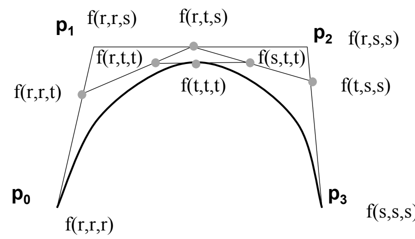
Drawing a Bézier curve

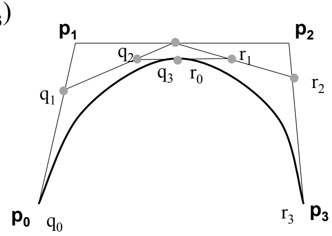
A 3[rd] degree Bézier curve

- p_0, p_1, p_2, p_3 = 4 original control points



Recursive interpolation

- From (p_0, p_1, p_2, p_3), we deduce 2 sets of control points: (q_0, q_1, q_2, q_3) and (r_0, r_1, r_2, r_3)



Algorithm

```
void bezier(Point p[])
{
    Point q[], r[];
    if(colinear(p)) {
      draw_line(p0,p3)
    } else {
      /* split p into q and r */
      split(p,q,r);
      bezier(q);
      bezier(r);
    }
}
```


Colinear

- Colinear checks if the 4 points p_0, p_1, p_2, p_3 are aligned
- Of course, you don't get the alignment easily (numerical problems, degree of accuracy, etc…)
- We need to have an approximation
- Return true if the 4 points are approximately on a straight line

Colinear

- If (p_0,p_3) is given by $ax+by+c = 0$,
- Then compute the distance of p_1 and p_2 from this line
- The distance should be within a certain threshold
- If $D^2(x,y) = (ax+by+c)^2/(a^2+b^2)$, and $D^2(P_1)<\varepsilon$ and $D^2(p_2)$)$<\varepsilon$ than the points are colinear

Split

- We need to compute R and Q from P
- Where
- q_0 = f(r,r,r), q_1=f(r,r,t), q_2=f(r,t,t), q_3=f(t,t,t)
- r_0 = f(t,t,t), r_1=f(t,t,s), r_2=f(t,s,s), r_3=f(s,s,s)
- You can split how ever you like, but it's convenient to split in 2 … using
  t=1/2