Costruzione di Interfacce
Lezione 21
Trackball

cignoni@iei.pi.cnr.it
http://vcg.isti.cnr.it/~cignoni

Ripulire

- I dati principali dell'app QTMoebius erano contenuti nella classe CIMoebius
  - chi possedeva l'oggetto?
    - soluzione brutta e pasticciata: una variabile globale dentro un cpp.
  - Diamo il possesso dell'oggetto al main form
  - la CIGLWidget deve solo avere un puntatore all'oggetto da rendere.
    - le var private SideNum e twist non servono piu'
    - gli slot non servono piu' qui.

Aggiungere al mainform

- Tutto dal designer
- nell Object explorer
  - aggiungere CIMoebius.h negli include delle definition
  - aggiungere CIMoebius ring tra i membri privati della classe Mainform
  - aggiungere rebuildRing() tra gli slot

CIGLWidget

- togliere dal cpp la var globale pp
- togliere gli slot dal .h e
- togliere le var twist e sidenum
  - e la loro init dal costruttore.
- togliere la parte di creazione dell'anello dalla initializeGL()
- aggiungere un puntatore pubblico a CIMoebiusRing nella classe.

designer

- ristabilire le giuste connessioni
  - togliere quelle dagli spinbox a glwidget
  - e metterle alla rebuild ring

MainForm

- scrivere l'implementazione della rebuild ring

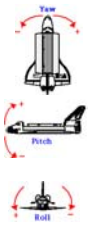
```
void MainForm::rebuildRing()
{
      CIMoebius::myShape s;
      s.BuildNGon(SideNumSpinBox->value(),1);
      ring.GenerateMoebiusRing(s,36,3,
           TwistSpinBox->value()*
           360/SideNumSpinBox->value());
       GLView->update();
}
void MainForm::init()
{
   GLView->ring=&ring;
   emit rebuildRing();
}
```

Interfacce di rotazione

- Come può un utente specificare una rotazione tramite un interfaccia?
- Due modalità:
  - Diretta: specifica valori numerici esatti
  - Interattiva: tramite movimenti del mouse
- Come rappresento una rotazione?
  - Euler Angle
  - Axis/angle
  - Quaternions

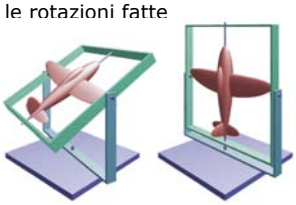
Euler Angle

- Una rotazione viene espressa come una serie di tre rotazioni sui tre assi.
- Deriva dal modo con cui si descrive l'orientamento di un aereo
  - Yaw
  - Pitch
  - Roll
- Intuitivo per piccoli valori di pitch e roll



Euler Angle

- Problema ordine rotazione
  - Il risultato dipende dall'ordine in cui faccio le tre rotazioni
- Problema Gimbal Lock
  - In alcune situazioni le rotazioni fatte su un asse possono su un altro asse
  - Se il pitch è a 90 gradi yaw e roll si possono annullare a vicenda.

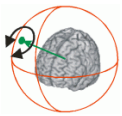


Gimbal lock nelle interfacce

- Capita ad esempio quando cerco di far specificare gli euler angle interattivamente all'utente:
  - Up/down: rot asse x
  - Left/right: rot asse y
  - Pgup/pgdn: rot asse z
- Si incarta.

Axis/angle

- Approccio OpenGL
  - Si specifica un'asse di rotazione e un angolo di rotazione
  - Molto generico
  - Poco intuitivo
    - Qual'e l'asse di rotazione per girare la testa in modo da guardare in basso a destra?



Quaternioni

- Cos'è un quaternione?
- Un estensione dei numeri complessi,

$$q = w + xi + yj + zk \quad dove \quad i \cdot i = j \cdot j = k \cdot k = -1$$

- Spesso rappresentato come una coppia scalare-vettore:

$$q = [w, \mathbf{v}] \quad dove \quad \mathbf{v} = (x, y, z)$$

Quaternioni

- Magnitudine

$$\|q\| = \sqrt{w^2 + x^2 + y^2 + z^2}$$

- Normalizzazione a quaternione unitario

$$q = q / \|q\|$$

Somma e prodotto

- Dati due quaternioni

$$q_1 = w_1 + x_1 i + y_1 j + z_1 k \quad e \quad q_2 = w_2 + x_2 i + y_2 j + z_2 k$$

$$q_1 = [w_1, \mathbf{v}_1] \; e \; q_2 = [w_2, \mathbf{v}_2] \quad dove \quad \mathbf{v}_1 = (x_1, y_1, z_1) \; e \; \mathbf{v}_2 = (x_2, y_2, z_2)$$

- Si definisce

$$q_1 + q_2 = [w_1 + w_2, \quad \mathbf{v}_1 + \mathbf{v}_2]$$

$$q_1 * q_2 = [w_1 w_2 - \mathbf{v}_1 \cdot \mathbf{v}_2, \quad \mathbf{v}_1 w_2 + w_1 \mathbf{v}_2 + \mathbf{v}_1 \times \mathbf{v}_2]$$

- Identità
  - somma $q_{I+} = [0, \quad (0,0,0)]$
  - prodotto $q_{I*} = [1, \quad (0,0,0)]$

Quaternioni e rotazioni

- Ogni quaternione unitario corrisponde ad una rotazione in 3d
- La moltiplicazione tra due quaternioni corrisponde alla composizione delle due rotazioni

Conversioni

- Da quaternione a matrice

$$\begin{bmatrix} 1-2x^2-2z^2 & 2xy-2wz & 2xz+2wy \\ 2xy+2wz & 1-2x^2-2z^2 & 2xy-2wx \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 \end{bmatrix}$$

- Da quaternione ad axis/angle

$$q = [w, \mathbf{v}] \quad axis = \mathbf{v} / |\mathbf{v}| \quad e \quad angle = 2\arccos(w)$$

Conversioni

- Da axis angle a quaternioni

$$axis = (a_x, a_y, a_z) \; e \; angle = \theta$$

$$q = [\mathbf{v}, w]$$

$$\mathbf{v} = (a_x \cdot \sin(\tfrac{\theta}{2}), a_y \cdot \sin(\tfrac{\theta}{2}), a_z \cdot \sin(\tfrac{\theta}{2}))$$

$$w = \cos(\tfrac{\theta}{2})$$

- Da euler angle a quaternion

$$q_x = [\cos(\tfrac{a}{2}), \quad (\sin(\tfrac{a}{2}),0,0)]$$

$$q_y = [\cos(\tfrac{b}{2}), \quad (0,\sin(\tfrac{b}{2}),0)]$$

$$q_z = [\cos(\tfrac{c}{2}), \quad (0,0,\sin(\tfrac{c}{2}))]$$

$$q = q_x * q_y * q_z$$

Evitare il gimbal lock

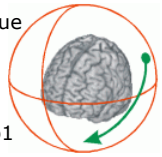
- Euler angle è molto intuitivo per piccole rotazioni: se ruoto di angoli piccoli quello che ottengo è esattamente quello che mi aspetto
- Soluzione:
  - Tenere la rotazione come un quaternione
  - Ad ogni pressione di tasto generare un quaternione corrispondente al piccolo euler angle
  - Ad es. se premo *left* genero un quaternione

$$q_y = [\cos(\tfrac{\delta}{2}), \quad (0,\sin(\tfrac{\delta}{2}),0)]$$

  - Comporre il risultato con moltiplicazione tra quaternioni e tenere il risultato come base;

Trackball

- Come si mappa il movimento del mouse in una rotazione?
  - Si immagina una sfera circoscritta all'oggetto con cui si vuole interagire
  - Ogni drag del mouse definisce due punti p1 e p2(inizio e fine del drag) sulla sfera
  - Si considera la rotazione che descrive l'arco di cerchio sulla superficie sferica delimitato da p1 e p2



Trackball

- La rotazione così calcolata viene trasformata in un quaternione e composta con la trasf corrente
- Se una volta rilasciato il mouse, si continua comporre con l'ultimo quaternione calcolato, si ottiene l'effetto di spinning.

Rotazioni senza trackball

- si mappa il movimento di drag del mouse in x e y in rotazioni in y e x.
- occorre uno stato delle rotazione nella CIGLWidget
- la rotazione specificata deve ovviamente essere relativa a dove ho iniziato a fare il dragging
- notare come funziona bene per piccole rotazioni ma se una delle due rot supera 90 gradi diventa difficile controllare l'altra

CIGLWidget.h

```
void mousePressEvent(QMouseEvent*e);
void mouseMoveEvent(QMouseEvent*e);
void mouseReleaseEvent(QMouseEvent*e);
private:
// posizione iniziale del dragging del mouse
      int sx,sy;
// variazione della rotazione
      float drx,dry;
// rotazione corrente sui due assi
      float rotx, roty;
```


CIGLWidget.cpp

```
void CIGLWidget::mousePressEvent(QMouseEvent*e)
{
   sx=e->x();
   sy=e->y();
   update();
}

void CIGLWidget::mouseMoveEvent(QMouseEvent*e)
{
   drx=(float(e->y()-sy)/size().height())*180.0f;
   dry=(float(e->x()-sx)/size().width())*180.0f;
   update();
}

void CIGLWidget::mouseReleaseEvent(QMouseEvent*e)
{
   rotx+=drx;
   roty+=dry;
   drx=0;
   dry=0;
   update();
}
```


CIGLWidget::paintGL()

```
void CIGLWidget::paintGL()
{
  glMatrixMode (GL_MODELVIEW);
  glClear(GL_COLOR_BUFFER_BIT |GL_DEPTH_BUFFER_BIT  );
  glLoadIdentity();
  gluLookAt(2,5,12,0,0,0,0,1,0);
  glColor3f(1,.4,.4);
  QString tmp;
  glRotatef(roty+dry,0,1,0);
  glRotatef(rotx+drx,1,0,0);
  ring->m.Draw();
}
```

CITrackball

- Classe che implementa una trackball
- Di solito è un oggetto della glview
- Interfaccia fondamentale nella glview
  - Si deve gestire Mousedown/mouseup/Mousemove
  - Avere un membro dove tenere la mat di rotazione corrente
  - Comunicare Resize alla trackball

CITrackball

```
class CIGLWidget
{
…
   // Gestione trackball
  CITrackball m_tb;
  Matrix44f m_matRot;

};

void CIGLWidget ::init()
{
…
   tb.Init(size.width(),size.height());
   matRot.SetIdentity();
}
void CIGLWidget::resizeGL( int w, int h )
{
…
   tb.Resize( w,h );
…
}
```


CITrackball

```
void CIGLWidget::mousePressEvent(QMouseEvent*e)
{
  tb.MouseDown( e->x(), e->y(), 0 );
   update();
}

void CIGLWidget::mouseMoveEvent(QMouseEvent*e)
{
  tb.CalcRotMatrix( rot, e->x(), e->y());
   update();
}

void CIGLWidget::mouseReleaseEvent(QMouseEvent*e)
{
  tb.MouseUp( e->x(), e->y());
   update();
}
```


CITrackBall

```
void CIGLWidget::paintGL()
{
   glMatrixMode (GL_MODELVIEW);
   glClear(GL_COLOR_BUFFER_BIT |GL_DEPTH_BUFFER_BIT  );
   glLoadIdentity();
   gluLookAt(2,5,12,0,0,0,0,1,0);
   glColor3f(1,.4,.4);
   QString tmp;
   glMultMatrix(rot);
   ring->m.Draw();
}
```