Computer Graphics & Image Processing: supplementary sheet

Bringing together everything we have learnt so far in the 2D part of the course, we address this question: how could we clip a Bezier cubic against a rectangle?

Here are three alternative methods.

❶ Don't bother, just use the Bezier drawing algorithm along with a line drawing algorithm which does clipping.

The problem with this is that it is inefficient: the clipping is done for every single line that is drawn. It would be more efficient to do the clipping at a higher level (i.e. at the level of the Bezier control points rather than the level of the many individual lines) and to use an unclipped line drawing algorithm used wherever possible.

❷ Use the fact that a Bezier cubic is guaranteed to lie inside its convex hull. This leads to a modified version of the line clipping algorithm where you need to test all four of the Bezier control points, to produce Q_0, Q_1, Q_2, Q_3. This needs sixteen comparison operations.

There are three cases to consider:

1. If $Q_0=Q_1=Q_2=Q_3=0$, then accept the whole Bezier and use the standard Bezier drawing algorithm, with no clipping tests, with unclipped line drawing.
2. If $Q_0 \wedge Q_1 \wedge Q_2 \wedge Q_3 \neq 0$, then reject the whole Bezier and there is nothing to draw.
3. Otherwise: if the line from P_0 to P_3 is sufficiently flat, draw that line with a clipped line drawing algorithm, else subdivide the Bezier and recurse on the two halves.

Case 1 requires four comparison operations. Case 2 requires three boolean operations and one comparison.

❸ Approximate the convex hull by the bounding rectangle of the Bezier. This requires sixteen comparison operations to get the bounding rectangle. Then compare the bounding rectangle of the Bezier against the clipping rectangle. There are three cases to consider:

1. Bounding rectangle completely inside clipping rectangle: accept the whole Bezier and use the standard Bezier drawing algorithm, with no clipping tests, with unclipped line drawing.
2. Bounding rectangle completely outside clipping rectangle: reject the whole Bezier and there is nothing to draw.
3. Otherwise: if the line from P_0 to P_3 is sufficiently flat, draw that line with a clipped line drawing algorithm, else subdivide the Bezier and recurse on the two halves.

Case 1 requires four comparison operations. Case 2 requires four more comparisons.

Methods ❷ and ❸ require that we implement software with two versions of the Bezier drawing algorithm and two versions of the line drawing algorithm. In each case there will be a version which does clipping and a version which does not. This extra code is the price we pay for a faster algorithm.