Interpolazione Matrici

Non esiste un algoritmo generale per decomporre una matrice nelle sue primitive e quindi interpolarne le primitive con la successiva. Per interpolare due matrici occorre infatti scomporre questa in traslazione/rotazione/scalatura perché se applicassimo una interpolazione lineare sui singoli componenti, anche se $\sum_i a_i(M\ p) = \sum_i (a_i M)p$ (garantito dalla linearità, con $\sum_i a_i = 1$) non sempre avremmo un risultato accettabile, basti pensare ad un quadrato che viene ruotato di 180°, se ogni vertice venisse banalmente portato nella posizione finale il rettangolo collasserebbe al centro per poi "risorgere" nella posizione finale…

Il problema dell'interpolazione si riduce quindi a dover recuperare le primitive dalla matrice totale. Il problema è però il recupero delle primitive è difficoltoso (se non impossibile) per 3 motivi: incorporazione, ordine e interazione.

- Incorporazione (absorbtion): una sequenza di trasformazioni dà un risultato indistinguibile da una singola trasformazione.
- Ordinamento: l'effetto di una traslazione seguita da una scalatura può essere facilmente ottenuto invertendone l'ordine.
- Interazione (interaction): molte trasformazioni cambiano tutte le colonne della matrice (si pensi alla scalatura, cambia la traslazione)

Possiamo sicuramente, nel caso in cui le trasformazioni sulla matrice siano state fatte nell'ordine traslazione -> rotazione -> scalatura andare a recuperare le primitive dalla matrice (useremo la forma implementa da Windows.Drawing.Drawing2D) generale risolvendo un semplice sistema dove

$$\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{pmatrix} = \begin{pmatrix} sx\cos a & sx\sin a & 0 \\ -sy\sin a & sy\cos a & 0 \\ dx & dy & 1 \end{pmatrix}$$

Ma questo è possibile farlo solamente se, ripeto, è stato rispettato l'ordine TRS (trasl, rot, scal) altrimenti io non avrei modo di sapere come la matrice è composta (già solo con RTR ho risultati complicati da interpretare e comunque data una matrice io non so come è stata composta). Quindi tratterò nel codice solamente casi dove ho una matrice TRS, che mi sembra un'approssimazione lecita, sto dicendo al grafico che se vuole usare il mio sistema di animazione deve comporre la matrice tramite le operazioni TRS in quest'ordine.

Tutto ciò, scritto in codice:

```
open System.Drawing
open System
open System.Windows.Forms

let decomposeMatrix (md:Drawing2D.Matrix) =
    let el = md.Elements
    let transl_x = el.[4]
    let transl_y = el.[5]
    let zoom_x = Math.Sqrt(float(el.[0]**2.f + el.[1]**2.f))
    let zoom_y = Math.Sqrt(float(el.[2]**2.f + el.[3]**2.f))
    let rotAngle = Math.Atan2(float el.[2], float el.[3])
    [|single transl_x; single transl_y; single zoom_x - 1.f; single zoom_y - 1.f; single(-
rotAngle*180./Math.PI)|]
```

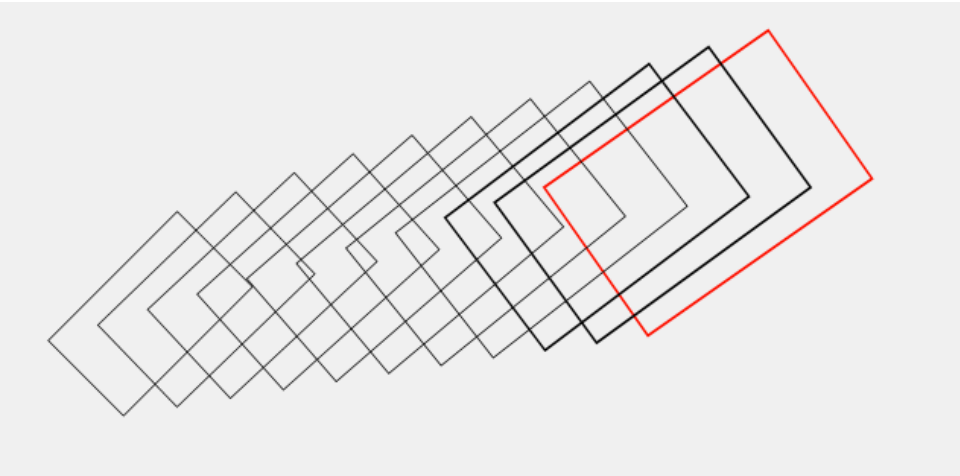
Per interpolare ora basta eseguire un'interpolazione lineare sulle singole primitive, cioè sui valori restituiti dalla funzione decomposeMatrix.

In codice:

```
let compose (els:single[]) =
    let m = new Drawing2D.Matrix()
    m.Translate(els.[0], els.[1])
    m.Rotate(els.[4])
    m.Scale(els.[2], els.[3])
    m

let showFrame (m:Drawing2D.Matrix) (f:Form) (frameNumber:int) path (prev:Drawing2D.Matrix)=
    let els = decomposeMatrix m
    let prevEls = decomposeMatrix prev
    els.[2] <- els.[2] - prevEls.[2]
    els.[3] <- els.[3] - prevEls.[3]
    for i in {0 .. 1 .. 4} do
        els.[i] <- els.[i] / single frameNumber
    for i in {0 .. 1 .. frameNumber-1} do
        let temp:single[] = [|0.f; 0.f; 1.f; 1.f; 0.f|]
        for j in {0 .. 1 .. 4} do
            temp.[j] <- els.[j] * single i
        temp.[2] <- temp.[2] + prevEls.[2] + 1.f
        temp.[3] <- temp.[3] + prevEls.[3] + 1.f
        f.Paint.Add(fun e ->
            let newM = compose(temp)
            newM.Multiply(prev, Drawing2D.MatrixOrder.Append)
            e.Graphics.Transform <- newM
            e.Graphics.DrawPath(Pens.Black, path)
        )
```

Il codice mostrato finora, con l'aggiunta di quello mostrato di seguito, produce il seguente risultato:



Codice per aprire Form e avviare le funzioni mostrate in precedenza correttamente:

```
//matrice finale
let x = new Drawing2D.Matrix()
//matrice iniziale
let y = new Drawing2D.Matrix()
//Trasformazioni su finale, in cui il "mondo" è quello dato dalla matrice iniziale
x.Translate(300.f, 150.f)
x.Rotate(10.f)
x.Scale(1.5f, 1.7f)
```

```
//Trasformazioni su iniziale, indica com'è l'oggetto in partenza
y.Translate(50.f, 25s0.f)
y.Rotate(-45.f)

//Disegno l'oggetto che verrà fatto ruotare
let path = new Drawing2D.GraphicsPath()
let rt = new RectangleF(10.f, 10.f, 120.f, 70.f)
path.AddRectangle(rt)
let el = decomposeMatrix x
let m = compose el
let f = new Form(Text="Matrix Interpolation", TopMost=true)

f.Show()
f.Paint.Add(fun e ->
//Disegna di rosso l'oggetto nella posizione finale risultante
    e.Graphics.SmoothingMode <- Drawing2D.SmoothingMode.AntiAlias
    use temp = x.Clone()
    temp.Multiply(y, Drawing2D.MatrixOrder.Append)
    e.Graphics.Transform <- temp
    e.Graphics.DrawPath(Pens.Red, path)
)
//Applica l'interpolazione e stampa i risultati intermedi
showFrame x f 10 path y

Application.Run(f)
```

Nel codice mostrato la figura che viene fatta ruotare è quella disegnata su *path* mentre la matrice iniziale e finale è rispettivamente la matrice *y* e *x*.