le informazioni riguardo il track matte sono distribuite sugli oggetti relativi ai due attori coinvolti: il livello che viene mascherato contiene le informazioni riguardo la modalità del track matte - normale o inverted - mentre il livello che maschera contiene le informazioni riguardo la proprietà da cui si ricava la trasparenza - luminosità o canale alpha. I livelli coinvolti vengono collegati in base alla loro posizione nell'array di oggetti-layer: un track matte layer agisce solo, eventualmente, sul livello successivo nell'array. In questa animazione i livelli che devono essere mascherati sono due: il dot layer e il name layer. Per questa ragione, è stato necessario duplicare l'animation layer, così che ognuno dei due livelli da mascherare potesse avere il track matte dedicato. Inoltre, affinchè venga elaborato correttamente da Lottie, è necessario mantenere un ordine preciso tra i livelli mascheranti e i livelli mascherati, in modo tale da rispettare l'associazione dei livelli nell'array del Lottie-JSON: il livello

mascherante deve essere sempre sopra al rispettivo livello mascherato nello stack dei layer.

L'animation layer viene animato nella proprietà di posizione orizzontale, perché percorra l'intera lunghezza del testo: l'animazione dipende dunque dai dati di input, poiché lo spostamento globale cambia in base allo spazio occupato dalla stringa di testo, e pertanto dovrà essere opportunamente adattata in fase di customizzazione.

Poiché è presente l'effetto Gaussian Blur, si è riscontrato lo stesso problema discusso precedentemente, risolto attraverso la stessa soluzione.

Testo - Titolo del Podcast

Il titolo del podcast è rappresentato da una struttura formata da un text layer, che contiene il campo testuale - il *title layer* - ed uno shape layer rettangolare - l'*underline layer* - imparentato al text layer.

Figura 4.9: Text layer e underline layer.

La struttura e le animazioni riprendono esattamente quelle descritte precedentemente.

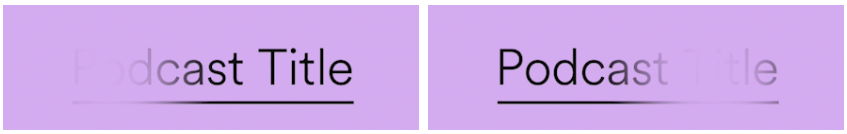


Figura 4.10: Animazione sul text layer e underline layer.

In questo caso è assente il dot layer, ma è presente l'underline layer che deve essere opportunamente gestito.

L'underline layer è stato realizzato mediante uno shape layer rettangolare, con stroke assente. Inizialmente era stato creato come una semplice linea, con stroke pari a 4px. Tuttavia, questa modalità creava problemi con l'animazione definita dall'animation layer. Si è osservato infatti che il track matte funziona correttamente su Lottie - in modalità inverted - quando deve mascherare forme su cui sia stato definito un fill.

La linea dell'underline layer deve essere lunga quanto il testo: la dimensione del rettangolo che la definisce deve adattarsi dunque alla stringa di

testo definita dall'utente. È stato dunque necessario pensare a come gestire la sua adattabilità ai dati in input. Così come nella waveform, si è deciso di costruire la shape rettangolare mediante expression, legando i suoi punti a quattro null layer. Questi null layer vengono opportunamente adattati e riposizionati in fase di customizzazione, in modo tale da essere posizionati all'inizio e alla fine del testo. Per facilitare questo processo, si definiscono via expression alcuni rapporti costanti tra i null layer: quelli inferiori vengono impostati di modo tale da avere sempre la stessa distanza dai superiori, e la stessa X Position del null layer superiore corrispondente.

CTA

La scena della call to action è stata definita in una precomposition, inserita poi come layer nella composition principale. Questa scena presenta un text layer che rappresenta il link al podcast - il *link layer* - ed una linea che sottolinea il testo - l'*underline layer*.

Figura 4.11: Text layer e underline layer.

La linea è stata realizzata mediante un semplice stroke, poiché - a differenza del caso precedente - non è necessario gestire alcun track matte. La linea deve estendersi per tutta la lunghezza del testo, ed è stato dunque necessario gestirne l'adattabilità. Si applica lo stesso approccio descritto precedentemente. I punti dello shape layer sono definiti mediante expression e legati alla posizione di specifici null layer che, in fase di customizzazione, si posizionano all'inizio e alla fine del testo.

Audio

Sono stati inseriti degli effetti sonori standard per accompagnare le animazioni di intro, outro e CTA, ed un audio layer con una sorgente placeholder che in fase di customizzazione conterrà la traccia audio caricata dall'utente.

4.1.2 Sviluppo: Customizzazione via JavaScript

Si descrive in questo paragrafo la gestione della personalizzazione del video, realizzata tramite uno script JavaScript e con il supporto della libreria di customizzazione.

Waveform

La waveform è la feature principale del progetto: questa viene animata in base alla traccia audio inserita dall'utente. Si ottiene - tramite un'elaborazione separata dalla customizzazione, che avviene precedentemente in fase di raccolta dei dati di personalizzazione - un array di campioni audio che rappresenta la traccia. Questo array contiene i dati del volume dell'audio, normalizzati in scala 0-1.

Il movimento dei singoli punti che definiscono la forma d'onda è stato precedentemente impostato su After Effects tramite expression, sulla base del valore assunto da uno slider control. Dal lato della customizzazione, dunque, è necessario modificare solamente il valore dello slider in base ai campioni audio a disposizione: per ogni campione si aggiunge un keyframe alla proprietà tramite il metodo *addKey(keyframe)* della classe KeyframeProperty. Nello specifico, per ogni i-esimo campione valutato, si aggiunge un keyframe lineare che ha come placement temporale il secondo identificato dal rapporto tra i e il framerate del video, sommato all'offset temporale dell'intro, e come valore il valore del campione. Una volta compiuta questa operazione, la waveform si anima di conseguenza.

```
let wSlider = comp.layer('Expression Controls').effect('
    Waveform')

//Remove potential previous keyframes.
for (let i = wSlider.keyframes.length-1; i >= 0; i--) {
    wSlider.removeKey(i)
}

//Add keyframes for every audio sample.
for (let i = 0; i < audioData.length; i++) {
    let key = new Keyframe ((i/comp.framerate)+introLength
        , audioData[i])
    wSlider.addKey(key)
}
```

L'array di campioni restituito dalla fase di raccolta dati, in realtà, viene elaborato prima di definire la customizzazione dello slider control della waveform. Se fosse utilizzato come riferimento per il valore dei keyframe senza alcuna elaborazione, infatti, l'animazione risultante sarebbe estremamente

scattosa: il segnale audio registra anche rumore e fenomeni istantanei, che introducono grande variabilità tra valori di campioni adiacenti. È necessario filtrare il segnale per ridurne il rumore ed uniformarne i valori, attraverso un opportuno *filtro di smoothing*. Un filtro di smoothing agisce sul segnale in modo da ottenere una distribuzione più uniforme tramite riduzione del rumore. Le funzioni di smoothing riassegnano ad ogni campione dell'array un valore basato sui valori dei campioni adiacenti, in modo tale da ridurre la variabilità tra i campioni stessi. Su After Effects questa operazione avviene tramite l'expression dedicata *smooth(width, samples)*. Il parametro *width* indica la larghezza - in secondi - della finestra di azione del filtro, mentre il parametro *samples* indica il numero di campioni distribuiti equamente sulla finestra da prendere in considerazione per la valutazione del nuovo valore. L'expression di smooth può essere definita direttamente sul valore dello slider control in cui sono definiti i keyframe. Questa expression, tuttavia, non è supportata su Lottie. È stato dunque necessario replicare questo processo via JavaScript. Si è scelto di implementare un'operazione di smoothing basata su un filtro a media mobile. Il filtro a media mobile valuta il campione i preso in considerazione e una finestra di larghezza n campioni, centrata su i: al campione i viene assegnato come valore la media dei campioni inclusi nella finestra. Questa funzione prende dunque in input l'array di campioni, e il numero di campioni che identifica la lunghezza della finestra. Solitamente si sceglie una lunghezza della finestra dispari, in modo tale da considerare nella media anche il valore del campione su cui il filtro sta agendo.

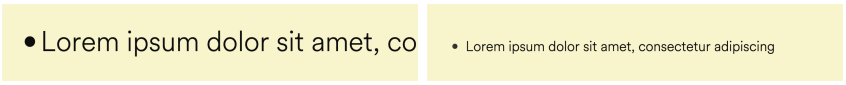
Per i campioni "di confine", si usa la tecnica di *zero padding*, ovvero si considerano tutti i valori della finestra che eccedono l'array pari a zero.

Esistono filtri di smoothing ancora più precisi nei riguardi del segnale originale di quello utilizzato. La waveform, tuttavia, deve rappresentare il segnale audio in input in modo tale da creare coerenza tra l'animazione e la traccia audio, ma non deve essere una riproduzione fedele al dettaglio. Per questa ragione si è scelto di mantenere il filtro a media mobile analizzato.

Testi - Speaker, Titolo e Link del Podcast

Sono presenti tre differenti campi testuali personalizzabili in base all'input dell'utente: il nome dello speaker e il titolo del podcast, visibili nella scena di intro, outro e waveform, ed il link al podcast, visibile nella scena CTA.

Ogni qualvolta siano presenti testi customizzabili, una delle operazioni da compiere è assicurarsi che il testo rimanga correttamente visibile, ovvero che la sua dimensione o la sua disposizione si adatti in base alla stringa di testo inserita.



(a) Text layer non adattato (b) Text layer adattato

Figura 4.12: Differenza tra un text layer non *responsive* e *responsive*.

Poiché è un'operazione ricorrente, il supporto all'adattabilità del testo è nativamente incluso nella libreria di customizzazione. Per attivarlo, è semplicemente necessario inserire i limiti di larghezza e/o altezza che il testo deve osservare. In questo caso tutti e tre i campi testuali si mantengono su una sola linea: è stato impostato solo un limite di larghezza tramite il metodo *setMaxWidth(maxWidth)* mantenendo la modalità di adattabilità di default, Responsive Font Size (quando il testo customizzato supera il limite massimo di larghezza, il font viene automaticamente diminuito fino a che il testo non rientra entro il limite).

Il valore dei campi testuali viene modificato tramite l'apposito metodo *setText(text)* della classe SourceText, estensione di KeyframeProperty.

```
let title = comp.layer('Podcast Title').text.sourceText

//Set width bound.
title.setMaxWidth(textMaxWidth)

//Set text value.
title.setText(podcastTitle)
```

I testi sono accompagnati da specifici elementi grafici.

Il nome dello speaker è accompagnato da un *dot*, realizzato tramite uno shape layer circolare. Se il font del nome dello speaker viene diminuito a fronte della customizzazione, allora è necessario ridimensionare lo shape layer in modo tale da mantenere le corrette proporzioni tra dot e testo. Per questa ragione, è stata creata una funzione che modifichi il valore di size del path ellittico dello shape layer in base all'altezza del testo corrispondente. Poiché il testo si dispone sempre su una sola riga, l'altezza corrisponde al font size. Oltre alla dimensione del dot, anche la sua posizione deve essere controllata e gestita: il dot deve essere sempre centrato verticalmente rispetto al testo. Nella stessa funzione, dunque, si modifica la transform di Y Position in base all'altezza del testo.

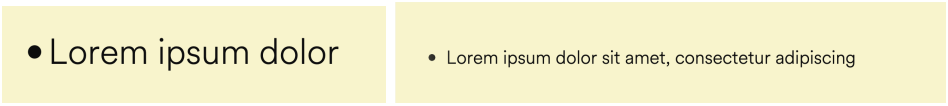


Figura 4.13: Adattabilità della struttura che rappresenta il nome dello speaker.

Sia la proprietà di size di una shape, sia la transform di posizione verticale sono istanze di KeyframeProperty, dunque il loro valore può essere facilmente modificato con il metodo *setValue(value)* definito dalla classe.

```
async function adaptDot (sourceTextObj, dot) {
    let height = await sourceTextObj.getFontSize()
    let dotSize = dot.shape(0).path.size.value[0]

    let newSize = (3/8)*height
    let value = (1/3)*height

    //Adapt dot size.
    dot.shape(0).path.size.setValue([newSize, newSize])

    //Adapt dot position.
    dot.transform('Y Position').setValue(-value)
}
```

Il titolo del podcast è invece accompagnato da un'underline, realizzata attraverso uno shape layer rettangolare. Questa forma è stata costruita su After Effects tramite expression, associando la posizione di ogni punto alla posizione di un relativo null layer. La linea deve estendersi per la lunghezza del testo.

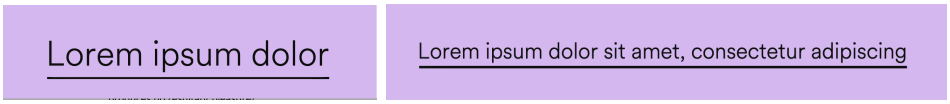


Figura 4.14: Adattabilità della struttura che rappresenta il titolo del podcast.

Il testo è allineato a destra: per modificare dunque la lunghezza della linea, si modifica la proprietà di X Position del null layer collegato al vertice superiore sinistro della shape, distanziandolo della lunghezza del testo dal null layer collegato al vertice superiore destro. I null layer inferiori sono stati impostati - via expression - per mantenere sempre la stessa X Position del corrispettivo superiore. Inoltre, anche in questo caso è necessario correggere la proprietà di Y Position della linea in base all'altezza del testo, ovvero - poiché rimane sempre su una sola linea - in base al suo font size: se aumentasse si potrebbe sovrapporre il testo alla linea, se diminuisse la distanza tra

testo e linea potrebbe risultare eccessiva. È necessario mantenere sempre lo stesso rapporto tra distanza e font size. Per fare ciò si modifica la proprietà di Y Position dei null layer che definiscono la parte superiore della linea. I null layer inferiori sono già stati impostati - via expression - per mantenere sempre una distanza fissa da quelli superiori, garantendo così uno spessore costante.

```
async function adaptLine (sourceTextObj, lineLayer,
   nullRefR, nullRefL) {

let width = await sourceTextObj.getTextWidth()
let height = await sourceTextObj.getFontSize()

//Adapt length of line.
let nRPos =  nullRefR.transform('X Position').value
nullRefL.transform('X Position').setValue(nRPos - width)

//Adapt vertical position of the line.
nullRefR.transform('Y Position').setValue(height/3)
nullRefL.transform('Y Position').setValue(height/3)
}
```

Il link del podcast presenta anche esso una linea di underline: anche in questo caso è necessario adattare la lunghezza della shape e la sua posizione in base alla larghezza e altezza del testo di riferimento. Vi sono due differenze rispetto alla struttura composta dal titolo del podcast e relativa linea. In questo caso, la linea è realizzata mediante un semplice stroke, ed è dunque governata da due soli null layer, uno di destra e uno di sinistra. Inoltre, il testo è allineato al centro, dunque entrambi i null layer dovranno essere spostati per adattare la lunghezza della linea al testo. In particolar modo, per assegnare la posizione corretta ai null layer, si somma e si sottrae alla posizione centrale della composition la metà della lunghezza del testo.

L'ultimo elemento da gestire, dal punto della customizzazione, sono le animazioni definite sulle strutture dei testi della scena principale - nome dello speaker e titolo del podcast, con rispettivi elementi grafici. Queste devono essere adattate su due dimensioni: spaziale, poiché ci si deve assicurare che percorrano l'intera lunghezza del testo relativo, e temporale, poiché devono essere distribuite e distanziate temporalmente rispetto alla lunghezza della scena.

Per quanto riguarda l'adattabilità temporale, ci si deve assicurare che vi sia un gap sufficiente tra l'animazione sul testo dello speaker e sul testo del titolo, e che siano più o meno equamente spaziate rispetto all'inizio e alla fine della scena della waveform. Inoltre, bisogna anche verificare che la

durata della traccia audio sia sufficientemente lunga per poter visualizzare le animazioni, e gestire la situazione altrimenti.

Ogni animazione viene definita rispettivamente da due keyframe sulla proprietà di X Position dello shape layer mascherante (animation layer). Secondo quanto richiesto in fase di design, le due animazioni devono essere lente e distanziate. Dunque, nel caso la scena sia breve - minore di 12 secondi - i keyframe vengono rimossi, e con essi le animazioni.

```
nameAnimation.transform('x position').removeKey(1)
nameAnimation.transform('x position').removeKey(0)
titleAnimation.transform('x position').removeKey(1)
titleAnimation.transform('x position').removeKey(0)
```

Poichè nel processo di customizzazione è possibile dunque che i keyframe vengano rimossi, per distanziare e spostare temporalmente le animazioni non è sufficiente modificare il placement temporale dei keyframe, ma è necessario crearli ad aggiungerli esplicitamente. Le animazioni vengono distanziate equamente in base alla durata dell'audio, assicurandosi sempre che vi siano - almeno - 6 secondi di distanza tra di loro.

```
if (audioData.length/comp.framerate >= 12) {
  var middlePoint = ((audioData.length/comp.framerate)/2
     - introLength)/2
  var ease = new Ease (30, 0)

  //Add keyframes to name animation.
  var middlePointName = (audioData.length/comp.framerate)
     /2 + introLength - middlePoint
  var nAnim1 = new Keyframe(middlePointName-3, -180, ease
     , ease)
  var nAnim2 = new Keyframe(middlePointName+3, 1260, ease
     , ease)
  nameAnimation.transform('x position').addKey(nAnim1)
  nameAnimation.transform('x position').addKey(nAnim2)

  //Add keyframes to title animation.
  var middlePointTitle = (audioData.length/comp.framerate
     )/2 + introLength + middlePoint
  var tAnim1 = new Keyframe(middlePointTitle-3, 1260,
     ease, ease)
  var tAnim2 = new Keyframe(middlePointTitle+3, -180,
     ease, ease)
  titleAnimation.transform('x position').addKey(tAnim1)
  titleAnimation.transform('x position').addKey(tAnim2)
}
```

L'animation layer si sposta in modo tale da percorrere l'intera lunghezza del testo. Adattare l'animazione dal punto di vista spaziale significa dunque

cambiare opportunamente i valori dei keyframe che la definiscono, inserendo i valori di posizione orizzontale di inizio e fine testo corretti e calcolati sulla base della sua lunghezza.

Immagine

L'immagine della scena è quella scelta dall'utente: si sostituisce la risorsa immagine nell'image layer relativo tramite il metodo offerto dalla libreria di customizzazione *setSource(img, mode)*. Non sono necessari specifici controlli riguardo al ridimensionamento e all'adattabilità dell'immagine, poiché vengono automaticamente gestiti dal metodo della classe ImageObject.

```
let img = comp.layer('Image').image
//Customize image (as embedded data).
img.setSource(imgData, 'embedded')
```

Assegnazione dei Colori

Uno dei task di customizzazione più importanti - e più delicati - è stata l'assegnazione dei colori ai diversi elementi della scena. Gli elementi del video il cui colore viene customizzato sono la waveform, lo sfondo nelle scene di intro, outro e waveform, e lo sfondo nella scena di CTA. La guideline fornita dall'azienda richiedeva che i colori fossero definiti automaticamente a partire dall'immagine-input dell'utente, in modo tale da creare un video che fosse coerente visivamente con l'immagine e con il branding del podcast. Per la waveform e per lo sfondo generale è stato richiesto che la palette dei colori risultante fosse di tonalità *pastello*, sia per ragioni stilistiche di design, sia per garantire la leggibilità dei testi del video.

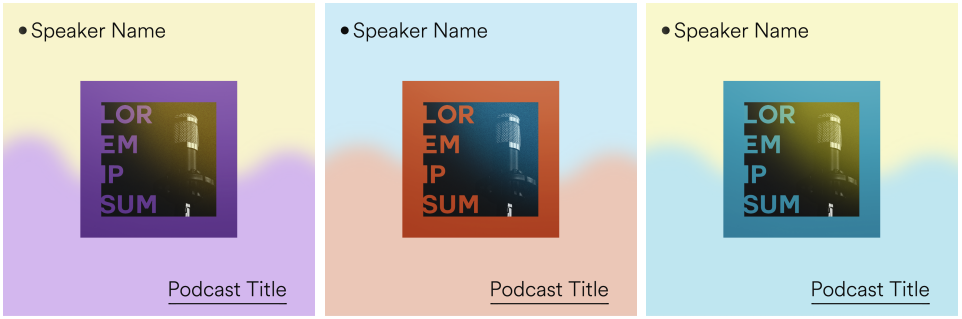


Figura 4.15: Associazione dei colori della scena waveform in base all'immagine.

È stato fondamentale capire e testare in che modo assegnare e trasformare i colori non solo per rispettare le indicazioni di design del progetto,

ma, soprattutto, per garantire la visibilità della waveform per ogni immagine caricata. I colori, infatti, devono essere tali da permettere di identificare il movimento della forma d'onda dell'audio - devono dunque garantire un contrasto di saturazione, tonalità e luminosità adeguato - e di poter sempre leggere i campi testuali della scena. Innanzitutto è stato necessario comprendere come recuperare la palette di colori dell'immagine. A questo scopo, ci si è appoggiati a Color Thief[1], una libreria JavaScript che permette di ottenere la palette di colori di un'immagine e il suo colore dominante. Per analizzare in che modo sono stati elaborati i dati di colore, si prende come riferimento lo spazio di colore HSV (hue-saturation-value): ogni colore viene rappresentato dalla sua tonalità, saturazione e luminosità.

Waveform
Il colore della waveform deriva dal colore dominante dell'immagine restituito da Color Thief. Il colore ottenuto viene reso pastello: la sua luminosità viene impostata su un valore piuttosto alto, la sua saturazione su un valore basso. La tonalità invece rimane invariata.

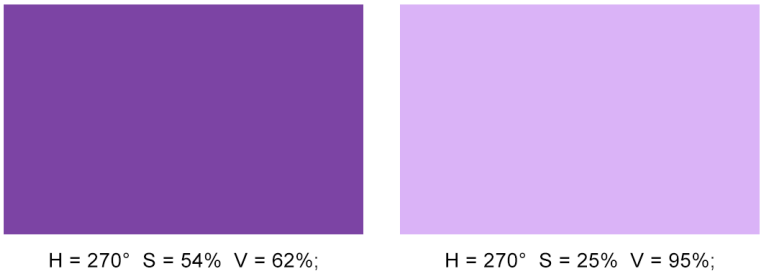


Figura 4.16: Colore e relativa tonalità pastello.

È stato necessario gestire un'eccezione. Se il colore dominante risulta essere visivamente assimilabile al nero o al bianco - ci si riferirà a questa tipologia di colori come *quasi-nero* e *quasi-bianco*- il risultato che si ottiene è discordante con l'immagine di input. Il nero puro è definito da un valore di luminosità pari a 0%. Colori quasi-neri sono dunque caratterizzati da valori bassi di luminosità. Il bianco puro è definito da valori di saturazione pari a 0% e luminosità 100%. Colori quasi-bianchi sono dunque caratterizzati da valori bassi di saturazione e alti di luminosità.

In ogni caso la tonalità - ovvero la componente di colore proprio - risulta visivamente ininfluente. Per un colore pastello, invece, la componente di

[1]Color Thief: `https://lokeshdhakar.com/projects/color-thief/`

tonalità è fondamentale a definirne l'aspetto. Ne consegue che se si schiarisce o si satura, rispettivamente per il quasi-nero e quasi-bianco, il colore per renderlo pastello, sarà il valore di tonalità, che prima non aveva importanti conseguenze visive, a definire il colore risultante.

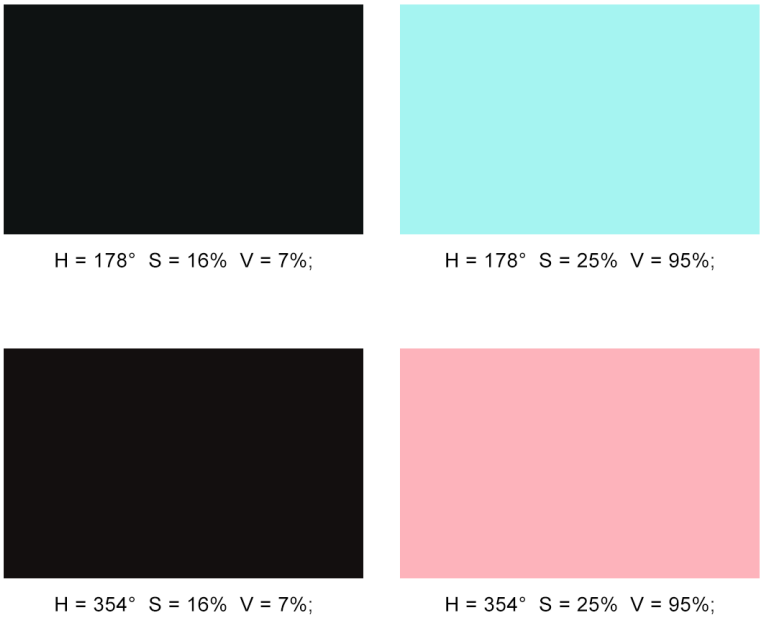


Figura 4.17: Colore e relativa tonalità pastello.

Come si può notare negli esempi sovrastanti, i due colori di partenza sono visivamente molto simili, ed assimilabili al nero. Tuttavia, i risultati che si ottengono tramite le operazioni definite per rendere un colore pastello sono estremamente differenti, poiché sono differenti le tonalità dei colori di partenza. Per mantenere coerenza tra la palette di colori del video e l'immagine in input, è stato necessario gestire come un'eccezione i colori quasi-bianchi e quasi-neri, più genericamente i colori *quasi-grigi*. Un colore quasi-grigio è caratterizzato o da un valore di saturazione particolarmente basso o da un valore di luminosità particolarmente basso. Si è scelto di identificare come valore "basso" un valore al di sotto di una soglia posta pari a 15%. Se il colore dominante dell'immagine risulta essere quasi-grigio secondo questo metodo di discriminazione, viene desaturato completamente. La luminosità viene aumentata in ogni caso. In questo modo, ci si assicura di ottenere un colore nella scala dei grigi, ovvero un colore la cui tonalità risulta ininfluente.

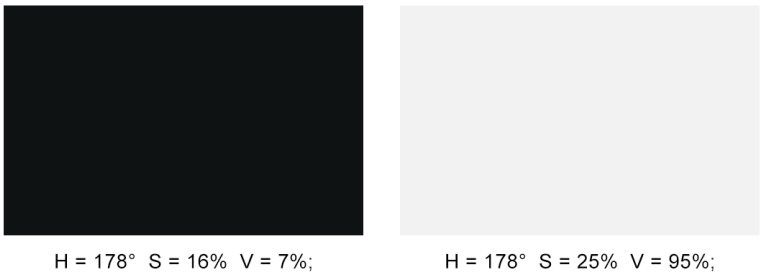


Figura 4.18: Colore associato alla waveform in caso di colori quasi-grigi.

Background
L'assegnazione del colore di sfondo è stata una delle operazioni di customizzazioni più complesse, poiché, pur fortemente dipendente dall'immagine inserita, è stato necessario scegliere un metodo di associazione di colore che mantenesse la visibilità e la coerenza della scena, gestendo opportunamente eventuali casi particolari ed eccezioni.

Il colore del background viene scelto sulla base del colore dominante della scena, e quindi del colore della waveform. Per poter garantire armonia visiva e contrasto necessario per la distinzione tra waveform e sfondo, l'idea alla base della scelta dei colori è stata quella di selezionare, all'interno della palette dell'immagine, quel colore che più si avvicina al complementare del colore dominante. Due colori complementari si definiscono tali se presentano valori "opposti" di tonalità. Il valore di tonalità si visualizza su una ruota cromatica, e viene misurata come l'angolo al centro identificato dal colore. La convenzione pone il rosso a 0-360°. Il colore complementare è dunque quello opposto - che dista 180° - sulla ruota cromatica.

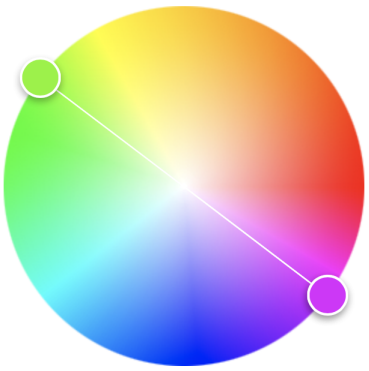


Figura 4.19: Colori complementari sulla ruota cromatica.[2]

La regola generale di assegnazione del colore di background prevede dunque di identificare la tonalità complementare al colore dominante e selezionare, tra i colori della palette dell'immagine, quello con minore distanza cromatica rispetto al complementare.

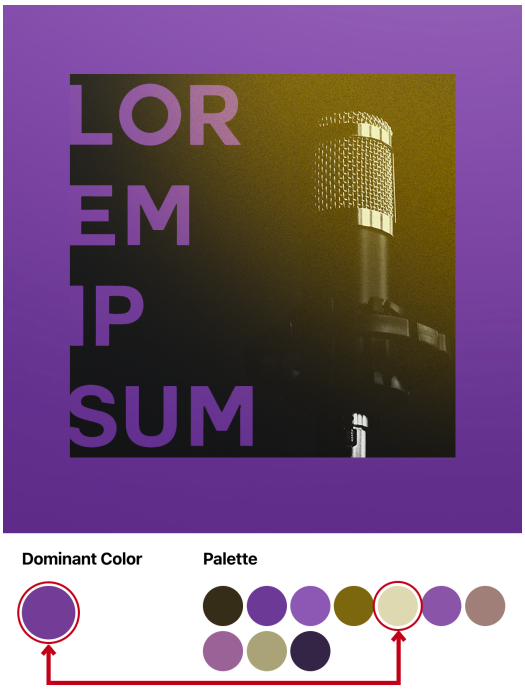


Figura 4.20: Colore dominante e colore secondario associato.[3]

Tuttavia, anche in questo caso è necessario gestire delle eccezioni.

Innanzitutto, un'immagine può avere una palette monocromatica, ovvero può essere composta da colori con tonalità simili. In questo caso il colore con maggiore distanza cromatica risulterà essere sempre troppo simile al colore dominante, e verrebbe a meno la visibilità della waveform. Una volta selezionato dunque il potenziale colore, è necessario controllare la distanza di tonalità tra questo e il colore dominante. Se questa è più piccola di una determinata soglia, il colore del background viene desaturato in modo tale da ottenere un colore neutro. Si è scelta come soglia da superare 60° di distanza.

[2]Immagine creata con Adobe Color: `https://color.adobe.com/it/create/color-wheel`

[3]Immagine creata con Color Thief: `https://lokeshdhakar.com/projects/color-thief/`

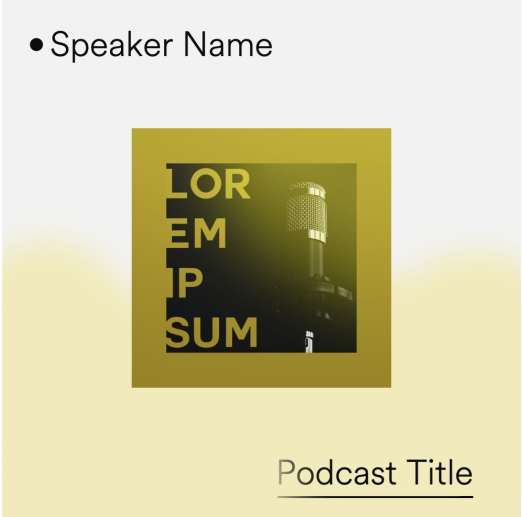
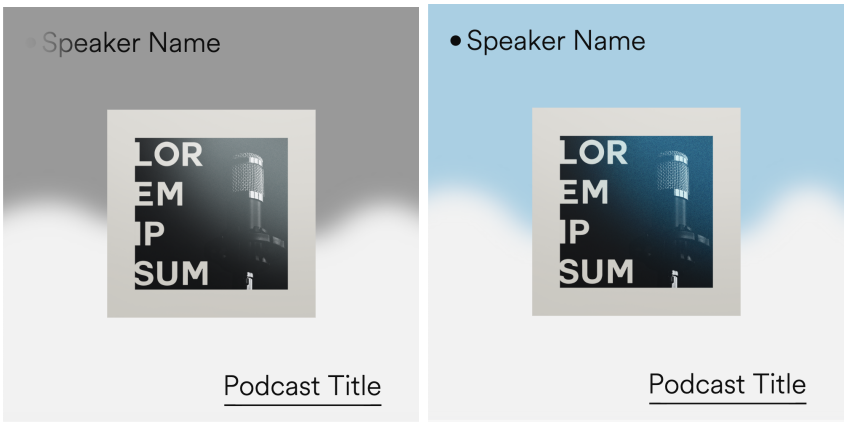


Figura 4.21: Associazione di colori in caso di palette monocromatica.

Il colore di background selezionato deve poi essere reso pastello. In base alla natura del colore di background scelto e del colore di waveform precedentemente calcolato, è necessario gestire delle eccezioni. Il caso di default è che entrambi i colori siano dei semplici generici colori pastello. I casi particolari da gestire sono quelli in cui sono presenti dei colori quasi-grigi. Si ricorda che se il colore dominante è quasi grigio, il colore della waveform risultante è un colore fisso, un grigio molto chiaro, vicino al bianco: si conoscono i valori precisi di saturazione e luminosità, perchè sono stati esplicitamente assegnati. Dunque, in questo caso, vi sono due opzioni:

- Il colore secondario selezionato è quasi-grigio (Fig.4.21 (a)). In questo caso, si desatura completamente il colore e se ne abbassa la luminosità al 60%, in modo tale da ottenere un colore grigio fisso che offra abbastanza contrasto con il colore chiaro della waveform.
- Il colore secondario selezionato non è quasi-grigio (Fig.4.21 (b)). In questo caso, si deve mantenere la tonalità del colore. Tuttavia, pur rendendo il colore pastello, si scelgono valori di saturazione e luminosità leggermente differenti rispetto alla regola di default per assicurare contrasto con il colore chiaro della waveform (si sceglie un valore di saturazione maggiore, e un valore di luminosità minore rispetto ai valori standard utilizzati precedentemente nel rendere un colore pastello).

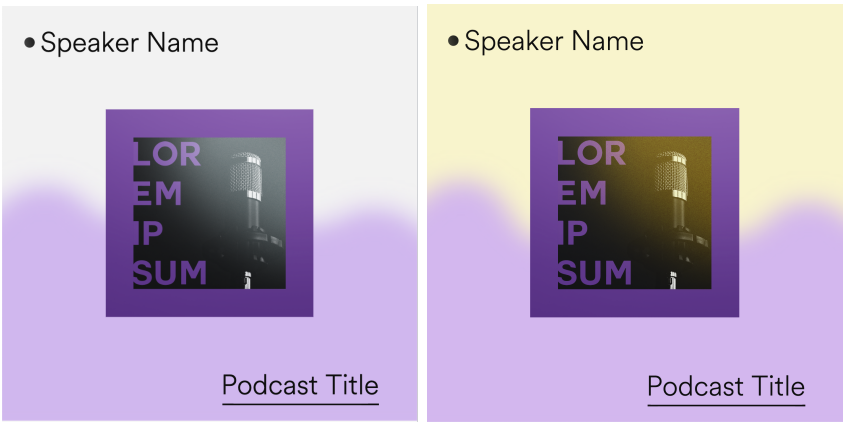


(a) Quasi-grigio. (b) Definito dalla tonalità.

Figura 4.22: Possibilità di associazione del colore di sfondo quando il colore dominante è quasi-grigio.

Se invece il colore della waveform mantiene la sua tonalità vi sono due ulteriori opzioni:

- Il colore secondario è quasi-grigio (Fig.4.22 (a)). In questo caso si desatura completamente e si pone un valore di luminosità alto: si ottiene un colore grigio molto chiaro.
- Il colore secondario non è quasi-grigio (Fig.4.21 (b)). Questo è il caso di default, in cui si rende semplicemente il colore selezionato un colore della stessa tonalità, ma pastello.



(a) Quasi-grigio. (b) Definito dalla tonalità.

Figura 4.23: Possibilità di associazione del colore di sfondo quando il colore dominante è definito dalla sua tonalità.

CTA

Il colore di sfondo della scena di *call to action* finale viene scelto sulla base del colore dominante della scena. A differenza di quanto accade per la waveform, il colore non viene reso pastello. Viene comunque elaborato in modo tale da assicurare la leggibilità del testo, che ha colore bianco.

Se il colore non è quasi-grigio, ed è dunque caratterizzato da una specifica tonalità, se ne aumenta la saturazione fino ad un valore piuttosto alto, e, se necessario, se ne diminuisce la luminosità per evitare che il colore risulti troppo sgargiante. Si impongono dunque due soglie, una per la saturazione, una per la luminosità: se i valori di saturazione sono superiori alla soglia relativa rimangono inalterati, altrimenti aumentano fino a raggiungere la soglia; se i valori di luminosità sono inferiori alla soglia relativa, posta piuttosto alta, rimangono inalterati, altrimenti diminuiscono fino a raggiungere la soglia.

Figura 4.24: Colore associato alla scena CTA se il colore dominante è definito dalla sua tonalità.

Se invece il colore è quasi-grigio, si lascia la saturazione inalterata, ma si porta la luminosità ad un valore piuttosto basso, così da ottenere un colore sufficientemente scuro per garantire la leggibilità del testo.

Figura 4.25: Colore associato alla scena CTA se il colore dominante è quasi-grigio.

Una volta ottenuti dunque i colori corretti da assegnare ai diversi elementi della scena, la customizzazione avviene tramite i comandi della libreria. Tutti e tre gli elementi sono degli shape layer. Si accede dunque alla shape relativa di ogni livello, e alla corrispondente proprietà di fill. Si imposta il colore del fill tramite il metodo di KeyframeProperty *setValue(value)*, dove value è la stringa HEX che rappresenta il colore.

```
let waveformColor = comp.layer('Waveform').shape(0).fill.
    color
//wColor is the rgb color of the waveform.
waveformColor.setValue(rgbToHex(wColor))
```

Outro

L'outro consiste nell'invertire l'animazione di intro, per riportare gli elementi alla stessa disposizione dell'inizio del video. L'animazione dei diversi elementi è stata già definita in fase di progetto. Poichè l'outro ha inizio quando termina la traccia audio inserita, è necessario in questa fase spostare i keyframe di outro opportunamente. Per le strutture di testo, l'immagine e la maschera dell'immagine è dunque sufficiente applicare il metodo *setKeyTime(index, time)* alla proprietà di interesse, assicurandosi di spostarli mantenendo il loro ordine. Diversa è la gestione della waveform: bisogna assicurarsi che, in qualunque modo termini la traccia audio, la waveform venga riportata allo stato di quiete. Nell'animazione di outro, infatti, la traccia audio non è più

in play. Si aggiunge quindi un unico keyframe con valore pari a zero sullo slider control che anima la waveform.

```
    //<-- stop waveform gradually -->
    let easeWaveform = new Ease(80,0)
    let k1 = new Keyframe(startOutro+5/25, 0, easeWaveform)
    wSlider.addKey(k1)

   //<-- speaker animation -->
    let sp = comp.layer('Name Surname')
    if (sp.transform('y position').keyframes.length > 0){
      if (startOutro+sOffset+1 > sp.transform('y position').
          keyframes[2].t) {
      sp.transform('y position').setKeyTime(3,startOutro+
          sOffset+1)
      sp.transform('y position').setKeyTime(2, startOutro+
          sOffset)
      }
      else {
        sp.transform('y position').setKeyTime(2, startOutro+
            sOffset)
        sp.transform('y position').setKeyTime(3,startOutro+
            sOffset+1)
      }
    }

    //...
```

CTA

La comparsa della scena di call to action, così come avviene per l'outro, è variabile e dipende dalla lunghezza della traccia audio inserita. Poichè gli elementi e le animazioni della call to action sono stati definiti in una precomposition, è sufficiente semplicemente spostare il layer della precomp sulla timeline, posizionandolo dopo la fine dell'animazione di outro. Per compiere questa operazione si cambia l'in-point, l'out-point e lo start time del layer della CTA secondo i metodi forniti dalla libreria di customizzazione.

```
    var cta = comp.layer('CTA')
    cta.setStartTime(startOutro + outroLength)
    cta.setInPoint(startOutro + outroLength)
    cta.setOutPoint(startOutro + outroLength + ctaLength)
```

Audio

Su After Effects sono stati definiti opportuni suoni standard che accompagnano l'animazione di intro, outro e CTA. Mentre l'audio di intro ha posizione temporale fissa, l'audio di outro dipende dal placement temporale della relativa animazione (che a sua volta dipende dalla lunghezza della traccia audio caricata dall'utente). L'audio layer interessato viene dunque spostato sul momento opportuno tramite i metodi di customizzazione di in-point, out-point e start time. L'audio di CTA è invece fisso rispetto alla precomposition, dunque con lo spostamento temporale del layer relativo alla CTA risulta già corretto.

Infine si inserisce la traccia audio scelta dall'utente. Nel Lottie era stato istanziato un audio layer con una sorgente placeholder e durata di un frame. L'inserimento della traccia audio prevede dunque che si sostituisca la sorgente del livello tramite il metodo *setSource(source, mode)* offerto dalla libreria di customizzazione, e che si adatti la durata del livello alla durata della sorgente (impostando opportunamente l'out-point del livello).

Lottie non presenta un supporto nativo per la riproduzione degli audio layer, ma deve necessariamente appoggiarsi ad un player audio esterno. È stata dunque utilizzata la libreria howler.js[4], tramite la quale vengono automaticamente riprodotti i suoni dell'animazione opportunamente personalizzati.

Composition

La durata della composition dipende dalla durata della traccia audio inserita, ed è dunque variabile. Viene definita con il metodo relativo *setDuration(value)* definito nella libreria, sommando la durata dei quattro macromomenti di animazione: intro, waveform, outro e CTA.

```
comp.setDuration(introLength + audioData.length/comp.
    framerate + outroLength + ctaLength)
```

4.2 Progetto Transcript

Il secondo progetto realizzato è stato il progetto Transcript. Questa demo permette di caricare un file audio ed ottenere un video con un testo animato. Il testo è la trascrizione di ciò che viene pronunciato nella traccia audio.

Il progetto Transcript prende in input gli stessi campi del progetto Waveform. Benché l'animazione del testo sia la feature principale del progetto,

[4]howler.js: `https://howlerjs.com`

si mantiene - in un ruolo secondario, come sfondo - l'animazione di una waveform sulla base della traccia audio inserita.

Il video può essere suddiviso in tre macro-momenti di animazione differenti:

- Intro, l'animazione di comparsa dei titoli.
- Transcript, l'animazione del testo ottenuto dalla traccia audio (e della waveform).
- CTA, la comparsa della call to action finale, ovvero del link al podcast.

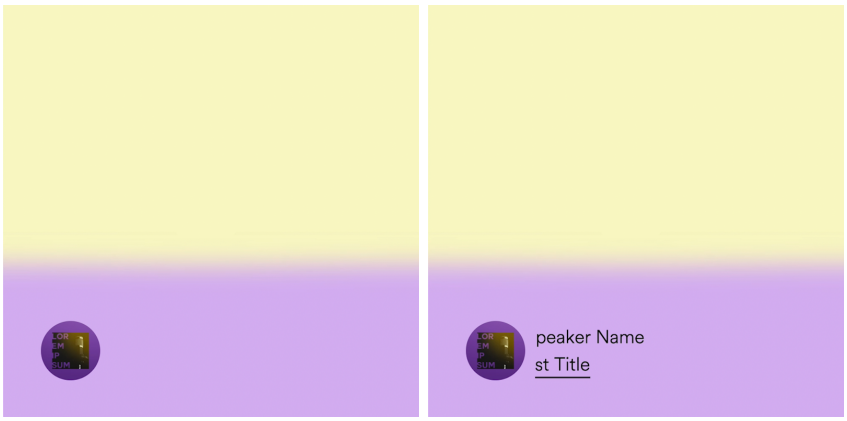


Figura 4.26: Intro

Figura 4.27: Transcript

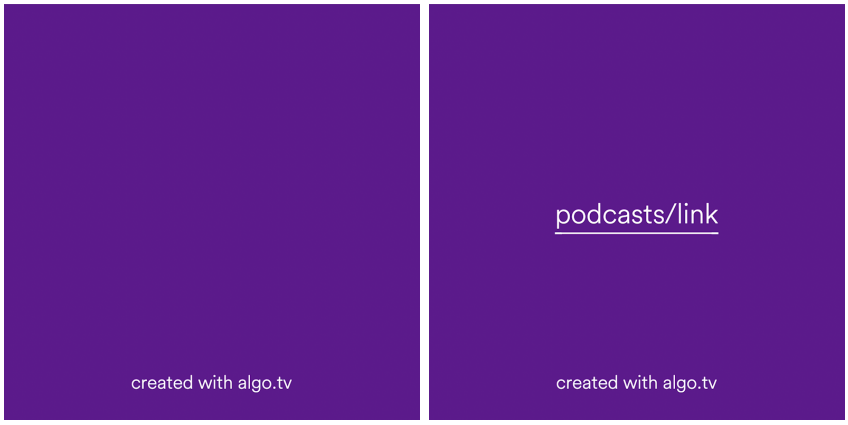


Figura 4.28: CTA

Il progetto dunque permette di ottenere - con gli stessi dati di input - un'alternativa alla preview Waveform. Per questa ragione, condivide con essa molti elementi e caratteristiche. Si analizzano dunque in modo approfondito le feature nuove e le differenze rispetto al progetto descritto precedentemente.

4.2.1 Sviluppo: After Effects

Si descrive in questo paragrafo la realizzazione del template video del progetto su After Effects. Non verranno descritte tutte le features del video, ma solo quelle che soggette a customizzazione.

Transcript

L'elemento di transcript viene realizzato attraverso un text layer. Su di questo è definito un animator, con un selector di tipo range, che controlla le proprietà di opacità e posizione. È questo animator che verrà opportunamente customizzato per definire l'animazione del testo. Il range selector viene impostato con le proprietà di default, fatta eccezione per i campi di *units* - impostato su *index* - e *based on* - impostato su *word*. I valori di start, end ed offset del selector si riferiscono dunque all'indice della parola all'interno della stringa di testo del layer.

Waveform

È stata realizzata con la stessa struttura del progetto Waveform. La forma d'onda del progetto Waveform risultava parzialmente coperta dall'immagine di copertina del podcast, per cui è stato inserito un numero di punti piuttosto alto, in modo tale da assicurarsi che fosse visibile e tale da percepirne

il movimento nonostante la porzione centrale nascosta. La forma d'onda del progetto Transcript è invece interamente visibile, quindi è stata costruita con un numero di punti minore. Inoltre, in questo caso la waveform è un'animazione di background, non è più il focus del video. Per non distrarre dunque dalla feature d'interesse principale - l'animazione del testo - si è applicato sull'array di campioni un filtro di smoothing più ampio ed è stata definita la forma d'onda con un offset di movimento più contenuto.

Testo - Speaker

Il nome dello speaker viene visualizzato attraverso un semplice text layer. Si imposta come track matte layer del testo uno shape layer rettangolare, più o meno corrispondente alla zona nella quale è stata inserita l'immagine del video. Viene poi definita un'animazione di intro sulla X Position del testo: il text layer viene mascherato dallo shape layer in modo tale da dare l'idea che il testo compaia da dietro l'immagine.

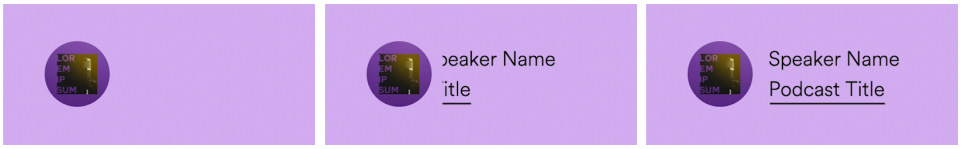


Figura 4.29: Animazione di comparsa dei testi nella scena di intro.

Testo - Titolo del Podcast

Il titolo del podcast è una struttura formata da un text layer, il testo, e uno shape layer, l'underline del testo. La sottolineatura del testo viene creata attraverso lo stesso metodo visto precedentemente nel progetto Waveform, ovvero con l'associazione dei punti della linea a specifici null layer che andranno opportunamente customizzati per adattarsi alla lunghezza del testo. È definita la stessa animazione di intro sulla X Position definita per il text layer dello speaker. Poiché la struttura deve essere mascherata da un track matte layer - lo stesso shape layer che maschera il testo dello speaker, duplicato - la linea viene realizzata attraverso uno shape layer rettangolare - e non tramite uno stroke.

Sia per il testo dello speaker, sia per il testo del titolo sono assenti le animazioni di scomparsa graduale presenti nel progetto Waveform. Non è necessario, infatti, riempire o animare ulteriormente la scena: l'animazione del testo e della waveform sono sufficienti.

Immagine

L'immagine è costituita da un image layer ed uno shape layer circolare utilizzato come maschera. Non è definita alcuna animazione di comparsa.

CTA

La struttura della CTA riprende esattamente quella vista per il progetto Waveform.

Audio

Sono stati inseriti degli effetti sonori standard, per accompagnare le animazioni di intro e CTA ed un audio layer con una sorgente placeholder che in fase di customizzazione conterrà la traccia audio caricata dall'utente.

4.2.2 Sviluppo: Customizzazione via JavaScript

Si descrive in questo paragrafo la gestione della personalizzazione del video, realizzata tramite uno script JavaScript e con il supporto della libreria di customizzazione.

Transcript

L'animazione del testo viene interamente definita con le funzioni di customizzazione. Innanzitutto, è necessario ottenere il testo che deve essere mostrato ed animato, ovvero la trascrizione della traccia audio inserita dall'utente. Questo è un processo che avviene in uno step precedente alla customizzazione, in fase di raccolta ed elaborazione dei dati. Per ottenere i dati testuali di una traccia audio è stata utilizzata Watson Speech To Text[5], un'API di IBM che converte una traccia audio in testo tramite operazioni di riconoscimento vocale. Si richiede dunque a questo servizio la trascrizione dell'audio caricato dall'utente. I dati ottenuti vengono opportunamente rielaborati per agevolare la customizzazione: vengono organizzati in un array di "parole", dove ogni parola è un oggetto che contiene la stringa di testo corrispondente, il tempo di inizio e il tempo di fine enunciazione della parola (espressi in secondi, relativamente alla traccia audio di riferimento). Sono questi i dati necessari per definire, infatti, l'animazione richiesta: le parole del testo devono comparire una alla volta in sincrono con l'audio, in modo tale che la loro comparsa coincida con la loro enunciazione.

[5]IBM Watson Speech To Text: `https://www.ibm.com/cloud/watson-speech-to-text`

L'animazione di comparsa della parola agisce su due proprietà: l'opacità e la posizione. Questa richiede che ogni parola diventi visibile cambiandone il valore di opacità - da opacità nulla a massima - e compiendo un movimento verticale verso l'alto, per allinearsi al resto della frase già visualizzata ed enunciata.

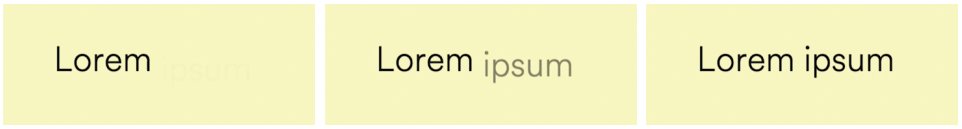


Figura 4.30: Animazione di comparsa di una parola del transcript.

Per ottenere questa animazione si utilizza l'animator definito in fase di progetto. Le proprietà dell'animator sono state precedentemente impostate con il valore iniziale che deve avere ogni parola: opacità minima e posizione riabbassata rispetto a quella che sarà la baseline del testo. L'animazione di comparsa della parola dipende dai parametri di start, end, ed offset del range selector dell'animator. In fase di progetto After Effects, è stato posto il parametro di start pari a zero, e di end pari ad un valore estremamente alto, in modo tale che sia sempre superiore al numero di parole all'interno della frase da visualizzare. Il parametro di offset indica lo spostamento da attuare sul range selezionato dai valori di start ed end. Se l'offset è pari a zero, significa che la selezione comprende tutte le parole tra gli indici di start ed end, ovvero tutte le parole della frase sono ad opacità nulla e in posizione riabbassata rispetto alla baseline del testo: non sono visibili. Se l'offset è pari ad end, nessuna delle parole rientra nel range di selezione, dunque tutte le parole sono ad opacità massima e posizionate sulla baseline del testo.

L'animazione di comparsa si crea dunque animando opportunamente l'offset: si aggiunge un keyframe alla proprietà per ogni parola all'interno della frase da visualizzare. Questo keyframe ha come placement temporale il momento in cui la parola termina di essere pronunciata (valore indicato nell'array di parole ottenuto dai dati, a cui ci si può riferire come *end time* della parola), e come valore l'indice di posizione della parola nella frase visualizzata. I keyframe hanno ease in e ease out specifici, in modo tale da creare un'animazione graduale: l'animazione è rapida in partenza e si addolcisce all'arrivo.

Il testo si deve adattare disponendosi su più linee. Si definisce dunque la larghezza massima del testo e si impone come modalità di adattamento del testo la modalità Fixed Font Size. In questo modo, il font size rimane quello definito in fase di progetto su After Effects; quando la larghezza del testo

supera il limite imposto, il testo si dispone su una nuova riga. Queste due operazioni si compiono chiamando i metodi relativi della libreria.

È necessario gestire esplicitamente l'evoluzione verticale del testo e l'aggiornamento della frase da visualizzare nel tempo. La libreria permette di poter adattare il testo sulla base di un bounding box aggiungendo, oltre al limite di larghezza già impostato, anche un limite di altezza. Tuttavia, non è questa la soluzione richiesta in questo caso. Infatti, secondo questo metodo di adattabilità, il font size diminuirebbe per far rientrare l'intero testo all'interno del bounding box definito. La richiesta di design è invece differente: quando l'altezza del testo supera un certo limite - graficamente la distinzione tra waveform e background - si aggiorna la frase visualizzata a schermo. Il testo globale viene dunque teoricamente suddiviso in una serie di blocchi, visualizzati uno per volta. Ogni blocco corrisponde ad una frase da visualizzare, ed all'interno del blocco avviene l'animazione parola-per-parola definita precedentemente. Quando il blocco di testo raggiunge il limite di altezza massimo, si mostra il blocco di testo successivo. Si aggiorna opportunamente anche il dato di offset, che dipende dall' indice della parola nella frase da visualizzare a schermo, non dall'indice della parola nell'array globale del testo.

(a) Blocco 1. (b) Blocco 2.

Figura 4.31: Blocchi di testo.

La gestione globale del testo avviene dunque nel seguente modo.

Si impone un limite di estensione orizzontale per il testo e si seleziona come modalità di adattabilità Fixed Font Size, attraverso i metodi specifici della libreria.

Si considera l'array di parole. Per ogni parola si aggiunge un keyframe, sulla proprietà di Source Text del layer, che ha come placement temporale

l'end time della parola (indicato nell'array), e come valore la stringa composta da tutte le parole comprese tra l'inizio della frase da visualizzare e la parola corrente. All'inizio dell'elaborazione, l'inizio della frase coincide con la prima parola dell'array.

Si definisce anche l'animazione di comparsa della parola, aggiungendo un keyframe alla proprietà di offset del range selector dell'animator definito nel text layer. Ogni keyframe ha come placement temporale l'end time della parola e come valore la posizione della parola nella stringa composta dalla frase visualizzata.

Ogni volta che si aggiunge una parola alla frase visualizzata - ovvero ogni volta che si aggiunge un keyframe alla proprietà di Source Text - si controlla l'altezza raggiunta dal testo (il testo viene automaticamente disposto su più linee quando necessario): se questa supera una determinata soglia è necessario aggiornare la frase, ovvero visualizzare il blocco di parole successivo. Se, nell'array di parole globale, la frase visualizzata raggiunge il termine massimo di altezza con l'inserimento della parola di indice n, aggiornare la pagina significa che la frase visualizzata successivamente diventa l'insieme di parole a partire dalla parola con indice $n + 1$ (fino alla parola che determina l'altezza massima del blocco). Non è necessario aggiornare solamente la stringa visualizzata nei keyframe definiti nella Source Text, ma anche l'offset. La proprietà di offset infatti considera la posizione della parola nella frase: al refresh della pagina, sull'offset viene aggiunto un keyframe di valore pari a 1.

Testi - Speaker e Titolo del Podcast

Sia il testo relativo allo speaker, sia il testo relativo al titolo del podcast si dispongono su una sola linea, dunque se ne definisce la responsiveness impostando per i due text later un limite orizzontale di estensione e mantenendo la modalità Responsive Font Size.

Sul text layer dello speaker è definita un'animazione di comparsa, che deve essere opportunamente gestita e adattata. L'animazione di comparsa agisce sulla posizione orizzontale. Il valore iniziale della posizione - ovvero il valore del primo keyframe definito - deve essere tale da assicurare che il testo sia completamente coperto dal suo track matte layer, affinché questo non sia visibile. Si cambia dunque il valore del primo keyframe in base alla lunghezza del testo.

La stessa animazione di comparsa è definita sul text layer del titolo: si adatta con lo stesso metodo descritto. È necessario anche adattare la lunghezza della linea di underline alla lunghezza del testo del titolo. Questo avviene tramite lo stesso metodo applicato nel progetto Waveform, ovvero riposizionando, in base alla lunghezza del testo, i null layer che definiscono

la forma della linea. In questo caso, poichè il testo è allineato a sinistra, i null layer di sinistra rimangono fissi all'inizio del testo, mentre i null layer di destra si spostano - rispetto a quelli di sinistra - di un offset pari alla lunghezza della stringa di testo.

La customizzazione dei colori, dell'audio, del movimento waveform, della CTA, e della durata della composition è stata realizzata secondo gli stessi metodi utilizzati nel progetto Waveform.

4.3 Feedback sull'uso della libreria di customizzazione.

La libreria di customizzazione è stata un importante elemento di aiuto per la realizzazione dei progetti demo. Delegando ad essa l'accesso e la modifica delle informazioni nel Lottie-JSON, la difficoltà nella personalizzazione del video è stata riscontrata semplicemente nella logica di customizzazione che - ovviamente - varia di progetto in progetto. Non è stato necessario fare controlli per verificare che la modifica di un valore del JSON non recasse problemi altrove e non è stato necessario gestire esplicitamente operazioni di adattamento della customizzazione che normalmente sono incredibilmente dispendiose in termini di tempo (come nel caso dei testi), poiché queste operazioni sono già state incluse nei metodi della libreria. Alcune delle operazioni necessarie per il funzionamento corretto dell'animazione dei video sarebbero state - se non impossibili - piuttosto macchinose, come nel caso di aggiunta e modifica dei keyframe, oggetti che nel JSON sono dipendenti tra di loro. Parte della praticità dell'uso della libreria risiedeva nel fatto che, nel momento in cui si avesse bisogno di una particolare operazione non inclusa tra i metodi a disposizione, è stato possibile aggiungerla - sia perché la libreria e i progetti sono stati sviluppati nello stesso contesto del tirocinio e in finestre temporali adiacenti, sia perché la libreria è stata pensata per essere scalabile ed espandibile. Rimangono alcune difficoltà derivanti da Lottie che la libreria non può gestire. È quindi, in ogni caso, fondamentale testare il template realizzato su After Effects per verificare che venga renderizzato correttamente da Lottie, e, nel caso, pensare a metodi di animazione con esso compatibili. Per esempio, nel caso dei progetti demo, particolare attenzione è stata riservata al corretto render dei gaussian blur e al corretto funzionamento dei track matte layer, entrambi elementi supportati da Lottie che richiedono tuttavia una configurazione e un'attenzione particolare.

Capitolo 5

Conclusioni

Nel presente lavoro di tesi sono stati analizzati il ruolo e le caratteristiche della creazione automatizzata di video in motion graphics personalizzati sulla base dei dati. In particolar modo, il trattato si è concentrato sulla produzione di video automatizzati visualizzati come web animation tramite l'utilizzo di Lottie, e sulla ricerca di una metodologia di ottimizzazione della customizzazione di tali video. La tesi è stata resa possibile grazie all'esperienza di tirocinio in Algo, che ha permesso in primis di poter osservare da vicino differenti progetti di video automation ed analizzare le loro diverse declinazioni, per quanto riguarda requisiti tecnici, scopi e metodologie di personalizzazione. Il tirocinio inoltre è stata sede di gran parte del lavoro presentato in questa tesi, poiché in questo ambito si è svolto lo sviluppo della libreria di customizzazione per animazioni Lottie ed il suo utilizzo in due progetti di video automation.

Prima di affrontare nello specifico la sezione operativa della tesi, è stato necessario comprendere l'ambiente e il contesto nel quale si colloca l'oggetto del lavoro. Per questa ragione è stato presentato come primo argomento della tesi il ruolo e la presenza sempre più dilagante del video come mezzo comunicativo nel panorama digitale. Ci si è concentrati poi, in particolare, sul video in motion graphics, analizzandone gli ambiti di utilizzo ed i principi di realizzazione derivanti dal mondo dell'animazione e dal mondo del graphic design, in modo tale da comprendere dove è possibile impiegare questa tipologia di video e come renderlo efficace. Dopodiché si è affrontata la definizione della video automation, le tipologie di video possibili sulla base della natura dei dati a disposizione, ed i suoi campi di applicazione, individuando un utilizzo particolarmente presente nel campo della divulgazione, della visualizzazione dei dati, e del marketing. I video utilizzati in campagne con questi scopi sono infatti spesso costruiti sulla base di dati - i quali rappresentano le caratteristiche di un prodotto, l'andamento di un fenomeno, o

le abitudini di consumo di un cliente - e pertanto beneficiano di un processo di produzione automatizzato. Tramite il tirocinio in Algo è stato possibile osservare diversi esempi di campagne di video automation e studiare - ed in questa sede esporre - alcuni processi di produzione dedicati. In particolar modo sono state analizzate due metodologie di produzione, la prima basata su operazioni di scripting su After Effects, la seconda sull'utilizzo di Lottie. La loro analisi ha portato ad individuarne i vantaggi, gli svantaggi e le differenze, di fondamentale importanza poiché hanno reso chiaro quali fossero le tipologie di progetto in cui Lottie - il focus della tesi - risulta essere preferibile, i suoi punti di forza e, soprattutto, eventuali carenze a cui porre rimedio con la ricerca svolta. Il resto del trattato si è focalizzato infatti su Lottie, il processo di produzione su di esso basato, ed il lavoro di ricerca e sviluppo portato avanti per ottimizzarlo.

È stata descritta la struttura base di un Lottie-JSON, analizzando il modo in cui le principali features di una generica composition in After Effects vengono codificate nell'oggetto, in modo tale anche da sottolinearne le gerarchie, le relazioni tra i diversi campi e la complessità della struttura. È stata resa dunque evidente la necessità di un metodo di accesso e modifica alle informazioni dell'oggetto più organico ed efficiente di quello manuale. Questo metodo è stato messo in pratica ed attuato tramite la libreria di customizzazione realizzata in sede di tirocinio, di cui sono state descritte la logica di realizzazione, la struttura, ed i metodi di personalizzazione messi a disposizione.

Infine, è stato spiegato il processo di produzione dei due progetti di video automation realizzati in sede di tirocinio. Questi vengono analizzati dalla realizzazione su After Effects fino alla customizzazione via JavaScript per mezzo della libreria. Questa analisi permette di comprendere come la realizzazione di un'animazione soggetta a personalizzazione ed un'animazione classica siano differenti anche in fase di creazione su After Effects: è necessario, infatti, già strutturare le informazioni della composition per poter facilitare il processo di customizzazione - per esempio servendosi di expression ed expression controls - e sempre tenere conto dei limiti intrinseci di Lottie, cercando alternative per le features non supportate. Inoltre, viene analizzato il processo di customizzazione, dal quale si può evincere come - grazie alla libreria sviluppata - sia possibile attuare le operazioni di personalizzazione in maniera più vasta, rapida ed efficace rispetto alla manipolazione manuale di un oggetto JSON.

L'utilizzo di Lottie non viene sicuramente reso privo di inconvenienti grazie al semplice uso della libreria. Rimangono features non supportate da Lottie, così come features e operazioni di personalizzazione non (ancora) disponibili tramite la libreria: questa supporta finora le operazioni più comuni e

maggiormente utilizzate. La libreria è stata tuttavia costruita con l'ottica di poter essere espansa per aggiungere la personalizzazione di più proprietà, raccogliendo eventuali future necessità personali e responsi da parte dell'azienda per la quale è stata creata.

Ringraziamenti

Questa tesi è stato il frutto del lavoro e dello studio svolto in diversi mesi. Ringrazio il professor Malnati, non solo per aver accettato di seguirmi in questo percorso, ma anche - e soprattutto - per gli insegnamenti che mi ha donato in questi anni di università.

Ringrazio il professor Ruffinengo, che mi ha supportata e sostenuta in tutti questi mesi, sia durante il tirocinio sia durante il percorso di tesi. Grazie per avermi dato questa possibilità e per aver riposto la tua fiducia in me, e soprattutto grazie per le tue lezioni che mi hanno permesso di avvicinarmi ed appassionarmi a questo universo fatto di arte e tecnica.

Ringrazio inoltre il team di Algo, specialmente Luca ed Ilenia, per avermi accolto e per avermi concesso l'opportunità di vivere e condividere il loro piccolo magico mondo.

Ora mi concedo un piccolo momento autoindulgente.

Più di tutti ringrazio la mia famiglia. Mamma, papà e Sergio: siete i miei pilastri. Mi siete sempre stati vicini, per ridere e per scherzare, per offrirmi conforto, per ascoltarmi e per tranquillizzarmi quando l'ansia prendeva il sopravvento, per darmi un abbraccio, per ricordarmi che in ogni caso siete e sarete fieri di me, per farmi capire che sono di più dell'insieme dei miei risultati. Mi avete preso la mano e non me l'avete mai lasciata andare. Vi voglio bene, immensamente.

Ringrazio la mia nonna che anche oltreoceano ha saputo sempre darmi affetto e starmi accanto, ed un pensiero va al mio nonno, che ci ha lasciato tanto tempo fa, ma a cui penso ancora ogni giorno: spero che tu possa essere fiero di me, mi manchi sempre.

Ringrazio poi il mio meraviglioso Alessandro, indubbiamente il più bel dono che questo percorso mi abbia dato, che dal giorno in cui abbiamo deciso di condividere la nostra vita non ha mai lasciato il mio fianco. Affrontare le avversità è più semplice se lo faccio insieme a te.

L'ultimo ringraziamento va infine a me stessa, per essere caduta e per essermi rialzata. Gli ultimi anni di università sono stati indubbiamente i

più difficili e tortuosi vissuti finora, ma gli ostacoli e i fallimenti incontrati nel cammino mi hanno permesso di capire e interiorizzare una verità fondamentale: c'è tanto di più nella vita che la lista degli obiettivi raggiunti. I piccoli momenti, gli attimi di gioia che riusciamo a strappare di qua e di là, il conforto della famiglia, le risate condivise con gli amici, l'arte, la musica e le storie che ci lasciano qualcosa dentro, tutto ciò è molto più importante di una linea sbarrata sulla *bucket list* della vita.

Chi mi conosce non si stupirà della mia capacità di inserire Star Wars in ogni contesto, e questo non poteva fare da eccezione. Quindi concludo questi bizzarri ringraziamenti - e con essi il mio percorso universitario - con una citazione del saggio maestro Yoda: *il più grande maestro, il fallimento è.*

Bibliografia e Sitografia

Software, API e strumenti online

Adobe After Effects. URL: https://www.adobe.com/products/aftereffects.html.
Adobe Color. URL: https://color.adobe.com/it/create/color-wheel.
Advanced Web Rankings - Google SERP features. URL: https://www.advancedwebranking.com/features-spread/.
Bodymovin. URL: https://aescripts.com/bodymovin/.
Color Thief. URL: https://lokeshdhakar.com/projects/color-thief/.
FFMPEG. URL: https://ffmpeg.org.
howler.js. URL: https://howlerjs.com.
IBM Watson Speech To Text. URL: https://www.ibm.com/cloud/watson-speech-to-text.
Lottie Web. URL: https://github.com/airbnb/lottie-web/blob/master/build/player/lottie.js.

Documentazioni ufficiali

Adobe After Effects Scripting Guide. URL: https://ae-scripting.docsforadobe.dev/index.html.
Adobe After Effects User Guide. URL: https://helpx.adobe.com/after-effects/user-guide.html.
Lottie Web Documentation. URL: https://github.com/airbnb/lottie-web.

Siti web di interesse

Algo. URL: https://algo.tv.
Illo. URL: https://illo.tv.
Imaginary Forces. URL: https://imaginaryforces.com.

Lottie. URL: https://airbnb.io/lottie/#/.
Ordinary Folk. URL: https://www.ordinaryfolk.co.

Articoli, pubblicazioni e risorse online

@jrcanest. *Lesson 2 of my @learnsquared Motion Design class is live! Talking about the 10 principles of Motion Design based on Disney's 12 principles.* [Online; ultimo accesso 17/02/2023]. Twitter. 18 Ott. 2016. URL: https://twitter.com/jrcanest/status/788441769864073216.

@netflixqueue. *SOME PERSONAL NEWS: Starting today there's a new logo animation before our originals. It shows the spectrum of stories, languages, fans, & creators that make Netflix beautiful — now on a velvety background to better set the mood. And before you ask: no, the sound isn't changing.* [Online; ultimo accesso 17/02/2023]. Twitter. 1 Feb. 2019. URL: https://twitter.com/netflixqueue/status/1091342921897406465?s=20.

Algo. *Bar graph daily update*. [Online; ultimo accesso 17/02/2023]. Dribbble. URL: https://dribbble.com/shots/14696840-Bar-graph-daily-update.

B, Withrow, Peal G e Abdul-Karim S. *Lottie: Behind the scenes of our new open-source animation tool*. [Online; ultimo accesso 12/02/2023]. Airbnb Design. URL: https://airbnb.design/introducing-lottie/.

Candy. [Online; ultimo accesso 17/02/2023]. Imaginary Forces. URL: https://imaginaryforces.com/project/candy.

Diagramma gerarchico dei principali oggetti di scripting After Effects. [Online; ultimo accesso 12/02/2023]. URL: https://ae-scripting.docsforadobe.dev/introduction/objectmodel.html.

Everything You Need To Know About 2022 Wrapped. [Online; ultimo accesso 20/02/2023]. Spotify - For the Record. 30 Nov. 2022. URL: https://newsroom.spotify.com/2022-11-30/everything-you-need-to-know-about-2022-wrapped/.

Ferreira, André. *A Guide To Motion Design*. [Online; ultimo accesso 17/02/2023]. LottieFiles. 26 Ago. 2022. URL: https://lottiefiles.com/blog/guides/guide-to-motion-design.

— *How Motion Design Applies to UI Design*. [Online; ultimo accesso 17/02/2023]. Infinum. 12 Ago. 2020. URL: https://infinum.com/blog/motion-design-ui-design/.

Get to Know Your Music Listening Personality from 2022 Wrapped. [Online; ultimo accesso 20/02/2023]. Spotify - For the Record. 30 Nov. 2022. URL:

https://newsroom.spotify.com/2022-11-30/get-to-know-your-music-listening-personality-from-2022-wrapped/.

Global Media Social Statistics. [Online; ultimo accesso 13/02/2023]. Datareportal. 2023. URL: https://datareportal.com/social-media-users.

Illo. *Character design for newsletter*. [Online; ultimo accesso 17/02/2023]. Dribbble. URL: https://dribbble.com/shots/18021763-Character-design-for-newsletter.

— *Dynamic loop with flashy colors*. [Online; ultimo accesso 17/02/2023]. Dribbble. URL: https://dribbble.com/shots/15177966-Dynamic-loop-with-flashy-colors.

Johns Hopkins University - Daily Covid Tracker. [Online; ultimo accesso 25/02/2023]. Algo. URL: https://algo.tv/jhu-covid-tracker.

Learn More About the Audio Aura in Your Spotify 2021 Wrapped With Aura Reader Mystic Michaela. [Online; ultimo accesso 20/02/2023]. Spotify - For the Record. URL: https://newsroom.spotify.com/2021-12-01/learn-more-about-the-audio-aura-in-your-spotify-2021-wrapped-with-aura-reader-mystic-michaela/.

Lustig, Amy. *Applying Design Principles to Motion Design*. [Online; ultimo accesso 17/02/2023]. Fable. 27 Giu. 2022. URL: https://www.fable.app/blog/applying-design-principles-to-motion-design/.

Madrioli, Giulia. *Rebranding Rai, Roberto Bagatti racconta l'evoluzione del design*. [Online; ultimo accesso 17/02/2023]. Digitalic. 25 Set. 2019. URL: https://www.digitalic.it/tech-news/rebranding-rai-roberto-bagatti.

Melvin, Jemma. *Personalized Video: Complete Guide (Benefits, Tips, Examples)*. [Online; ultimo accesso 17/02/2023]. Wyzowl. 10 Feb. 2023. URL: https://www.wyzowl.com/personalized-video/.

Most popular video content type worldwide in 3rd quarter 2022, by weekly usage reach. [Online; ultimo accesso 13/02/2023]. Statista. 2023. URL: https://www.statista.com/statistics/1254810/top-video-content-type-by-global-reach/.

Motion graphics explained: definition, history and examples. [Online; ultimo accesso 17/02/2023]. Adobe. URL: https://www.adobe.com/uk/creativecloud/animation/discover/motion-graphics.html.

New Netflix logo shows importance of motion graphics. [Online; ultimo accesso 17/02/2023]. JMP UK. URL: https://www.jmpuk.com/new-netflix-logo-animation-shows-importance-of-motion-graphics/.

Nisttahuz, Daniel. *Life in Motion: A Guide to Animating Mobile Data Visualizations*. [Online; ultimo accesso 17/02/2023]. TopTotal. URL: https:

//www.toptal.com/designers/data-visualization/mobile-data-visualization.

Philips, Miklos. *How to Use Powerful Gestalt Principles in Design (with Infographic)*. [Online; ultimo accesso 18/02/2023]. TopTotal. URL: https://www.toptal.com/designers/visual/infographic-gestalt-principles-of-design.

Rai Rebrand. [Online; ultimo accesso 17/02/2023]. Eloisa Studio. URL: https://eloisa.studio/rai-rebrand.

Silveira, Felippe. *What is Motion Graphics?* [Online; ultimo accesso 17/02/2023]. Mowe Studio. URL: https://mowe.studio/what-is-motion-graphics/.

Spotify Wrapped: What Makes The Campaign So Successful? [Online; ultimo accesso 20/02/2023]. For The Curious. 13 Dic. 2022. URL: https://www.forthecurious.co.uk/news/spotify-wrapped-what-makes-the-campaign-so-successful/.

Spotify Wrapped: What marketers can learn from the viral campaign. [Online; ultimo accesso 20/02/2023]. SproutSocial. 7 Dic. 2022. URL: https://sproutsocial.com/insights/spotify-wrapped/.

Taino, Zela. *The Audio Aura Story: Mystical to Mathematical*. [Online; ultimo accesso 20/02/2023]. Spotify R&D Engineering. 17 Dic. 2021. URL: https://engineering.atspotify.com/2021/12/the-audio-aura-story-mystical-to-mathematical/.

The art and science of Spotify Wrapped. [Online; ultimo accesso 20/02/2023]. Protocol. URL: https://www.protocol.com/newsletters/sourcecode/spotify-wrapped.

The Key Role of Video Marketing. [Online; ultimo accesso 13/02/2023]. Digital Marketing Institute. 2018. URL: https://digitalmarketinginstitute.com/blog/the-importance-of-video-marketing.

The State of Video Marketing 2023. [Online; ultimo accesso 13/02/2023]. Wyzowl. 2022. URL: https://wyzowl.s3.eu-west-2.amazonaws.com/pdfs/Wyzowl-Video-Survey-2023.pdf.

The Wait Is Over. Your Spotify 2021 Wrapped Is Here. [Online; ultimo accesso 20/02/2023]. Spotify - For the Record. 1 Dic. 2021. URL: https://newsroom.spotify.com/2021-12-01/the-wait-is-over-your-spotify-2021-wrapped-is-here/.

Work with Data-driven animation. [Online; ultimo accesso 17/02/2023]. Adobe. 10 Nov. 2022. URL: https://helpx.adobe.com/after-effects/using/data-driven-animations.html.

Filmografia

Arctic Monkeys. *Arctic Monkeys - Do I Wanna Know? (Official Video)*. [Online; ultimo accesso 17/02/2023]. Youtube. 19 Giu. 2013. URL: https://www.youtube.com/watch?v=bpOSxM0rNPM.

Imaginary Forces. *Candy Main Ttile*. [Online; ultimo accesso 17/02/2023]. Youtube. 9 Mag. 2022. URL: https://www.youtube.com/watch?v=MvU6cJqgCCk.

Movie Titles. *Saul Bass: The Seven Year Itch (1955) title sequence*. [Online; ultimo accesso 17/02/2023]. Youtube. 6 Set. 2017. URL: https://www.youtube.com/watch?v=Gg1MA03VZB4.

Bibliografia

Thomas, Frank, Johnston, Ollie. *The Illusion of Life: Disney Animation*. United States: Abbeville Press, 1982.

Appendice

Si rimanda alla documentazione della libreria per la customizzazione di file Lottie-JSON, consultabile al seguente link: `https://docs.google.com/document/d/1QVlzI7kYgOxyC2FQe1-BJ1u1MjbW7XR1lIHwBkHc5FQ/edit?usp=share_link`