Alma Mater Studiorum

Università degli Studi di Bologna

FACOLTÀ DI INGEGNERIA

Corso di Laurea in Ingegneria Elettronica

Insegnamento: **Geometria e Algebra**

IMPLEMENTAZIONE DI UN SISTEMA DI RESA ACUSTICA DEL COLORE A TAVOLOZZA CONTINUA AI FINI DELLA SPERIMENTAZIONE

Tesi di Laurea di:

Emanuele Cecchetelli

Relatore:

Prof. Massimo Ferri

Correlatore:

Ing. Ludovico Ausiello

SESSIONE AUTUNNALE

ANNO ACCADEMICO 2004-2005

Indice

Introduzione

Le moderne tecnologie elettroniche e informatiche offrono delle possibilità prima impensabili per sostegni a vari tipi di inabilità e in particolare per gli handicap visivi. Da una parte ci sono risultati già consolidati per facilitare l'accesso ai testi: tastiere e stampanti Braille, lettori vocali di testi, ecc. Maggiori problemi rendono più ardua la sfida dell'accesso alle immagini da parte del non vedente. Esempi ne sono il progetto EAV dell'Università delle Canarie e anche il progetto VIDET sviluppato anni fa presso l'Università di Bologna. Nell'ambito di quest'ultimo era già stato affrontato il problema della resa del colore; in tal caso la soluzione che era stata adottata era quella di una trasduzione tattile.

Questa tesi prosegue una ricerca sulla trasduzione acustica del colore ad uso di non vedenti ed eventualmente di daltonici. Tale studio è cominciato nel 2003 presso il DEIS e il DM dell'Università di Bologna con l'implementazione di un programma che forniva una risposta acustica ad una selezione di un colore nell'ambito di un insieme finito. Il presente lavoro estende le potenzialità del sistema ad una selezione praticamente continua di colori rendendo possibile la resa di migliaia di sfumature. Il sistema, sviluppato in Visual Basic 5.0, quindi realizzato per piattaforma Microsoft, è stato dotato di un'interfaccia utente user-friendly e di importanti accessori (memoria cache colori, cronometro, ecc…) concepiti per facilitare e velocizzare il successivo lavoro di sperimentazione.

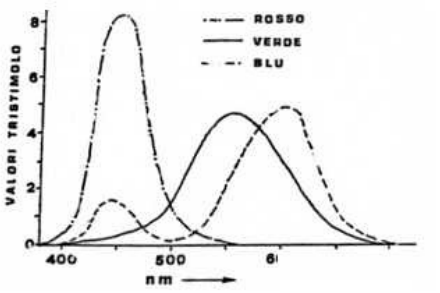
Dopo un primo capitolo, in cui si introducono le nozioni di base sulla colorimetria e si descrivono alcuni particolari dell'udito, viene illustrato il software sviluppato e si motivano le scelte progettuali. Il secondo capitolo fa riferimento a tutti gli aspetti della codifica e alla sua implementazione, mentre nel terzo si descrivono gli strumenti sviluppati e integrati nel programma e gli accorgimenti adottati, al fine di snellire le procedure di sperimentazione. L'elaborato si conclude con un'appendice in cui è esposto tutto il codice con cui è stato realizzato l'applicativo.

Capitolo 1

1.1 Il colore

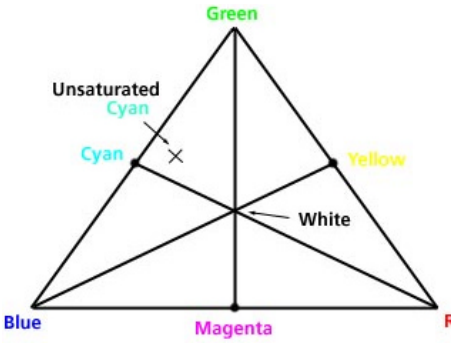
L'uomo mostra una capacità sensoriale nello spettro elettromagnetico caratterizzato da lunghezze d'onda comprese tra 380 nm e 780 nm; in questo range di frequenze si percepiscono i colori come luce riflessa da oggetti illuminati da una sorgente. Un oggetto può essere visto solo se è illuminato; pertanto il suo colore dipende sia dalla sorgente luminosa sia dalla sua capacità e modalità di assorbire la luce. Ad esempio, in un contesto di luce bianca, l'oggetto apparirà bianco se non assorbe luce riflettendola completamente, oppure nero se è capace di assorbire tutta la radiazione luminosa che lo incide. [1].

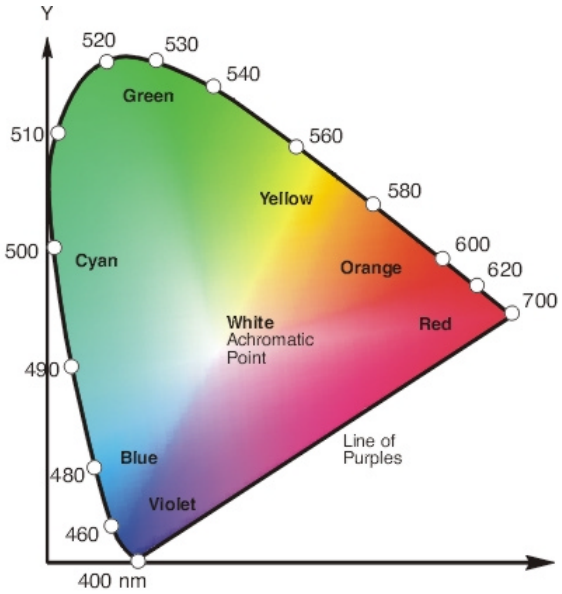
Nella retina esistono tre diversi centri di stimolo il cui massimo di sensibilità spettrale è situato nel rosso, nel verde e nel blu. Possiamo definire i valori X (red), Y (green), Z (blue) come i valori cromatici di base, che indicano l'intensità con la quale i tre centri di stimolo sono sollecitati dalla radiazione incidente. La codifica più semplice per il colore, dunque, ricalca esattamente il funzionamento dell'occhio ed è costituita da tre vettori (tristimolo) mediante i quali possiamo rappresentare ogni singolo elemento del nostro spazio vettoriale; lo spazio dei colori visibili dall'occhio umano è pertanto tridimensionale. Nella figura possiamo osservare le componenti dei tre vettori in funzione dello spettro elettromagnetico [2].



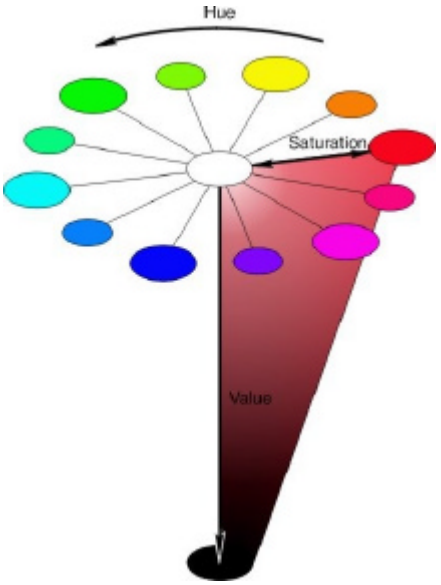
Dai tre colori primari si ottengono tutti i colori dello spettro del visibile. In figura è rappresentato il triangolo dei colori di James Clerk Maxwell. Ai

vertici sono situati i colori fondamentali: rosso, verde e blu. Un gran numero di colori possono essere riprodotti miscelando i tre colori primari.





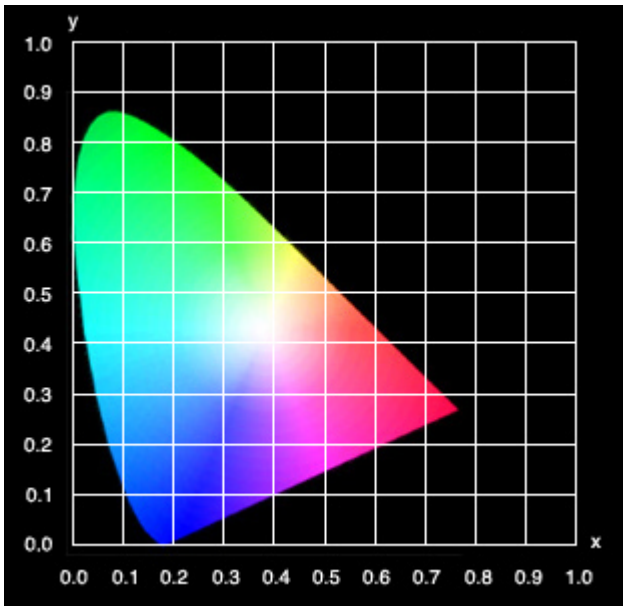
Per descrivere il colore esistono principalmente due sistemi: il primo consiste nel fornire l'intensità delle tre componenti di luce rossa, verde e blu (da cui l'acronimo RGB); il secondo si basa su tre valori che rappresentano la tonalità, la saturazione e la luminosità (HSV) che è un metodo più intuitivo.



La tonalità o tinta (hue) [3] rappresenta non un colore ma una famiglia di colori; tuttavia non tutti i colori hanno una tinta: è il caso dei colori acromatici come bianco, nero e la scala di grigi. La saturazione [4] invece descrive la pienezza di un'area in proporzione alla brillanza del colore dell'area stessa mentre la luminosità o luminanza indica il quantitativo di luce totale.

Qualsiasi colore rappresentato in RGB può essere descritto con il metodo HSV e viceversa, quindi entrambi i metodi necessitano di tre variabili per descrivere lo stesso colore.

Tinta e saturazione insieme generano un particolare spazio: la cromaticità. Questa, insieme alla luminosità forma una nuova base dello spazio dei colori. La varietà delle cromaticità che l'occhio umano medio riesce a percepire si rappresenta con il diagramma delle cromaticità, sviluppato dalla CIE (*Commission Internationale d'Eclairage*) nel 1931 [5] [6] [7].



Al centro del diagramma c'è il bianco e lungo la parte curva del perimetro ci sono i colori saturi dello spettro luminoso: in senso antiorario, rosso, giallo, verde, blu, viola. I colori centrali sono insaturi (il bianco è il più insaturo di tutti). Il diagramma rappresenta quindi le tinte (lungo il perimetro) e le saturazioni (dal perimetro verso il centro).

Ogni colore del diagramma può inoltre avere una diversa luminosità (per esempio un certo verde esiste anche in una versione più scura) e quindi al diagramma va aggiunta una terza dimensione, appunto quella della luminosità. A luminosità zero, tutto il diagramma è nero, a luminosità ridotta il centro non sarà bianco ma grigio, e anche tutti gli altri colori

saranno più scuri. Il diagramma CIE 1931 è a due dimensioni, ma aggiungendo la luminanza diventa tridimensionale, esattamente come le altre rappresentazioni del colore.

1.2 L'udito: capacita' e limiti di elaborazione

L'udito è un senso particolarmente esercitato dai soggetti non vedenti; studi effettuati su piccoli mammiferi (gatti) parzialmente o totalmente privi della vista hanno evidenziato l'attivazione di una particolare zona della corteccia visiva (EAV Cortex) da parte di stimoli sonori [8],[9]. Anche gli altri sensi vengono affinati, come ad esempio il tatto, usato nel sistema VIDET [10] ma è l'udito che ha la maggior capacità di veicolare più informazione in meno tempo. Su questa direzione è stato realizzato il progetto EAV [11], [12], [13]. Al fine di utilizzare l'udito per il riconoscimento dei colori è necessario conoscerne alcuni importanti difetti.

Il nostro apparato uditivo ha una capacità di elaborazione limitata: esiste un fenomeno molto importante e complesso che prende il nome di mascheramento; si può avere mascheramento in frequenza, in intensità ed infine mascheramento temporale [14].

Il mascheramento in frequenza è diretta conseguenza della conformazione fisica dell'orecchio interno e fa sì che un suono, sia esso puro, a banda stretta o a banda larga, percepito in un istante di tempo t_0 alteri la soglia uditiva per un certo tempo t_{Mf} in una banda Δf intorno alle frequenze del primo impulso. Maggiore è la banda del primo impulso, più esteso risulta lo span Δf delle frequenze la cui soglia è aumentata (un impulso di Dirac altera quindi la soglia di udibilità in tutto lo spettro udibile).

Il secondo tipo di mascheramento è quello in intensità: in presenza di due suoni s_1 ed s_2, aventi analoghe caratteristiche spettrali, se la pressione acustica generata da s_1 supera di 10dB quella generata da s_2 allora il nostro apparato uditivo non è in grado di percepire il suono di intensità minore.

Il mascheramento temporale, infine, fa sì che il nostro orecchio non riesca a riconoscere come distinti due suoni se tra di essi non vi è un certo intervallo di tempo minimo t_{Mf}, ma li interpreti invece come un primo suono seguito da un riverbero o da un'eco, a seconda dell'entità del ritardo tra i due.

La conoscenza di questi fenomeni è determinante sia per la scelta dei tre suoni da utilizzare per la codifica sia, in luce di ulteriori sviluppi del sistema, per migliorare la sensibilità di ogni suono col suo rispettivo colore, quindi sugli aspetti riguardanti la dinamica audio della trasformazione.

Capitolo 2

PlayRGBSound

Aspetti Primari:
Dall'immagine al suono

La codifica del colore è il tema centrale del progetto. Data una tavolozza continua, sotto forma di immagine, l'applicativo deve convertire in suono il colore di un qualsiasi suo punto da noi scelto tramite il puntatore del mouse. La continuità della trasformazione consente di cogliere con l'udito anche le minime sfumature di colore.

L'obiettivo viene raggiunto con due stadi di elaborazione, che verranno trattati separatamente: l'estrapolazione del colore di un punto dalla tavolozza, che da un'immagine di input restituisce la terna RGB del pixel puntato, e la trasformazione del colore in suono, che, partendo dalla terna, crea ed esegue il suono univocamente associato.

2.1 Estrapolazione del colore

Ci poniamo il problema di come conoscere le componenti di colore di ogni pixel di una tavolozza continua sotto forma di immagine. Si tratta di ricreare una funzione molto usata nei software di fotoritocco: spostando il puntatore del mouse su di una tavolozza continua, o in alcuni casi, sull'immagine stessa, una funzione analizza il colore e restituisce in tempo reale le componenti RGB del pixel puntato.

Utilizzando opportune API (Application Programming Interface) è possibile individuare il colore di un qualsiasi pixel dello schermo, indipendentemente dall'applicazione di cui fa parte. Nel web esistono in merito vari esempi di codice il cui denominatore comune è appunto l'utilizzo in particolare di due API per la grafica, denominate GetPixel e BitBlt, e uno di questi esempi è stato utilizzato come punto di partenza per la

realizzazione dello strumento. Tra le istruzioni chiave è da menzionare "*C = GetPixel(hD, P.X, P.Y)*" da cui si nota come GetPixel partendo da un identificativo dell'applicazione corrente e le coordinate in essa del puntatore, restituisce il colore alla variabile C sottoforma di numero Long, il quale verrà in seguito utilizzato sia dall'istruzione *"Picture1.BackColor = C "* per colorare il riquadro, sia da un ulteriore blocco di codice dedito all'estrapolazione della terna RGB tramite opportuni calcoli algebrici; i tre numeri vengono poi visualizzati su schermo a fianco del riquadro colorato. In aggiunta, sono state visualizzate sia le coordinate assolute sia quelle relative, del pixel puntato. Il tool è inoltre funzionate anche spostando il mouse su una qualsiasi applicazione in background sullo schermo.

2.2 Dal colore video al colore audio

In questa tesi si propone in primo luogo un software in grado di trasformare lo spazio dei colori in uno spazio, o meglio, sottospazio acustico, in modo lineare e continuo.

La base di colori in generale più utilizzata nell'ambito dei personal computer è sicuramente l'RGB, [15]; una terna di valori da un byte, corrispondente ad interi compresi tra 0 a 255, indica l'intensità del colore corrispondente. Il corrispettivo acustico consiste in tre suoni, opportunamente scelti. Combinando le tre componenti RGB si ottiene lo spazio dei colori, allo stesso modo, effettuando un mixaggio in ampiezza dei tre suoni si genera uno sottospazio acustico. Tale sottospazio è molto limitato se lo si confronta con lo spazio acustico reale, cioè quello che contiene tutti i suoni udibili dall'orecchio umano, infatti non esistono tre suoni che miscelati tra loro possono rappresentare un qualsiasi suono da 20Hz a 20kHz; tuttavia l'obiettivo del progetto è la codifica dei tre colori tramite tre suoni miscelati in ampiezza, quindi questa caratteristica diventa secondaria, mentre di primaria importanza è la scelta di tre suoni molto diversi tra loro, magari con tre spettri diversi, senza sovrapposizioni (quindi ortogonalità), tali da poter essere distinti il più possibile nell'ascolto simultaneo. In questo senso si possono fare molti studi ed esperimenti e ciò induce a realizzare un software in grado di gestire l'audio con flessibilità e potenza, con la possibilità di utilizzare files audio non prefissati ma sostituibili. A tal fine, utilizzando la piattaforma Microsoft, è bene orientarsi sulle librerie DirectX per poter affidare la gestione dell'audio al potente pacchetto DirectSound, compatibile con qualsiasi tipo di file wav.

Tuttavia il formato wav risulta essere quello più idoneo per il progetto: si devono utilizzare suoni di alta qualità e di breve durata, anche in accordo con l'eventuale utilizzo di questa codifica nel sistema EAV, che usa suoni praticamente impulsivi; non esiste quindi alcuna necessità di utilizzare formati compressi come il formato mp3.

2.2.1 La trasformazione lineare

E' il perno della codifica, ovvero quella parte di codice che descrive la relazione tra il volume di un suono e il peso della relativa componente di colore.

Una breve premessa riguardo la dinamica dell'audio: DirectSound, come verrà spiegato in seguito, richiede un numero intero da 0 a –10000 in qualità di parametro inerente al volume di un suono che deve essere eseguito. Tale numero indica i centesimi di dB di attenuazione, per cui raggiungeremo il volume massimo con il valore 0 e volume nullo con –10000 che corrisponde a un'attenuazione di 100dB del suono prodotto.

La trasformazione consiste in tre semplici equazioni lineari identiche, una per componente, che ricevendo un numero intero compreso tra 0 e 255 restituiscono un numero compreso tra 0 e –10000, dove allo 0 in input corrisponde –10000 in output e a 255 deve corrispondere 0. Pensando a una funzione di trasferimento lineare, con questi dati si può già formulare l'equazione, ripetuta per ogni colore.

Rvol = (10000 / 255) * R – 10000
Gvol = (10000 / 255) * R – 10000
Bvol = (10000 / 255) * R – 10000

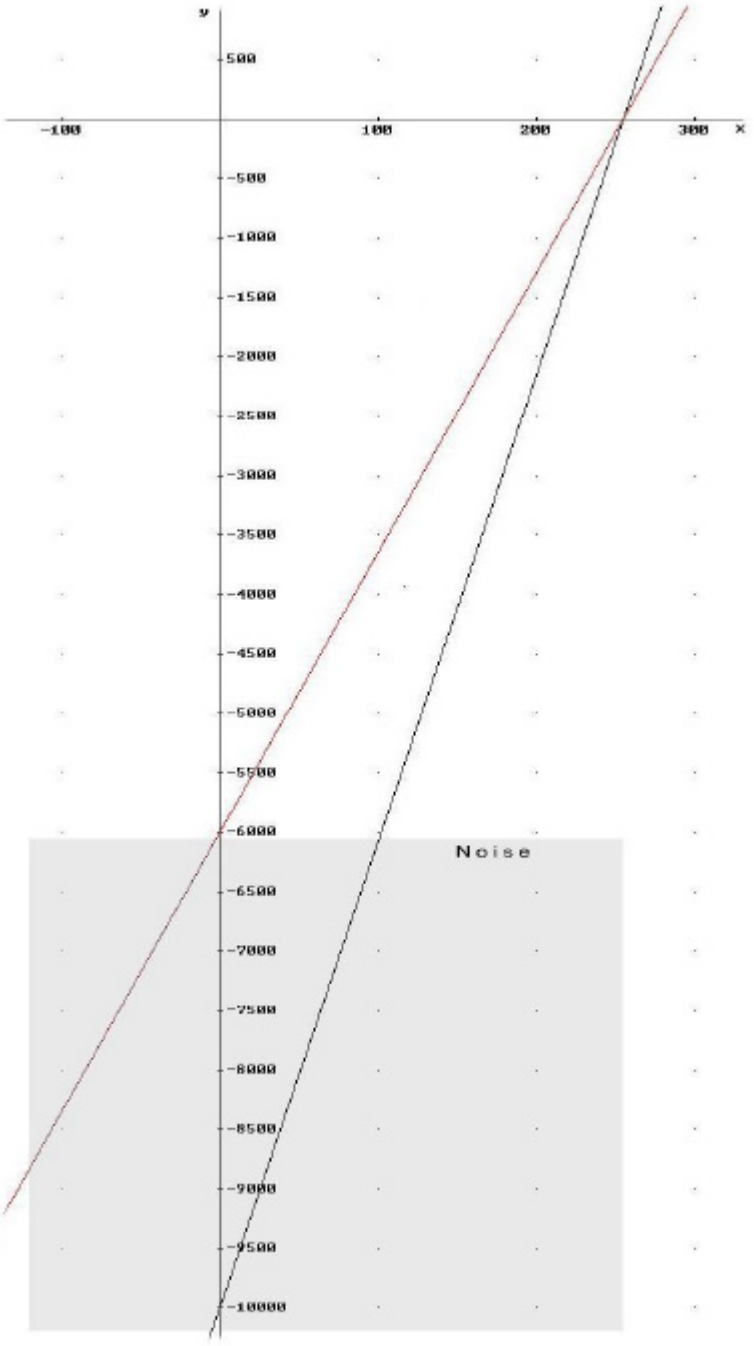
Di fatto è stata la prima equazione che è stata usata nel software.

Sono però ora necessarie opportune considerazioni di carattere tecnico. Supponiamo di provare la codifica con questa equazione su un personal computer dotato una scheda audio di media qualità, il cui rapporto Segnale/Rumore effettivo si attesta intorno ai 70dB. Considerando che il massimo livello sonoro corrisponde a 0dB, si trova che la dinamica "utile" della scheda è di 70dB. Questo dato va in contrasto con la dinamica del DirectSound che è invece pari a 100dB, come descritto sopra, la quale viene quindi inevitabilmente limitata. Ciò significa che quando avviene la codifica con colori scuri, i cui corrispondenti livelli sono compresi tra –100dB e –70dB, noi non avvertiremmo nessun suono, tranne il rumore della scheda, che appunto li sovrasta. Questo problema affligge tutta la codifica perché

abbiamo una distorsione nella trasformazione: nel caso dei colori chiari si nota come la sensibilità del suono ai cambiamenti del colore è piuttosto bassa e quindi nelle sfumature non c'è fedeltà: il suono prodotto non è molto attinente al colore che lo ha generato. L'inconveniente, che non esisterebbe nel caso si in cui si utilizzino interfacce audio di fascia alta o professionale, con le quali si arriva facilmente a rapporti Segnale/Rumore di 105dB, è tuttavia facilmente risolvibile: è sufficiente adattare la dinamica del della trasformazione alla dinamica utile dell'hardware che gestisce l'audio. Ciò si traduce nella modifica della costante relativa alla massima attenuazione sull'equazione di trasferimento: di fatto il volume nullo è associato a –70dB e non a –100dB. Sostituendo questo valore nell'equazione viene ristabilito il matching tra ingresso e uscita; i miglioramenti in fase di verifica sono notevoli, la precisione e la fedeltà nelle sfumature sono molto più affinate. Considerando che l'applicativo deve poter funzionare su personal computer senza particolari caratteristiche, si è proceduto a sensibilizzare ulteriormente la codifica immettendo sull'equazione una costante pari a –6000 che è idonea alla maggior parte delle schede audio, di qualità medio-bassa, presenti sugli odierni personal computer; ciò significa che la codifica usata è ottimizzata per normali schede audio, ma non solo: anche se ci fosse l'hardware capace di sfruttare 100dB di dinamica sarebbe comunque necessario fare prove in un luogo decisamente silenzioso o usare un'adeguata amplificazione, con la quale si rischierebbe però di lavorare con volumi piuttosto alti quando i colori sono molto chiari. Sfruttare una dinamica sopra certi valori diventa quindi oneroso. Tuttavia nel caso di utilizzo con schede audio di fascia superiore è necessario amplificare notevolmente il segnale di uscita per accorgersi che anche il colore 0,0,0 produce un leggero suono, appena sopra il livello del rumore, ma decisamente trascurabile.

Nel grafico che segue sono mostrate le due rette di trasformazione relative sia a 100dB che a 60dB di dinamica.

In riferimento al codice, le equazioni della trasformazione sono contenute nella funzione PlayRGB, in cui compaiono anche le varie istruzioni per la riproduzione che saranno trattate in seguito.



2.2.2 La gestione dell'audio con DirectSound

In riferimento alla piattaforma Microsoft, le librerie DirectX sono attualmente uno standard di fatto sulla gestione ad alte prestazioni di grafica e audio, come ad esempio nel settore dei videogiochi, ma attualmente sono fondamentali anche per la gestione di periferiche

multimediali di vario genere. Si tratta di API che estendono le API a 32 bit di Windows aggiungendovi numerose e potenti funzioni multimediali e grafiche. DirectX è costituito da vari moduli, ognuno dedicato a una specifica funzione: tra i più importanti citiamo DirectDraw, Direct3d, DirectSound [16],[17],[18].

Il Microsoft DirectSound è il componente di DirectX che gestisce l'audio digitale: effettua mixing a bassa latenza, accelerazione hardware, ed ha accesso diretto alle risorse audio; è compatibile con tutti i file .wav, dove i suoni sono rappresentati formato PCM (Pulse Code Modulation). Il funzionamento di DirectSound consiste nell'assegnare un buffer per ogni file sonoro e per la riproduzione è necessario richiamare il buffer voluto con un apposita istruzione. In questo modo, richiamando più buffers in contemporanea si otterrà una riproduzione simultanea dei vari suoni.

E' possibile modificare alcuni parametri di riproduzione, come il volume e la frequenza: nel nostro caso il primo è fondamentale per effettuare il mixing mentre il secondo non è stato utilizzato ma ciò non significa che non possa trovare applicazione in una futura codifica. Il VisualBasic 5.0, usato per lo sviluppo dell'applicativo, dispone delle librerie DirectX7.0 e DirectX8.0, pertanto ne è stata scelta la versione più recente.

Verranno ora descritte in dettaglio gli aspetti e le procedure inerenti l'inizializzazione di DirectSound8, quindi il caricamento e il mixing dei tre suoni.

2.2.2.1 Sound Buffers Primari e Secondari

Prima dell'inizializzazione è necessario soffermarsi sull'architettura del DirectSound8: sono previsti due differenti tipi di buffer, pertanto è importante conoscerne le differenze.

Quando viene inizializzato DirectSound si crea e viene gestito, automaticamente, il Primary Sound Buffer necessario per il mixing dei suoni e per l'invio di essi al dispositivo d'uscita; è indispensabile quindi la creazione di un Secondary Sound Buffer per contenere e riprodurre ogni singolo suono.

2.2.2.2 L'inizializzazione di DirectSound8

Essendo il DirectSound8 incluso nelle librerie DirectX8 sarà essenziale inizializzare la variabile dx, che crea tutti gli oggetti di DirectX. DS è l'oggetto DirectSound8 e serve per creare e gestire tutta la parte audio wave del programma. Vengono quindi creati tre oggetti di tipo DirectSoundSecondaryBuffer8, ognuno in corrispondenza a un colore fondamentale.

Con l'istruzione "*Set DS = dx.DirectSoundCreate("")*" si crea l'oggetto DirectSound8 che gestisce sia la riproduzione che la registrazione dei file wave mentre con "*DS.SetCooperativeLevel Me.hwnd, DSSCL_NORMAL*" si imposta la priorità dell'accesso alle risorse audio, che prevede 4 opzioni in ordine: Normal, Priority, Exclusive,Write-Primary. A differenza di altre applicazioni, quali i giochi ad esempio, questo software non deve richiedere alta priorità al sottosistema audio. Con questo termina l'inizializzazione vera e propria di DirectSound che ora è pronto per il caricamento dei tre suoni, realizzato dalla funzione LoadWave: di essa vengono tralasciati i dettagli delle prime due righe, che sono molteplici, mentre con un po' di intuito si può notare come le successive istruzioni eseguono l'effettivo caricamento dei tre files di nome r.wav, g.wav e b.wav dal percorso *App.path*.

Una volta caricati i files, DirectSound è pronto per riprodurre i suoni. Sono state anche previste due routines di gestione errore che avvisano l'utente del problema riscontrato: sull'inizializzazione di DirectSound e sul caricamento dei file.

2.2.2.3 Il mixing

In riferimento al codice, la funzione sicuramente più importante di PlayRGBSound si chiama PlayRGB: essa contiene in sequenza le tre equazioni di trasferimento, la copiatura della terna RGB sul primo elemento del vettore della memoria, l'azzeramento della posizione di riproduzione dei suoni, il settaggio delle attenuazioni e le istruzioni per l'avvio della riproduzione sonora. Per effettuare il mixing è necessario impostare a ogni suono la relativa attenuazione, che è stata trattata in precedenza: in riferimento al colore rosso, l'istruzione "*suonoR.SetVolume Rvol*" esegue il settaggio alla variabile Rvol dell'attenuazione di *suonoR*. La variabile Rvol, calcolata in precedenza dalla relativa funzione di trasferimento, contiene il valore in centesimi di dB dell'attenuazione relativa al rosso. La funzione PlayRGB si conclude con l'esecuzione dei tre suoni mediante l'istruzione "*suonoR.Play DSBPLAY_DEFAULT* ".

2.3 Il caricamento di immagini

Per ultimo in questo capitolo, trattiamo brevemente la funzionalità di caricamento delle immagini integrata nel programma, utile per esplorare acusticamente le varie sfumature e tonalità presenti in una qualsiasi immagine. In alto a destra compare un FileBox in cui sono visualizzati tutti i files di tipo immagine contenuti nella cartella dell'applicativo. Cliccando il file scelto avviene l'apertura dell'immagine a dimensioni reali. Lo spazio dedicato all'immagine, di circa 500x500 pixel sembrerebbe abbastanza ridotto ma in realtà è più che sufficiente visto che le immagini da utilizzare per questo scopo non richiedono grandi dimensioni e questa funzionalità ha un utilizzo limitato in sede di sperimentazione poiché quest'ultima avviene richiamando direttamente i colori in memoria precedentemente reimpostati.

Le immagini più interessanti saranno quelle contenenti più sfumature possibili, quindi un eccellente esempio è rappresentato dal Gamut RGB [19], che contiene tutte le possibili sfumature cromatiche.

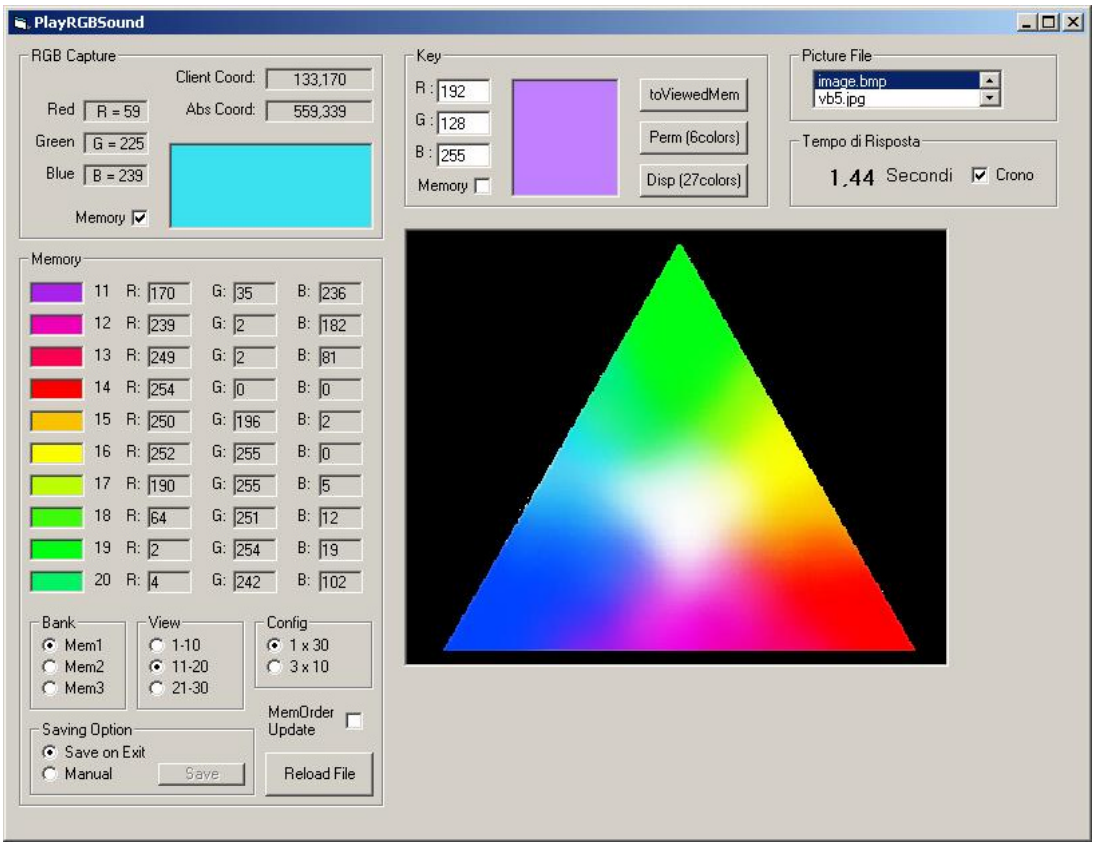
Capitolo 3

PlayRGBSound

Aspetti Secondari:
L'obiettivo della sperimentazione

Una volta raggiunti gli obiettivi primari, è lecito orientare lo sviluppo del software verso la sperimentazione, fondamentale per trarre conclusioni sull'efficacia di questa codifica del colore.

PlayRGBSound è di fatto un programma nato per la sperimentazione con soggetti non vedenti; pertanto è stato studiato per facilitare e velocizzarne le procedure tramite l'aggiunta di particolari funzioni e strumenti: una memoria configurabile che contiene fino a 90 colori di cui 10 visualizzati su schermo, uno strumento che consente di generare un colore digitando i rispettivi valori RGB, un cronometro per conoscere il tempo di risposta del partecipante, ecc.



3.1 Memoria

Uno sguardo alla tipologia della sperimentazione: è intuibile che una delle chiavi sta nel confronto di colori; una probabile procedura può consistere nel sottoporre un certo numero di non vedenti, in serie, a una sequenza prefissata di colori; d'altra parte l'individuazione sulla tavolozza del colore voluto è un'operazione molto lenta, imprecisa e logorante, perché consiste nel puntare col mouse, ma con molta precisione, il pixel giusto. E' opportuno quindi implementare una memoria tipo cache che contenga gli ultimi colori suonati e averli subito visualizzati su schermo e a disposizione per risuonarli a piacimento, cliccando semplicemente sul colore stesso. Se la capacità della memoria è abbondante, è possibile tenere memorizzata un'intera sequenza di colori e in questo modo la sperimentazione non necessita più del prelievo dei colori sulla tavolozza, risultando quindi estremamente più veloce e facile. Copiando gli elementi della memoria su un file possiamo evitare la ricostruzione della sequenza di colori in memoria ad ogni avvio del programma, prima di essere utilizzata. Incrementandone ancora la capacità è possibile tenere memorizzate più sequenze di colori; a tal fine è necessario organizzare opportunamente la memoria e suddividerla in sottomemorie; un'altra evoluzione, mirata alla flessibilità di utilizzo, consiste nell'offrire la possibilità di usare due diverse configurazioni memoria, lasciando quindi scegliere all'utente quale delle due utilizzare in base al numero di sequenze con cui deve lavorare e alle dimensioni delle stesse. Tutto ciò comporta un aumento della complessità della gestione della memoria, della visualizzazione degli elementi e dei vari pulsanti e selettori per i controlli, quindi alla disposizione sullo schermo dei vari oggetti e all'interfaccia grafica.

3.1.1 Interfaccia grafica

La memoria implementata nella prima versione del software conteneva 10 elementi numerati dall'alto verso il basso, tutti visualizzati in colonna direttamente sullo schermo. Per ogni colore compaiono, in riga, i tre numeri RGB e a fianco a sinistra un riquadro rispettivamente colorato, che rende immediato all'utente il colore memorizzato. Nelle successive versioni del software, in cui è stata ampliata memoria a 30 elementi e poi a 90, si è scelto di tenere visualizzati sempre 10 elementi su schermo, per ovvie ragioni di spazio; esisterà quindi un selettore che consente di scegliere quale decina di colori visualizzare.

3.1.2 Organizzazione

Il software integra un sistema di memorizzazione composto da tre memorie indipendenti da 30 elementi l'una, selezionabili mediante il selettore Bank, che distingue appunto quale delle 3 memorie è quella corrente. Tramite il selettore Config sono disponibili all'utente due differenti granularità: 1x30 e 3x10. Nel primo caso ogni memoria contiene 30 elementi contigui e il selettore View seleziona il blocco da visualizzare; questa configurazione consente quindi di memorizzare 3 lunghe sequenze di colori. La configurazione 3x10 consiste nella suddivisione di ogni memoria in 3 blocchi indipendenti, per un totale quindi di 9 blocchi da 10 elementi; in questo caso il blocco visualizzato su schermo è il blocco attivo ed è l'unico a essere influenzato dall'ingresso di nuovi colori. Non deve trarre inganno l'invarianza della numerazione delle celle di memoria visualizzate: anche in modalità 3x10 la numerazione rimane da 1 a 30 in modo tale da evitare confusione nel distinguere il banco corrente.

3.1.3 Aggiornamento e controlli

Quando un colore viene riprodotto il software provvede anche alla sua memorizzazione, se questa è stata abilitata. Il programma contiene tre diversi strumenti che restituiscono colori sottoforma di suoni: RGB Capture, Key e Memory; ognuna di queste sorgenti ha un controllo on off, chiamato Memory per i primi due e MemOrderUpdate per il terzo, che consente appunto di abilitare o disabilitare l'accesso alla memoria. L'immissione di nuovi colori è regolata da un semplice filtro che esclude, tramite un confronto, i colori già presenti in memoria, per non sprecarla inutilmente. L'ordine d'immissione è ovviamente cronologico e l'ultimo colore memorizzato è il primo in alto e seguono verso il basso i precedenti, in numerazione crescente.

Sono necessarie alcune precisazioni: delle tre sorgenti di colori riprodotti è facile intuire che la terza, la memoria, è diversa dalle altre due, semplicemente perché riceve dati da se stessa: quando si clicca un colore in essa contenuto viene attivata la stessa procedura usata per le altre due sorgenti, quindi il colore viene processato dal filtro. Quest'ultimo, trovando una corrispondenza con uno dei suoi elementi, introduce il colore in prima posizione di memoria ed elimina il suo doppione. Il risultato è che il contenuto della memoria non è stato cambiato, ma è stato semplicemente riordinato in senso cronologico; ora si spiega il significato di MemOrderUpdate.

Usufruendo di questo meccanismo è possibile riordinare a piacimento gli elementi della memoria semplicemente cliccando sui vari colori contenuti.

3.1.4 Salvataggio e lettura dati su file

Per sopperire alla volatilità della memoria è stato implementato l'uso del file mem.txt, posto sulla directory dell'applicativo, consentendo quindi all'utente di non perdere alcun dato alla chiusura del programma.

In fase di avvio una funzione carica i dati dal file e li copia nella memoria, ripristinandola. Nel caso in cui il caricamento del file non vada a buon fine, ad esempio quando il file mem.txt non esiste, una funzione errore provvede alla creazione dello stesso e i 90 elementi vengono tutti settati sul colore nero 0,0,0.

Per quanto concerne il salvataggio dei dati, sono previste due modalità: quella automatica e quella manuale. La prima è di default e prevede che all'atto della chiusura del programma venga avviato automaticamente il salvataggio degli elementi della memoria; i dati vengono così congelati e al successivo avvio ritroveremo i gli stessi colori dell'ultima sessione di lavoro. Scegliendo l'opzione del salvataggio manuale invece si disabilita quello automatico: il salvataggio può essere effettuato solo cliccando l'apposito tasto. Qualora il software dovesse essere chiuso senza aver salvato, il file non viene modificato. Su questa direzione è stato aggiunto il pulsante Reload File, utile per annullare modifiche involontarie del contenuto della memoria.

3.1.5 Implementazione

La memoria, suddivisa in tre parti uguali, si basa su un unico vettore T di 90 terne di numeri da 1 byte. Pertanto si è creato un tipo di dati strutturato di nome TernaRGB. I selettori Bank e View agiscono su due variabili di stato, rispettivamente h e j: la prima rappresenta la selezione delle memorie codificate in tre numeri: 0 corrisponde alla prima memoria, 30 alla seconda e quindi 60 alla terza. Allo stesso modo j sarà 0 se il selettore è in prima posizione, 10 se è nella seconda e 20 se è selezionata la terza decina di elementi. Queste due variabili agiranno come offsets agli indici del vettore nella funzione che gestisce la visualizzazione su schermo e identificano la memoria corrente nella funzione di aggiornamento.

L'aggiornamento della memoria, eseguito dalla funzione chiamata MemoryUpdate, consiste principalmente in un algoritmo contenente operazioni di confronto, scrittura e conseguente riordinamento del vettore memoria. La funzione PlayRGB termina con il salvataggio della terna riprodotta sull'elemento 0 del vettore T; di conseguenza MemoryUpdate esegue un confronto di questa terna con tutti gli elementi della memoria corrente. Se non è presente, viene scritta sul primo elemento e vengono spostati di un passo tutti gli altri. Se invece la terna in T(0) è presente sull'elemento T(x) generico, la scrive ugualmente sul primo elemento ed esegue lo spostamento fino a T(x) compreso, lasciando invariati i successivi.

In maniera analoga agli altri selettori Config agisce sulle dimensioni della porzione di vettore da trattare per l'aggiornamento: se è selezionata la modalità 1x30 MemoryUpdate agisce su 30 elementi ed è attivo l'offeset h che determina quale delle tre memorie utilizzare. Nel caso in cui invece si lavori in modalità 3x10 la funzione agisce solo su 10 elementi contigui e per determinare quale delle 9 sotto-memorie sia quella corrente, sono necessarie le due variabili h e j che intervengono come offsets: j permette di identificare le tre sottomemorie di ogni banco individuato da h.

3.2 Key

Per superare gli ormai noti limiti del prelievo del colore sulla tavolozza, è stato implementato un semplice strumento che permette di restituire il colore cercato immettendo da tastiera le tre componenti RGB. Sono presenti tre caselle testo di input sulle quali agisce un filtro che accetta solo numeri interi tra 0 e 255 digitati da tastiera; ogni casella è quindi associata a una componente fondamentale e viene immediatamente visualizzato il colore su un rettangolo; esattamente come accade per i rettangoli della memoria, cliccandoci sopra viene riprodotto il relativo suono e se il controllo Memory è abilitato, il colore verrà anche inserito in memoria.

Esiste una funzione che consente di copiare il colore sul segmento di memoria visualizzato, cosa che può essere utile considerando che in caso di avvio del software senza file mem.txt tutta la memoria viene settata di default con colore 0,0,0 cioè il nero.

Questo strumento è molto utile per rendere disponibili colori particolari o colori i quali pesi sono scalati a pochi livelli: utilizzando per esempio soltanto 5 livelli per componente (1=255, ¾=192, ½=128, ¼=64, 0=0) siamo

già in grado di distinguere 125 elementi cromatici diversi. In questa direzione sono state aggiunte due funzioni generatrici di colori: Perm e Disp

3.2.1 La funzione Perm

Dati i tre numeri delle componenti, la funzione Perm genera i sei colori (3!=6) ottenuti dalle permutazioni di essi e li va a scrivere direttamente in memoria, nel segmento selezionato. L'operazione viene come sempre elaborata dal filtro di ingresso della memoria evitando di avere più celle che contengono lo stesso colore.

Il codice relativo non prevede iterazioni ed è stato scritto per esteso, ripetendo per sei volte un blocco di codice contentente la chiamata MemoryUpdate ove di volta in volta vengono cambiate le assegnazioni delle variabili generando le permutazioni.

3.2.2 La funzione Disp

Analogamente alla funzione Perm, questa funzione genera le disposizioni con ripetizione relative ai tre valori immessi. Ci aspettiamo quindi 3^3=27 nuove terne di valori che verranno immesse in memoria, che deve essere però configurata a 3x30; in caso contrario non è possibile memorizzare i 27 valori e pertanto la funzione Disp viene disabilitata.

Il codice con cui è stata implementata la funzione si sviluppa in tre iterazioni FOR NEXT annidate, con relativi contatori, che lavorano su di un vettore di tre elementi su cui in precedenza sono stati copiati i numeri immessi da tastiera. Si hanno un totale di 27 cicli in cui ogni colore originato viene proposto al filtro della memoria tramite la funzione MemoryUpdate

3.3 Cronometro

Uno dei dati più importanti rilevati dalla sperimentazione è proprio il tempo di risposta del partecipante agli stimoli acustici. Le precedenti sperimentazioni [13] sono state effettuate tramite cronometro fisico che ora non è più necessario: PlayRGBSound integra infatti un apposito cronometro. Questo, una volta abilitato il relativo controllo, parte nell'istante in cui si riproduce un qualsiasi colore e si arresta, visualizzando il tempo trascorso,

premendo un qualunque tasto della tastiera, azione che potrà essere compiuta direttamente dal partecipante. Alla successiva riproduzione di un colore il cronometro viene prima azzerato e poi avviato. Nel caso non venga più premuto alcun tasto il cronometro si fermerà a 300 secondi.
La precisione è di 10 ms.

Conclusioni

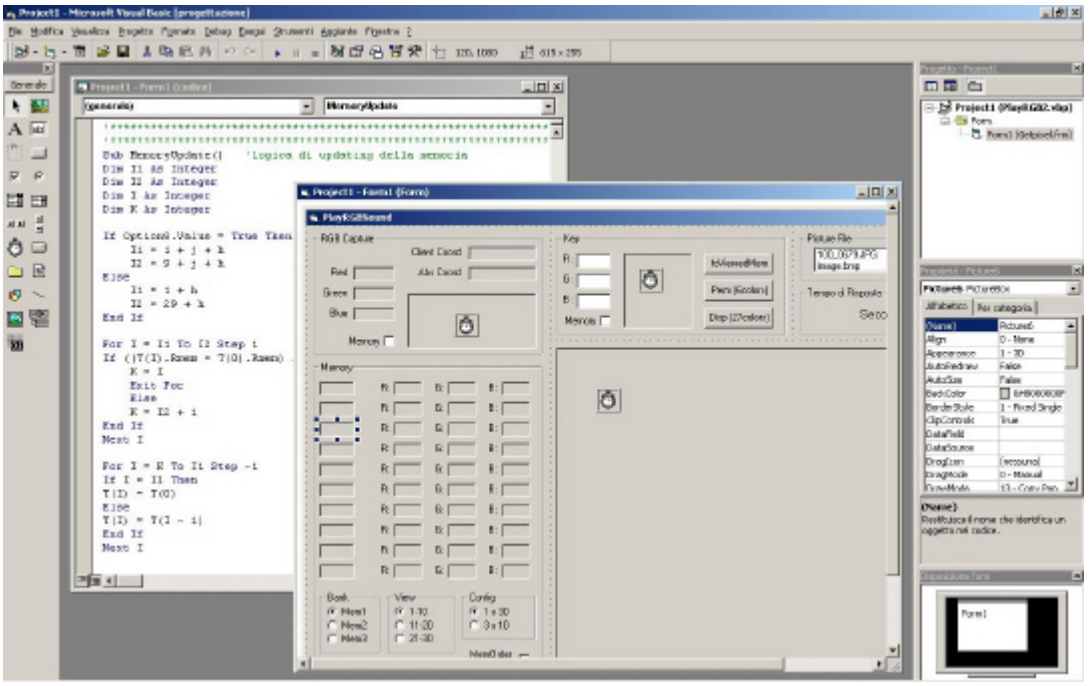
Il presente lavoro di tesi consiste nell'implementazione di un programma per la resa acustica del colore, ottenuta miscelando una base di tre suoni. L'innovazione ottenuta rispetto a programmi preesistenti è l'aver reso possibile la rappresentazione di una tavolozza continua anziché discreta.

L'interfaccia grafica, semplice e intuitiva, l'integrazione di diversi strumenti atti a gestire e velocizzare le procedure di sperimentazione e l'elevata portabilità lo rendono un valido supporto per la sperimentazione di questo tipo di codifica del colore.

Il sistema, già collaudato, è pronto per la sperimentazione.

Appendice

Il codice



```
Option Explicit

'variabili x il caricamento dell'immagine
Dim Pic As Picture
Dim kk As Integer

'variabili inizializzazzione DirectSound
Dim dx As New DirectX8
Dim DS As DirectSound8
Dim suonoR As DirectSoundSecondaryBuffer8
Dim suonoG As DirectSoundSecondaryBuffer8
Dim suonoB As DirectSoundSecondaryBuffer8
Dim Rvol As Long
Dim Gvol As Long
Dim Bvol As Long

'variabili rgb key
Dim Rkey As Integer
Dim Gkey As Integer
Dim Bkey As Integer
```

```
'variabili e funzioni API per la cattura rgb
Dim R As Integer, G As Integer, B As Integer
Private Type POINTAPI
   X As Long
   Y As Long
End Type

Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal X As Long, ByVal Y
As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As
Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long
Private Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long
Private Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As Long
Private Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, ByVal Y As
Long) As Long
Private Declare Function ReleaseDC Lib "user32" (ByVal hwnd As Long, ByVal hdc As Long) As
Long
Private Declare Function ScreenToClient Lib "user32" (ByVal hwnd As Long, lpPoint As
POINTAPI) As Long
Private Declare Function WindowFromPoint Lib "user32" (ByVal xPoint As Long, ByVal yPoint As
Long) As Long

'variabili per la memoria
Private Type TernaRGB
   Rmem As Integer
   Gmem As Integer
   Bmem As Integer
End Type

Dim T(90) As TernaRGB
Dim Bank As Byte
Dim j As Integer 'numero della decina di valori di memoria (0, 10, 20)
Dim h As Integer 'numero banco di memoria (0, 30, 60)
'variabile conteggio cronometro
Dim TimeCount As Integer

'****************************** FINE DICHIARAZIONI **********************

'dettagli cronometro
Private Sub Check4_Click()
Label75.Caption = ""
End Sub

'dettagli key
Private Sub Command1_Click()
Bank = j + h
WriteToBank
End Sub

'generazione delle permutazioni dei valori inseriti, key
Private Sub Command2_Click()
T(0).Rmem = Rkey
T(0).Gmem = Gkey
T(0).Bmem = Bkey
MemoryUpdate
T(0).Rmem = Bkey
```

```
T(0).Gmem = Rkey
T(0).Bmem = Gkey
MemoryUpdate
T(0).Rmem = Gkey
T(0).Gmem = Bkey
T(0).Bmem = Rkey
MemoryUpdate
T(0).Rmem = Rkey
T(0).Gmem = Bkey
T(0).Bmem = Gkey
MemoryUpdate
T(0).Rmem = Bkey
T(0).Gmem = Gkey
T(0).Bmem = Rkey
MemoryUpdate
T(0).Rmem = Gkey
T(0).Gmem = Rkey
T(0).Bmem = Bkey
MemoryUpdate
DrawMemory
End Sub

'generazione delle disposizioni con ripetizione dei valori inseriti, key
Private Sub Command3_Click()
Dim disp(2) As Byte
Dim d1, d2, d3 As Byte
disp(0) = Rkey
disp(1) = Gkey
disp(2) = Bkey
For d1 = 0 To 2 Step 1
   For d2 = 0 To 2 Step 1
      For d3 = 0 To 2 Step 1
      T(0).Rmem = disp(d3)
      T(0).Gmem = disp(d2)
      T(0).Bmem = disp(d1)
      MemoryUpdate
      Next d3
   Next d2
Next d1
DrawMemory
End Sub

Private Sub Command5_Click()
LoadFile
End Sub

Private Sub Command6_Click()
Save
End Sub

Private Sub Form_Load()
Dim I As Integer
'gestione DirectSound
Me.Show
On Local Error Resume Next
Set DS = dx.DirectSoundCreate("")
```

```
If Err.Number <> 0 Then
      MsgBox "Unable to start DirectSound"
      End
End If
DS.SetCooperativeLevel Me.hwnd, DSSCL_NORMAL
LoadWave

'gestione caricamento immagine
File1.Path = App.Path

'caricamento file immagine di memoria
LoadFile

End Sub

Private Sub Form_Unload(Cancel As Integer)
'salvataggio su file in chiusura del programma (congelamento)
If Option9.Value = True Then
   Save
End If

End Sub

'*************** redraw memoria su cambiamento option - visualizzazione ********
Private Sub Option1_Click()
UpdateConfig
DrawMemory
End Sub
Private Sub Option2_Click()
UpdateConfig
DrawMemory
End Sub
Private Sub Option3_Click()
UpdateConfig
DrawMemory
End Sub
Private Sub Option4_Click()
UpdateConfig
DrawMemory
End Sub
Private Sub Option5_Click()
UpdateConfig
DrawMemory
End Sub
Private Sub Option6_Click()
UpdateConfig
DrawMemory
End Sub
'*************** fine redraw

'click sul riquadro di key
Private Sub Picture2_Click()
R = Rkey
G = Gkey
B = Bkey
PlayRGB
```

```
If Check1.Value = 1 Then
   MemoryUpdate
End If
StartCrono
End Sub

'click sull'immagine caricata
Private Sub Picture3_Click()

PlayRGB

If Check2.Value = 1 Then
   MemoryUpdate
End If
StartCrono
End Sub

'stop cronometro
Private Sub Picture2_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub

Private Sub Picture3_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub
Private Sub Picture4_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub
Private Sub Picture5_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub
Private Sub Picture6_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub
Private Sub Picture7_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub
Private Sub Picture8_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub
Private Sub Picture9_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub
Private Sub Picture10_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub
Private Sub Picture11_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub
Private Sub Picture12_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub
Private Sub Picture13_KeyDown(KeyCode As Integer, Shift As Integer)
StopCrono
End Sub
```

```
'****************************************
'****************************************
' Memorie PLAY Click

Private Sub Picture4_Click()
MemoryClick
StartCrono
End Sub
Private Sub Picture5_Click()
MemoryClick
StartCrono
End Sub
Private Sub Picture6_Click()
MemoryClick
StartCrono
End Sub
Private Sub Picture7_Click()
MemoryClick
StartCrono
End Sub
Private Sub Picture8_Click()
MemoryClick
StartCrono
End Sub
Private Sub Picture9_Click()
MemoryClick
StartCrono
End Sub
Private Sub Picture10_Click()
MemoryClick
StartCrono
End Sub
Private Sub Picture11_Click()
MemoryClick
StartCrono
End Sub
Private Sub Picture12_Click()
MemoryClick
StartCrono
End Sub
Private Sub Picture13_Click()
MemoryClick
StartCrono
End Sub
'****************************************
'****************************************

'inizio filtri text box
Private Sub Text1_KeyPress(KeyAscii As Integer)
If (KeyAscii < 48 Or KeyAscii > 57) Then
   KeyAscii = 0
   Text1.Text = ""
End If
If Len(Text1.Text) > 2 Then
```

```
   KeyAscii = 0
End If
End Sub

Private Sub Text2_KeyPress(KeyAscii As Integer)
If (KeyAscii < 48 Or KeyAscii > 57) Then
   KeyAscii = 0
   Text2.Text = ""
End If
If Len(Text2.Text) > 2 Then
   KeyAscii = 0
End If
End Sub

Private Sub Text3_KeyPress(KeyAscii As Integer)
If (KeyAscii < 48 Or KeyAscii > 57) Then
   KeyAscii = 0
   Text3.Text = ""
End If
If Len(Text3.Text) > 2 Then
   KeyAscii = 0
End If
End Sub
'fine filtri text box

'rgb capture - utilizzo delle API
Private Sub Timer1_Timer()
   Static lX As Long, lY As Long
   On Local Error Resume Next
   Dim P As POINTAPI, h As Long, hD As Long, C As Long
   Dim lColor As Long


   GetCursorPos P
   If P.X = lX And P.Y = lY Then Exit Sub
   lX = P.X: lY = P.Y
   lblData(0).Caption = lX & "," & lY
   h = WindowFromPoint(lX, lY)

   hD = GetDC(h)

   ScreenToClient h, P
   lblData(1).Caption = P.X & "," & P.Y
   C = GetPixel(hD, P.X, P.Y)
   If C = -1 Then
      BitBlt Picture1.hdc, 0, 0, 1, 1, hD, P.X, P.Y, vbSrcCopy
      C = Picture1.Point(0, 0)
   Else
      Picture1.PSet (0, 0), C
   End If
   ReleaseDC h, hD
   Picture1.BackColor = C

    'estrapolazione della terna rgb dal numero long
    lColor = C
    R = lColor Mod &H100
```

```
    lColor = lColor \ &H100
    G = lColor Mod &H100
    lColor = lColor \ &H100
    B = lColor Mod &H100

   'visualizzazione
   lblData(2).Caption = " R = " & R
   lblData(3).Caption = " G = " & G
   lblData(4).Caption = " B = " & B
End Sub

'caricamento immagine
Private Sub File1_DblClick()
kk = File1.ListIndex
Set Pic = LoadPicture(App.Path & "\" & File1.List(kk))
Set Picture3.Picture = Pic
End Sub



'funzione per il caricamento dei 3 file wav sui buffer
Sub LoadWave()
Dim bufferDesc As DSBUFFERDESC
bufferDesc.lFlags = DSBCAPS_CTRLFREQUENCY Or DSBCAPS_CTRLPAN Or
DSBCAPS_CTRLVOLUME Or DSBCAPS_LOCSOFTWARE
Dim sFile As String
sFile = App.Path & "\r.wav"
Set suonoR = DS.CreateSoundBufferFromFile(sFile, bufferDesc)
sFile = App.Path & "\g.wav"
Set suonoG = DS.CreateSoundBufferFromFile(sFile, bufferDesc)
sFile = App.Path & "\b.wav"
Set suonoB = DS.CreateSoundBufferFromFile(sFile, bufferDesc)
If Err.Number <> 0 Then
   MsgBox "Unable to find" + sFile
   End
End If
End Sub

'funzione per la codifica: trasformazione, mixing, riproduzione e copia dei valori in memoria
Sub PlayRGB()

Rvol = (6000 / 255) * R - 6000
Gvol = (6000 / 255) * G - 6000
Bvol = (6000 / 255) * B - 6000

T(0).Rmem = R
T(0).Gmem = G
T(0).Bmem = B

suonoR.SetCurrentPosition 0
suonoG.SetCurrentPosition 0
suonoB.SetCurrentPosition 0

suonoR.SetVolume Rvol
suonoG.SetVolume Bvol
suonoB.SetVolume Gvol
```

```
suonoR.Play DSBPLAY_DEFAULT
suonoG.Play DSBPLAY_DEFAULT
suonoB.Play DSBPLAY_DEFAULT
End Sub



Private Sub Timer2_Timer()

'enable / disable bottone save

If Option10.Value = True Then
   Command6.Enabled = True
   Else: Command6.Enabled = False
End If

'enable / disable bottone disp

If Option7.Value = True Then
   Command3.Enabled = True
   Else: Command3.Enabled = False
End If


'filtro per avere solo un numero da 0 a 255 su textbox

'text1
If (Text1.Text = "") Then
   Exit Sub
End If
If (CInt(Text1.Text) < 256 And CInt(Text1.Text) >= 0) Then
   Rkey = CInt(Text1.Text)
   Else
   Text1.Text = ""
End If

'text2
If (Text2.Text = "") Then
   Exit Sub
End If
If (CInt(Text2.Text) < 256 And CInt(Text2.Text) >= 0) Then
   Gkey = CInt(Text2.Text)
   Else
   Text2.Text = ""
End If

'text3
If (Text3.Text = "") Then
   Exit Sub
End If
If (CInt(Text3.Text) < 256 And CInt(Text3.Text) >= 0) Then
   Bkey = CInt(Text3.Text)
   Else
   Text3.Text = ""
```

```
End If

Picture2.BackColor = RGB(Rkey, Gkey, Bkey)
End Sub

'**************************************************************************
'**************************************************************************
Sub MemoryUpdate()   'logica di updating della memoria
Dim I1 As Integer
Dim I2 As Integer
Dim I As Integer
Dim K As Integer

If Option8.Value = True Then
   I1 = 1 + j + h
   I2 = 9 + j + h
Else
   I1 = 1 + h
   I2 = 29 + h
End If

For I = I1 To I2 Step 1
If ((T(I).Rmem = T(0).Rmem) And (T(I).Gmem = T(0).Gmem) And (T(I).Bmem = T(0).Bmem))
Then
   K = I
   Exit For
   Else
   K = I2 + 1
End If
Next I

For I = K To I1 Step -1
If I = I1 Then
T(I) = T(0)
Else
T(I) = T(I - 1)
End If
Next I

DrawMemory

End Sub

'Gestione dei controlli della memoria - visualizzazione
Sub DrawMemory()

'colori sui riquadri
Picture4.BackColor = RGB(T(j + h + 1).Rmem, T(j + h + 1).Gmem, T(j + h + 1).Bmem)
Picture5.BackColor = RGB(T(j + h + 2).Rmem, T(j + h + 2).Gmem, T(j + h + 2).Bmem)
Picture6.BackColor = RGB(T(j + h + 3).Rmem, T(j + h + 3).Gmem, T(j + h + 3).Bmem)
Picture7.BackColor = RGB(T(j + h + 4).Rmem, T(j + h + 4).Gmem, T(j + h + 4).Bmem)
Picture8.BackColor = RGB(T(j + h + 5).Rmem, T(j + h + 5).Gmem, T(j + h + 5).Bmem)
Picture9.BackColor = RGB(T(j + h + 6).Rmem, T(j + h + 6).Gmem, T(j + h + 6).Bmem)
Picture10.BackColor = RGB(T(j + h + 7).Rmem, T(j + h + 7).Gmem, T(j + h + 7).Bmem)
Picture11.BackColor = RGB(T(j + h + 8).Rmem, T(j + h + 8).Gmem, T(j + h + 8).Bmem)
Picture12.BackColor = RGB(T(j + h + 9).Rmem, T(j + h + 9).Gmem, T(j + h + 9).Bmem)
```

```
Picture13.BackColor = RGB(T(j + h + 10).Rmem, T(j + h + 10).Gmem, T(j + h + 10).Bmem)

'terne
Label6.Caption = T(j + h + 1).Rmem
Label8.Caption = T(j + h + 1).Gmem
Label10.Caption = T(j + h + 1).Bmem
Label20.Caption = T(j + h + 2).Rmem
Label38.Caption = T(j + h + 2).Gmem
Label56.Caption = T(j + h + 2).Bmem
Label21.Caption = T(j + h + 3).Rmem
Label39.Caption = T(j + h + 3).Gmem
Label57.Caption = T(j + h + 3).Bmem
Label22.Caption = T(j + h + 4).Rmem
Label40.Caption = T(j + h + 4).Gmem
Label58.Caption = T(j + h + 4).Bmem
Label23.Caption = T(j + h + 5).Rmem
Label41.Caption = T(j + h + 5).Gmem
Label59.Caption = T(j + h + 5).Bmem
Label24.Caption = T(j + h + 6).Rmem
Label42.Caption = T(j + h + 6).Gmem
Label60.Caption = T(j + h + 6).Bmem
Label25.Caption = T(j + h + 7).Rmem
Label43.Caption = T(j + h + 7).Gmem
Label61.Caption = T(j + h + 7).Bmem
Label26.Caption = T(j + h + 8).Rmem
Label44.Caption = T(j + h + 8).Gmem
Label62.Caption = T(j + h + 8).Bmem
Label27.Caption = T(j + h + 9).Rmem
Label45.Caption = T(j + h + 9).Gmem
Label63.Caption = T(j + h + 9).Bmem
Label28.Caption = T(j + h + 10).Rmem
Label46.Caption = T(j + h + 10).Gmem
Label64.Caption = T(j + h + 10).Bmem

'numerazione
Label65.Caption = (j + 1)
Label66.Caption = (j + 2)
Label67.Caption = (j + 3)
Label68.Caption = (j + 4)
Label69.Caption = (j + 5)
Label70.Caption = (j + 6)
Label71.Caption = (j + 7)
Label72.Caption = (j + 8)
Label73.Caption = (j + 9)
Label74.Caption = (j + 10)

End Sub

'click sui riquadri della memoria
Sub MemoryClick()
PlayRGB
If Check3.Value = 1 Then
   MemoryUpdate
End If
End Sub
```

```
'contatore cronometro
Private Sub Timer3_Timer()
TimeCount = TimeCount + 1
If TimeCount = 30000 Then
   StopCrono
End If
End Sub

Sub StartCrono()
If Check4.Value = 1 Then
   TimeCount = 0
   Timer3.Enabled = True
End If
End Sub

Sub StopCrono()
If Timer3.Enabled = True Then
   Timer3.Enabled = False
   Label75.Caption = TimeCount / 100
End If
End Sub

'scrittura dello stesso colore su 10 elementi
Sub WriteToBank()
Dim I As Byte
R = Rkey
G = Gkey
B = Bkey
PlayRGB
For I = (1 + Bank) To (10 + Bank) Step 1
T(I) = T(0)
'T(I).Rmem = T(0).Rmem
'T(I).Gmem = T(0).Gmem
'T(I).Bmem = T(0).Bmem
Next I
DrawMemory
End Sub

'caricamento file immagine di memoria e gestione degli errori
Sub LoadFile()

On Error Resume Next 'in caso di errore continua!
Dim I As Byte
Open App.Path & "\mem.txt" For Input As #1
If Err.Number = 0 Then
   For I = 1 To 90 Step 1
   Input #1, T(I).Rmem, T(I).Gmem, T(I).Bmem
   Next I
   Close #1
   Else
   MsgBox "File mem.txt non trovato. Reset della memoria"
End If
DrawMemory
End Sub

'salvataggio dati su file
```

```
Sub Save()
Dim I As Integer
Open App.Path & "\mem.txt" For Output As #1    ' Apre il file per l'output.
For I = 1 To 90 Step 1
Write #1, T(I).Rmem, T(I).Gmem, T(I).Bmem
Next I
Close #1
End Sub

'gestione selettori della memoria e aggiornamento delle relative variabili
Sub UpdateConfig()
If Option4.Value Then
   h = 0
ElseIf Option5.Value Then
   h = 30
ElseIf Option6.Value Then
   h = 60
End If


If Option1.Value Then
   j = 0
ElseIf Option2.Value Then
   j = 10
ElseIf Option3.Value Then
   j = 20
End If
End Sub
```

Bibliografia

[1] Zannini. Il colore e la sua misura.

[2] C.I.E. C.I.E. tristimulus values.

[3] http://www.boscarol.com/pages/cs/015-tinta.html
"Attributi percettivi del colore: attributi cromatici: tinta: rosso, giallo, verde e così via"

[4] http://www.boscarol.com/pages/cs/020-saturazione.html
"Attributi percettivi del colore: attributi cromatici: pienezza,croma, saturazione"

[5] C.I.E. The C.I.E. color space.

[6] C.I.E. The C.I.E. chromaticity diagram.

[7] http://www.boscarol.com/pages/cs/540-xy.html
"I colori che l'occhio può vedere: il diagramma CIE 1931"

[8] M. D. Sanders N. G. Page, J. P. Bolger. Auditory evoked phosphenes in optical nerve disease. Journal of Neurol. Neurosurg. Psychiatry, 45:7–12, 1982.

[9] M. Korte J. P. Rauschecker. Auditory compensation of early blindness in cat cerebral cortex. Journal of Neuroscience, 13:4538–4548, 1993.

[10] M. Ferri C. Melchiorri G. Vassura C. Bonivento, L. Di Stefano. The VIDET project: basic ideas and early results. In Workshop on Service Robotics, pages 1–22, 1997.

[11] L. Diaz-Saco N. Sosa J. L. Gonzales Mora A. Rodriguez-Hernandez L. F. Rodriguez-Ramos, H. M. Chulani. Image and sond processing for the creation of a virtual acoustic space for the blind people.

[12] L. F. Rodriguez-Ramos L. Diaz-Saco N. Sosa J. L. Gonzalez-Mora, A. Rodriguez-Hernandez. Development of a new space perception system for blind people, based on the creation of a virtual acoustic space.

[13] Ludovico Ausiello. Codifica sinestetica dei colori mediante suoni in un dispositivo per videolesi. 2003-2004

[14] Renato Spagnolo. Manuale di acustica applicata. UTET, 2001.

[15] http://www.cie.co.at/cie/

[16] http://www.microsoft.com
DirectSound – dsound.doc

[17] http://www.vincedx.altervista.org/Lessons.php?id=57&from=0#
"DirectSound8 - Suonare un Wave"

[18] http://www.freevbcode.com/ShowCode.Asp?ID=1082#Main
"Direct Sound Tutorial"

[19] http://www.boscarol.com/pages/cms/115-gamut.html
"Ogni periferica è un caso a sé: il gamut"