2D Curves

Screen Coordinates (where mouse clicks happen) $\xrightarrow{\text{gluUnProject}}$ World Coordinates (where you draw → glVertex)

$\xleftarrow{\text{gluProject}}$ * describe this

gluProject(x, y, z, mv, p, vp, &sx, &sy, &sz)

3 Modes → Add, move, delete

- Add is easy with gluUnProject
- for move & delete we need to be able to select points.

How:

1. Unproject mouse click. The closest control point to this point is the selected control point.
   - Why is this not ideal?
     → Better if you can have a distance threshhold

```
if ( closest && distance < ? )
        select Point.
```

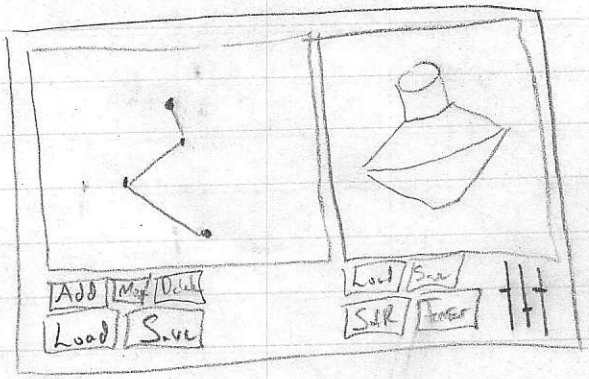
2. Project all control points to screen coordinates & find distances. Distances are now in pixels. Projection is not very costly since # of control points is fairly small.

```
best dist = 1e10000;
for   i = 0 to numpoints-1
begin     p := Project ( points [i] );
          if ( distanceSquared (mousepoint, p) < bestdist)
                bestindex = i
                best dist = distanceSquared (mousepoint, p)
end for
if (best dist < 7)
      selected Index = bestindex;
```

Assignment 2

Ideally use QT. Why?

1. Multiple windows easier
2. File Dialogs for saving/opening files



If you haven't, do my QT tutorial!

Differences between QT & glut

→ OpenGL code no longer begins with main. Instead you create a class descended from GLWidget.

→ no more callback functions. Now the OpenGL functions have standard names (determined by GLWidget).

- initializeGL(); → called after GLWidget constructor & the GL environment is initiated. Occurs before screen is first drawn.
- resizeGL(int width, int height);
- paintGL();
- mousePressEvent(QMouseEvent *e)
- mouseReleaseEvent(QMouseEvent *e)
- mouseMoveEvent(QMouseEvent *e)
- mouseDoubleClickEvent ...
- keyPressEvent(QKeyEvent *e)
- keyReleaseEvent(QKeyEvent *e)

- glutPostRedisplay() → updateGL();
- makeCurrent(); → important when using multiple windows.
- no need for glFlush();

Programs
- assistant
- designer
- qmake

→ be sure to use "make clean" somewhat often

Signals & Slots
→ basically it's a way of linking events & event handlers
→ tutorial shows you how to link a signal & a slot in designer
→ writing your own.

```
class blah {
    public:
        ____
    public slots:
        setPointMode (int drawMode);
                                ↳ define this in body!
    signals:
        sendCurve (vector<Point> controlPoints);
    private:
        ____
};
```

→ to trigger a signal
emit sendCurve(points);
→ can have more than one slot receive the same signal.

- example from assignment p.2