UE MOBJ [4L103]

Jean-Paul Chaput
Jean-Paul.Chaput@lip6.fr

SESI

2018-2019

IX.2

```
class CellWidget : public QWidget {
    Q_OBJECT;
  public:
                      CellWidget          (QWidget* parent=NULL);
    virtual          ~CellWidget          ();
             void     setCell             (Cell*);
    inline   Cell*    getCell             () const;
    inline   QRect    boxToScreenRect     (const Box&) const;
    inline   QPoint   pointToScreenPoint(const Point&) const;
    inline   Box      screenRectToBox     (const QRect&) const;
    inline   Point    screenPointToPoint(const QPoint&) const;
    virtual QSize     minimumSizeHint     () const;
    virtual void      resizeEvent         (QResizeEvent*);
  protected:
    virtual void      paintEvent          (QPaintEvent*);
    virtual void      keyPressEvent       (QKeyEvent*);
  private:
    Cell* cell_;
    Box   viewport_;
};
```

IX.2

```
CellWidget::CellWidget(QWidget* parent)
  : QWidget  (parent)
  , cell_    (NULL)
  , viewport_(Box(0,0,500,500))
{
  setAttribute     (Qt::WA_OpaquePaintEvent);
  setAttribute     (Qt::WA_NoSystemBackground);
  setSizePolicy    (QSizePolicy::Expanding    // X direction.
                   ,QSizePolicy::Expanding);  // Y direction.
  setFocusPolicy   (Qt::StrongFocus);
  setMouseTracking(true);
}
```

IX.2

```
QSize  CellWidget::minimumSizeHint() const
{ return QSize(500,500); }

void  CellWidget::resizeEvent(QResizeEvent* event) {
  const QSize& size = event->size();

// Assume the resize is always done by drawing the bottom right corner.
  viewport_.setX2( viewport_.getX1() + size.width () );
  viewport_.setY1( viewport_.getY2() - size.height() );

  cerr << "CellWidget::resizeEvent()␣viewport_:" << viewport_ << endl;
}
```

IX.2

```
void  CellWidget::keyPressEvent(QKeyEvent* event) {
  event->ignore();
  if (event->modifiers() & (Qt::ControlModifier|Qt::ShiftModifier))
    return;

  switch( event->key()) {
    case Qt::Key_Up:    goUp    (); break;
    case Qt::Key_Down:  goDown  (); break;
    case Qt::Key_Left:  goLeft  (); break;
    case Qt::Key_Right: goRight (); break;
    default: return;
  }
  event->accept();
}
```

IX.2

```
void  CellWidget::goRight() {
  viewport_.translate( Point(20,0) );
  repaint();
}

void  CellWidget::goUp() {
  viewport_.translate( Point(0,20) );
  repaint();
}
```

IX.3

```
void  CellWidget::paintEvent (QPaintEvent* event) {
  QPainter painter(this);
  painter.setBackground( QBrush( Qt::black ) );
  painter.eraseRect    ( QRect( QPoint(0,0), size() ) );

  painter.setPen( QPen( Qt::darkGreen, 1 ) );
  QRect rect = boxToScreenRect(box);
  painter.drawRect( rect1 );

  painter.setPen  ( QPen  ( Qt::red, 0 ) );
  painter.setBrush( QBrush( Qt::red ) );
  // ...
  painter.drawRect( rect2 );
}
```

IX.3

```
void  CellWidget::query(unsigned int flags, QPainter& painter) {
  if ((not cell_) or (not flags)) return;

  const vector<Instance*>& instances = cell->getInstances();
  for (size_t i; i<instances.size() ; ++i) {
    Point         instPos = instances[i]->getPosition();
    const Symbol* symbol  = instances[i]->getMasterCell()->getSymbol();
    if (not symbol) continue;

    if (flags & InstanceShapes) {
      const vector<Shape*>& symbol->getShapes();
      for (size_t j=0 ; j<=shapes.size() ; ++j) {
        BoxShape* boxShape = dynamic_cast<BoxShape*>(shapes[j]);
        if (boxShape) {
          Box   box  = boxShape->getBoundingBox();
          QRect rect = boxToScreenRect(box.translate(instPos));
          painter.drawRect(rect);
        }
      }
    }
```

IX.3

```
void  CellWidget::setCell(Cell* cell) {
  cell_ = cell;
  repaint();
}
```