Costruzione di Interfacce
Lezione 29
ancora MMeditor,

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

Cambiamo il mouse

- Qcursor
  - Classe che incapsula un cursore del mouse
  - Img standard (freccia, clessidra, ecc.)
  - Img custom
- QWidget.setCursor()
  - Ogni widget puo' decidere quale sia il cursore da far vedere quando il mouse attraversa lo spazio del widget

```
if(val) setCursor(QCursor(Qt::PointingHandCursor));
else setCursor(QCursor(Qt::SizeAllCursor));
```


zoom nella trackball

- si usa la rotella del mouse
- solito meccanismo si fa override di una funzione virtuale
- notare che il movimento della rotella e' espresso in 120-esimi
  - ad ogni scatto della rotellina corrisponde un delta di 120.
  - serve per futuri mouse con la rotella non a scatti
- `void CIGLWidget::wheelEvent(QWheelEvent*e)`

zoom

```
void CIGLWidget::paintGL()
{
  glClear(GL_COLOR_BUFFER_BIT |GL_DEPTH_BUFFER_BIT  );
  glMatrixMode (GL_MODELVIEW);
  glLoadIdentity();
  gluLookAt(2,5,6,0,0,0,0,1,0);
  glMultMatrix(rot);
  QTime tm;
  glScalef(zoomFactor,zoomFactor,zoomFactor);
  if(sg)     sg->drawGL(tm.elapsed()/1000.0f);
}
```



```
void CIGLWidget::wheelEvent(QWheelEvent*e)
{
  const int WHEEL_DELTA =120;
  if(!editing) {
      int notch = e->delta()/ WHEEL_DELTA;
      for(int i=0; i<abs(notch);++i)
           if(notch<0) zoomFactor*=1.2f;
                 else zoomFactor/=1.2f;
      update();
  }
}
```


picking

- per ora a noi interessa fare picking di una mattonella.
- usiamo gluunproject e facciamoci dare il punto 3d della superficie
- abbiamo bisogno
  - di salvarci le matrici al posto giusto
  - di leggere lo zbuf al momento giusto.
- aggiungiamo un po di campi a
  - CIGLWidget per ricordarsi che si vuole piccare
  - CIGrid per fare effettivamente il picking

Picking

- Quindi
  - Nella *mousePressEvent* Ci si segna il punto dello schermo dove fare il picking
  - Nella *paintGL* Si fa effettivamente il picking chiamando la *CIGrid::pickGL* DOPO aver disegnato tutto.
  - La *CIGrid::pickGL* determina la posizione del punto 3d cliccato e da quello capisce la cella selezionata.


```
void CIGLWidget::mousePressEvent(QMouseEvent*e)
{
  if(!editing) {
      tb.MouseDown( e->x(), e->y(), 0 );
      update();
  }
  else
  {
      pickPos=Point2i( e->x(), e->y());
      update();
  }
}
```



```
void CIGLWidget::paintGL()
{
   glMatrixMode (GL_MODELVIEW);
...
   QTime tm;
   glScalef(zoomFactor,zoomFactor,zoomFactor);
   if(sg)       sg->drawGL(tm.elapsed()/1000.0f);
   if(pickPos != Point2i(0,0)){
               Point2i out;
               bool ret=sg->g->pickGL(pickPos,out);
               if(ret) sg->g->G(out[0],out[1]).selected =! sg->g-
   >G(out[0],out[1]).selected;
               pickPos = Point2i(0,0);
   }
}
```


pickGL

```
bool CIGrid::pickGL(Point2i in, Point2i &pos){
   float z;
   glReadPixels(in[0],vp[3]-in[1],1,1,GL_DEPTH_COMPONENT
   ,GL_FLOAT,&z);
   return pickGL(Point3d(in[0],vp[3]-in[1],z),pos);
}
bool CIGrid::pickGL(Point3d in, Point2i &pos)
{
   Point3d o;
   gluUnProject(in[0],in[1],in[2],mm,mp,vp,&o[0],&o[1],&o[2]);
   qDebug("picked %f %f %f\n",o[0],o[1],o[2]);
        if(o[0]<0 || o[0]>sx ) return false;
        if(o[1]<0 || o[1]>sy ) return false;
        pos[0]=o[0];
        pos[1]=o[1];
        return true;
}
```


selezione

- per far vedere la selezione si cambia semplicemente il colore di base che viene blendato con la texture
- per rialzare gli elementi selezionati

```
void CIGrid::raiseSelected(float f)
{
 vector<CICell>::iterator ii;
 for (ii=g.begin();ii!=g.end();++ii)
        if((*ii).selected)
                for(int k=0;k<4;++k)
                        (*ii).V[k].z()+=f;
}
```


RubberBanding

- selezione di aree rettangolari
- necessita di un altro stato nel widget.
- ci si deve salvare il punto di inizio di inizio.
- e un po' di funz di appoggio nella grid
  - clear selection
  - select rect.

paint e rubberbanding

```
void CIGLWidget::paintGL()
...
   if(sg)       sg->drawGL(tm.elapsed()/1000.0f);
   if(rubberBanding && pickPos!=Point2i(0,0)){
                Point2i out;
                bool ret=sg->g->pickGL(pickPos,out);
                if(ret)
                {
                        if(rbStart==Point2i(-1,-1))
                                rbStart=out;
                        Point2i rbEnd= out;
                        sg->g->clearSelection();
                        sg->g->selectRect(rbStart,rbEnd);
                        pickPos = Point2i(0,0);
                }
   }
}
```


funzioni di editing

- Come prima cosa colleghiamo la wheel del mouse all'altezza dell'area selezionata
  - Aggiungiamo alla classe CIGrid un metodo per alzare la zona selezionata

```
void CIGrid::raiseSelected(float f)
{
   vector<CICell>::iterator ii;
   for (ii=g.begin();ii!=g.end();++ii)
    if((*ii).selected)
           for(int k=0;k<4;++k)
                  (*ii).V[k].z()+=f;
}
```

  - E invochiamo questo metodo direttamente nell'handler del evento mouse wheel


```
void CIGLWidget::wheelEvent(QWheelEvent*e)
{
   const int WHEEL_DELTA =120;
  int notch = e->delta()/ WHEEL_DELTA;
   if(!editing) {
        for(int i=0; i<abs(notch);++i)
                  if(notch<0) zoomFactor*=1.2f;
                                        else
   zoomFactor/=1.2f;
        update();
   }
   else
   {
        sg->g->raiseSelected(notch);
        update();
   }
}
```


Flatten

- Altre funzioni di editing: Flatten
  - la zona selezionata viene appiattita sull'altezza media.
- Aggiungiamo, nella classe cella,
  - Funzione per restituire l'altezza media della cella.

```
class CICell
{ public:
   bool selected;
   Point3f V[4];
   Point3f n;
   float tmpH[4];
   void ComputeNormal();
   void drawGL();
   float AvgH() {return (V[0][2]+V[1][2]+V[2][2]+V[3][2])/4.0f;}
}
```


Flatten

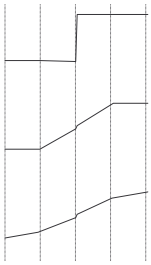
- Aggiungiamo
  - Uno slot flatten alla CIWidget che chiama
  - Una funzione flattenSelected alla classe CIGrid
  - Un pulsante sul tab
  - Un collegamento tra il release del pulsante e lo slot della widget
- La funzione flattenSelected fa due passate,
  - prima calcola l'altezza media.
  - Poi la setta sull'area selezionata

flatten

```
void CIGrid::flattenSelected()
{  // first step compute average height
   int i,j;
   int cnt=0;
   float avgH=0;
   for(i=0;i<sx;++i)
        for(j=0;j<sy;++j)
                if(G(i,j).selected)       {
                        avgH+=G(i,j).avgH();
                        ++cnt;
                }
   avgH/=cnt;    // second step apply the compute average height
   vector<CICell>::iterator ii;
   for (ii=g.begin();ii!=g.end();++ii)
        if((*ii).selected)
                for(int k=0;k<4;++k)
                                (*ii).V[k][2]=avgH;
}
```

Smooth

- Aggiungiamo una funzione che faccia uno smoothing della zona selezionata
  - Significa mediare ogni valore con gli adiacenti
  - Serve a far sparire gradini e fare pendi morbidi…
- Aggiungiamo alla classe cella,
  - campi temporanei per mantenere valori intermedi di altezze nelle celle



Smooth

- Ogni cella della griglia ha 4 vertici
  - Ad ogni vertice si assegna la media delle avgH() delle celle selezionate intorno a lui
  - Per ridurre il codice
    - esprimo ogni vertice di cella come offset rispetto alla cella stessa (eg il vert rosso in fig e' (1,1))
    - Esprimo gli adiacenti come offset rispetto a alla stessa

smooth

```
void CIGrid::smoothSelected()
{
    int bp[4][2]={{0,0},{1,0},{1,1},{0,1}}; // base position of each vertex of the cell
    int dp[4][2]={{-1,-1},{0,-1},{0,0},{-1,0}}; // delta of the four adj of a cell vertex
    int i,j,k,q;
    int cnt;
    float tH;
    for(i=0;i<sx;++i)
          for(j=0;j<sy;++j) if(G(i,j).selected)
                    for(k=0;k<4;++k) //for each cell vertex; {
                       tH=0;cnt=0;
                       for(q=0;q<4;++q) // foreach adj of a cell vertex
                              if(G(i+bp[k][0]+dp[q][0],j+bp[k][1]+dp[q][1]).selected){
                                  tH+=G(i+bp[k][0]+dp[q][0],j+bp[k][1]+dp[q][1]).avgH();
                                  ++cnt;
                              }
                              G(i,j).tmpH[k]=tH/cnt;
                       }
    for(i=0;i<sx;++i)
       for(j=0;j<sy;++j)      if(G(i,j).selected)
                    for(int k=0;k<4;++k) G(i,j).V[k][2]=G(i,j).tmpH[k];
}
```