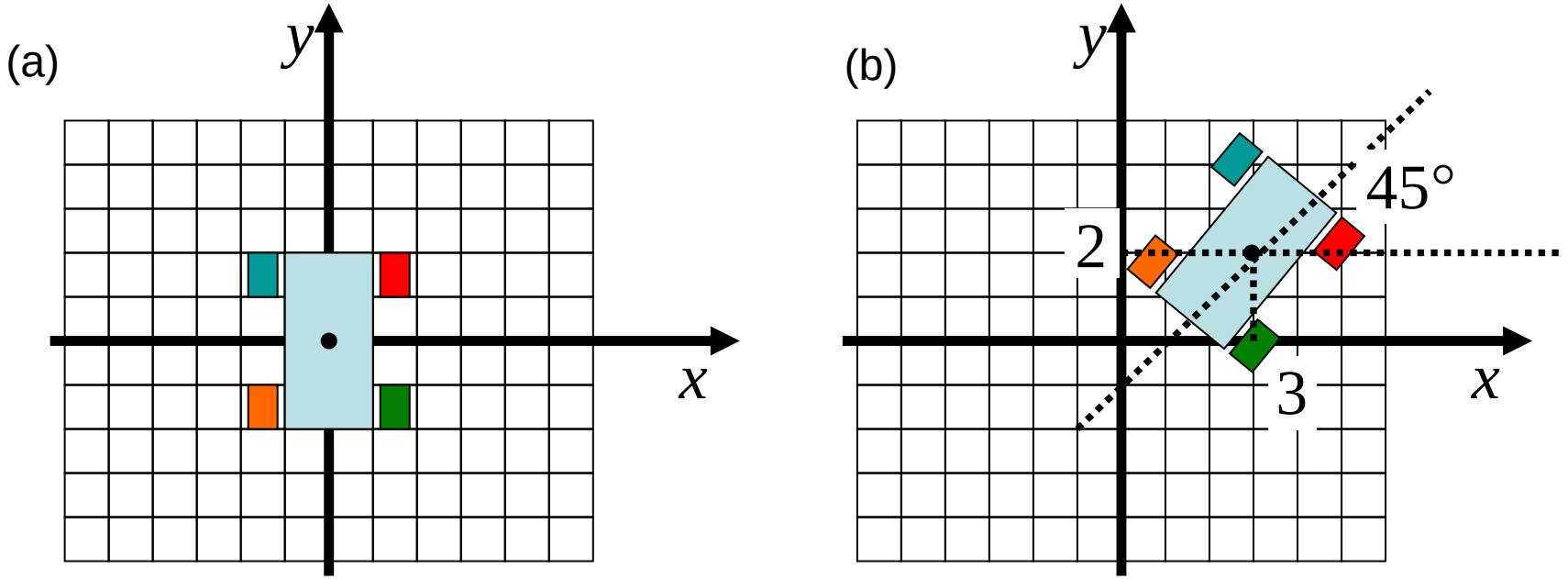
Grafica Computazionale

Ordine delle trasformazioni

Fabio Ganovelli
fabio.ganovelli@isti.cnr.it
a.a. 2005-2006

Ordine delle trasformazioni dell'implementazione

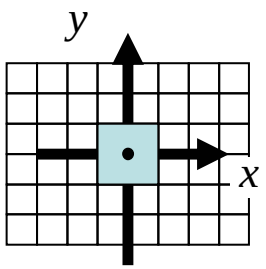
- Poniamo di voler disegnare la "macchina" specificata dalla figura (a) nella posizione specificata nella figura (b) usando la sola primitiva di disegno DrawQuad() che disegna un quadrato di lato 2 centrato in (0,0)



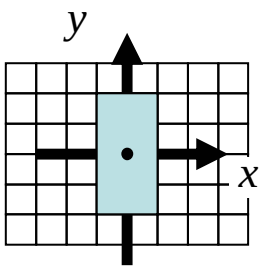
Ruote di dimensione (0.25,0.5) posizionate a (1.5,1.5),(-1.5,1.5),(-1.5,-1.5),(1.5,-1.5)

Incominciamo dalla carrozzeria

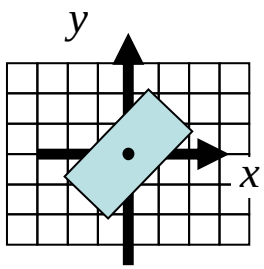
- Scaliamo il quadrato in maniera non uniforme
- Ruotiamolo di 45° in senso orario
- Trasliamolo di (3,2)



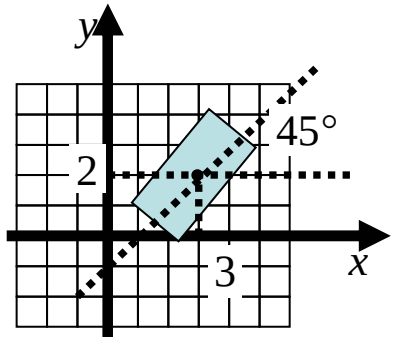
$scale(1,2)$



$rotate(-45)$



$translate(3,2)$



Attenzione! La matrice per cui deve essere moltiplicato ogni punto è:

$$translation\_matrix(3,2) \cdot rotation\_matrix(-45) \cdot scale\_matrix(1,2)$$

Si ma il codice come viene?

Notate come è scritta la Traslate (e le altre trasformazioni)

```
void Translate(float x, float y, float z) {
      *current_matrix = (*current_matrix)*TranslationMatrix(x,y,z);
};
```

Il frammento di codice dovrebbe essere:

```
....
Translate(3,2); // modelview_matrix = TM(3,2)
Rotate(-45);    // modelview_matrix = TM(3,2)*RM(-45)
Scale(1,2);     // modelview_matrix = TM(3,2)*RM(-45) * SM(1,2)
DrawQuad();
....
```

Ogni posizione del vertice p_i del quadrato viene trasformato da:
p_i' = viewport_matrix*projection_matrix*modelview_matrix*p_i

Può apparire controintuitivo: perché non invertiamo le matrici nella Translate e l'ordine dei comandi nel codice?...proviamo

All'incontrè!...(V.Capossela)

Riscriviamo la Traslate (e le altre trasformazioni)

```
void Translate(float x, float y, float z) {
     *current_matrix = TranslationMatrix(x,y,z) *(*current_matrix);
};
```

Il frammento di codice dovrebbe essere:

```
....
Scale(1,2);     // modelview_matrix = SM(1,2)
Rotate(-45);    // modelview_matrix = RM(-45) * SM(1,2)
Translate(3,2); // modelview_matrix = TM(3,2)*RM(-45) * SM(1,2)
DrawQuad();
....
```

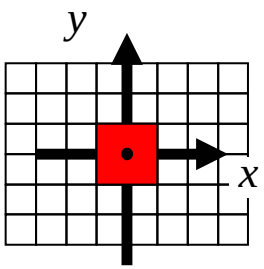
Dopo le trasformazioni otteniamo la stessa matrice!

Ma allora funziona!

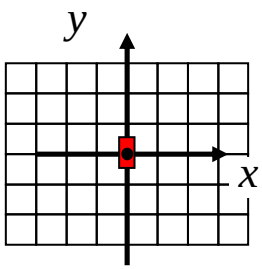
Ok, andiamo avanti…

La ruota

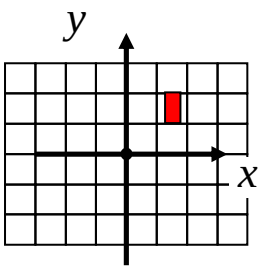
- Scaliamo il quadrato in maniera non uniforme
- Trasliamolo al posto giusto rispetto al frame della macchina
- Da qui in poi applichiamo la rototraslazione applicata alla carrozzeria



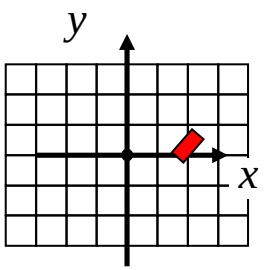
scale(0.25,0.5)



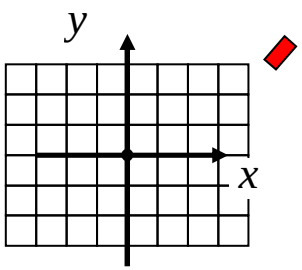
translate(1.5,1.5)



rotate(−45)



translate(3,2)



Il codice

```
....
Scale(1,2);     // modelview_matrix = SM(1,2)
Rotate(-45);    // modelview_matrix = RM(-45) * SM(1,2)
Translate(3,2); // modelview_matrix = TM(3,2)*RM(-45) * SM(1,2)
DrawQuad();
// ruota rossa
modelview_matrix.SetIdentity();
Scale(0.25,0.5);    // modelview_matrix = SM(0.25,0.5)
Translate(1.5,1.5); // modelview_matrix = TM(1.5,1.5)*SM(0.25,0.5)
Rotate(-45);    // modelview_matrix = RM(-45)*TM(1.5,1.5)*SM(0.25,0.5)
Translate(3,2); // modelview_matrix = TM(3,2 )* RM(-45)* TM(1.5,1.5)*SM(0.25,0.5)
DrawQuad();
// ruota ciano
modelview_matrix.SetIdentity();
Scale(0.25,0.5);    // modelview_matrix = SM(0.25,0.5)
Translate(-1.5,1.5); // modelview_matrix = TM(1.5,1.5)*SM(0.25,0.5)
Rotate(-45);    // modelview_matrix = RM(-45)*TM(1.5,1.5)*SM(0.25,0.5)
Translate(3,2); // modelview_matrix = TM(3,2 )* RM(-45)* TM(1.5,1.5)*SM(0.25,0.5)
DrawQuad();
```

Sembra funzionare

- Ok, in verità le ruote le vorrei larghe 1 invece di 0.5
- Modifichiamo il codice…

Il codice con le ruote larghe 1

```
....
Scale(1,2);     // modelview_matrix = SM(1,2)
Rotate(-45);    // modelview_matrix = RM(-45) * SM(1,2)
Translate(3,2); // modelview_matrix = TM(3,2)*RM(-45) * SM(1,2)
DrawQuad();
// ruota rossa
modelview_matrix.SetIdentity();
Scale(0.5,0.5);         // modelview_matrix = SM(0.25,0.5)
Translate(1.5,1.5);     // modelview_matrix = TM(1.5,1.5)*SM(0.25,0.5)
Rotate(-45);            // modelview_matrix = RM(-45)*TM(1.5,1.5)*SM(0.25,0.5)
Translate(3,2);         // modelview_matrix = TM(3,2 )* RM(-45)* TM(1.5,1.5)*SM(0.25,0.
DrawQuad();
// ruota ciano
modelview_matrix.SetIdentity();
Scale(0.5,0.5);     // modelview_matrix = SM(0.25,0.5)
Translate(-1.5,1.5); // modelview_matrix = TM(1.5,1.5)*SM(0.25,0.5)
Rotate(-45);    // modelview_matrix = RM(-45)*TM(1.5,1.5)*SM(0.25,0.5)
Translate(3,2); // modelview_matrix = TM(3,2 )* RM(-45)* TM(1.5,1.5)*SM(0.25,0.5)
DrawQuad();
```

Devo modificare tutte le Scale(…) delle ruote…un po' noioso

DrawWheel(1,2)

- Come sarebbe comoda una primitiva che disegna la ruota della dimensione voluta
- Se l’avessi avuta sarebbe bastato cambiare quella
- La scrivo:

```
void DrawWheel(float width,float height){
Scale(width/2, height/2);
DrawQuad();
};
```

- Ora in che punto del codice la metto?

In nessuno!

```
// ruota rossa
modelview_matrix.SetIdentity();
Scale(0.25,0.5);      // modelview_matrix = SM(0.25,0.5)
Translate(1.5,1.5); // modelview_matrix = TM(1.5,1.5)*SM(0.25,0.5)
Rotate(-45);     // modelview_matrix = RM(-45)*TM(1.5,1.5)*SM(0.25,0.5)
Translate(3,2); // modelview_matrix = TM(3,2 )* RM(-45)* TM(1.5,1.5)*SM(0.25,0.5)
⇨DrawWheel();   // modelview_matrix = SM(0.25,0.5)* RM(-45)* TM(3,2 )*TM(1.5,1.5)
```

Punto sbagliato!

```
// ruota rossa
modelview_matrix.SetIdentity();
Scale(0.25,0.5);      // modelview_matrix = SM(0.25,0.5)
⇨DrawWheel();   // modelview_matrix = SM(0.25,0.5)
Translate(1.5,1.5); // modelview_matrix = TM(1.5,1.5)*SM(0.25,0.5)
Rotate(-45);     // modelview_matrix = RM(-45)*TM(1.5,1.5)*SM(0.25,0.5)
Translate(3,2); // modelview_matrix = TM(3,2 )* RM(-45)* TM(1.5,1.5)*SM(0.25,0.5)
```

Alla fine la matrice è giusta ma la ruota è stata già trasformata!!

Ripetizioni

- E la ripetizione della rototraslazione della carrozzeria la posso evitare?
- No perché le matrici più a sinistra nel prodotto sono le ultime che specifico
- Torniamo all’ordine originario…

```
Translate(3,2);          // modelview_matrix = TM(3,2)
Rotate(-45);             // modelview_matrix = TM(3,2)*RM(-45)

Scale(1,2);              // modelview_matrix = TM(3,2)*RM(-45) * SM(1,2)
DrawQuad();
Scale(1,0.5);            // modelview_matrix = TM(3,2)*RM(-45)

Translate(1.5,1.5);      // modelview_matrix = TM(3,2)*RM(-45)* TM(1.5,1.5)
Scale(0.25,0.5);         // modelview_matrix = TM(3,2)*RM(-45)*TM(1.5,1.5)* SM(0.25,0.5)
DrawQuad();
Scale(4,2);              // modelview_matrix = TM(3,2)*RM(-45)*TM(1.5,1.5)
Translate(-1.5,-1.5);    // modelview_matrix = TM(3,2)*RM(-45)

Translate(-1.5,1.5);     // modelview_matrix = TM(3,2)*RM(-45)* TM(-1.5,1.5)*
Scale(0.25,0.5);         // modelview_matrix = TM(3,2)*RM(-45)*TM(-1.5,1.5)* SM(0.25,0.5)
DrawQuad();
Scale(4,2);              // modelview_matrix = TM(3,2)*RM(-45)*TM(-1.5,1.5)
Translate(1.5,-1.5);     // modelview_matrix = TM(3,2)*RM(-45)
```

Per ogni ruota, trasformo per il frame della carrozzeria, disegno e poi ripristino la modelview_matrix eseguendo l'inversa delle trasformazioni. Oppure?....

Matrix stack

- Ci teniamo una pila di matrici modelview_matrix (projection_matrix)

- PushMatrix(); aggiunge current_matrix allo stack
- PopMatrix(); rimpiazza current_matrix con quella in cima allo stack che viene rimossa dallo stack

```
Translate(3,2);         // modelview_matrix = TM(3,2)
Rotate(-45);            // modelview_matrix = TM(3,2)*RM(-45)

PushMatrix();
Scale(1,2);             // modelview_matrix = TM(3,2)*RM(-45) * SM(1,2)
DrawQuad();
PopMatrix();            // modelview_matrix = TM(3,2)*RM(-45)

PushMatrix();
Translate(1.5,1.5);     // modelview_matrix = TM(3,2)*RM(-45)* TM(1.5,1.5)
Scale(0.25,0.5);        // modelview_matrix = TM(3,2)*RM(-45)*TM(1.5,1.5)* SM(0.25,0.5)
DrawQuad();
PopMatrix();            // modelview_matrix = TM(3,2)*RM(-45)

PushMatrix();
Translate(-1.5,1.5);    // modelview_matrix = TM(3,2)*RM(-45)* TM(-1.5,1.5)
Scale(0.25,0.5);        // modelview_matrix = TM(3,2)*RM(-45)*TM(-1.5,1.5)*SM(0.25,0.5)
DrawQuad();
PopMatrix();            // modelview_matrix = TM(3,2)*RM(-45)
```

Vantaggi del matrix stack

- Praticità: non devo tener traccia di tutte le trasformazioni che ho fatto per invertirle
- Efficienza: non devo applicare tutte le inverse
- Evito l’errore numerico dovuto alle operazioni in floating point