SSX curved patches formula information

SSX and SSX Tricky use Bi- cubic Bézier surfaces for the curved surfaces used in the maps. See https://web.archive.org/web/20021105121623id_/http://www.gamasutra.com:80/gdc2002/features/rayner/rayner_pfv.htm for details.

The formula for calculating a Bi-cubic Bézier patch is as follows:

$$\boldsymbol{Q}(\boldsymbol{u},\boldsymbol{v}) = \begin{bmatrix}\boldsymbol{u}^3\boldsymbol{u}^2\boldsymbol{u},\mathbf{1}\end{bmatrix} \begin{vmatrix}-\mathbf{1} & \mathbf{3} & -\mathbf{3} & \mathbf{1}\\ \mathbf{3} & -\mathbf{6} & \mathbf{3} & \mathbf{0}\\ -\mathbf{3} & \mathbf{3} & \mathbf{0} & \mathbf{0}\\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{vmatrix} \begin{vmatrix}\boldsymbol{P00} & \boldsymbol{P01} & \boldsymbol{P02} & \boldsymbol{P03}\\ \boldsymbol{P10} & \boldsymbol{P11} & \boldsymbol{P12} & \boldsymbol{P13}\\ \boldsymbol{P20} & \boldsymbol{P21} & \boldsymbol{P22} & \boldsymbol{P23}\\ \boldsymbol{P30} & \boldsymbol{P31} & \boldsymbol{P32} & \boldsymbol{P33}\end{vmatrix} \begin{vmatrix}-\mathbf{1} & \mathbf{3} & -\mathbf{3} & \mathbf{1}\\ \mathbf{3} & -\mathbf{6} & \mathbf{3} & \mathbf{0}\\ -\mathbf{3} & \mathbf{3} & \mathbf{0} & \mathbf{0}\\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{vmatrix} \begin{vmatrix}\boldsymbol{v}^3\\ \boldsymbol{v}^2\\ \boldsymbol{v}\\ \mathbf{1}\end{vmatrix}$$

SSX uses 8x8 patches when the surfaces are close to the user so u and v take on 8 evenly spaced values from 0.0 to 1.0 when calculating a patch. P00 to P33 are the patch control points.

However a major point to realise about the SSX and SSX Tricky maps is they pre-multiple the control points and store the pre-multiplied control points in the map files:

$$\begin{vmatrix}\mathbf{C00} & \mathbf{C01} & \mathbf{C02} & \mathbf{C03}\\ \mathbf{C10} & \mathbf{C11} & \mathbf{C12} & \mathbf{C13}\\ \mathbf{C20} & \mathbf{C21} & \mathbf{C22} & \mathbf{C23}\\ \mathbf{C30} & \mathbf{C31} & \mathbf{C32} & \mathbf{C33}\end{vmatrix} = \begin{vmatrix}-\mathbf{1} & \mathbf{3} & -\mathbf{3} & \mathbf{1}\\ \mathbf{3} & -\mathbf{6} & \mathbf{3} & \mathbf{0}\\ -\mathbf{3} & \mathbf{3} & \mathbf{0} & \mathbf{0}\\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{vmatrix} \begin{vmatrix}\boldsymbol{P00} & \boldsymbol{P01} & \boldsymbol{P02} & \boldsymbol{P03}\\ \boldsymbol{P10} & \boldsymbol{P11} & \boldsymbol{P12} & \boldsymbol{P13}\\ \boldsymbol{P20} & \boldsymbol{P21} & \boldsymbol{P22} & \boldsymbol{P23}\\ \boldsymbol{P30} & \boldsymbol{P31} & \boldsymbol{P32} & \boldsymbol{P33}\end{vmatrix} \begin{vmatrix}-\mathbf{1} & \mathbf{3} & -\mathbf{3} & \mathbf{1}\\ \mathbf{3} & -\mathbf{6} & \mathbf{3} & \mathbf{0}\\ -\mathbf{3} & \mathbf{3} & \mathbf{0} & \mathbf{0}\\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{vmatrix}$$

Then the formula to generate a patch becomes:

$$\boldsymbol{Q}(\boldsymbol{u},\boldsymbol{v}) = \begin{bmatrix}\boldsymbol{u}^3\boldsymbol{u}^2\boldsymbol{u},\mathbf{1}\end{bmatrix} \begin{vmatrix}\mathbf{C00} & \mathbf{C01} & \mathbf{C02} & \mathbf{C03}\\ \mathbf{C10} & \mathbf{C11} & \mathbf{C12} & \mathbf{C13}\\ \mathbf{C20} & \mathbf{C21} & \mathbf{C22} & \mathbf{C23}\\ \mathbf{C30} & \mathbf{C31} & \mathbf{C32} & \mathbf{C33}\end{vmatrix} \begin{vmatrix}\boldsymbol{v}^3\\ \boldsymbol{v}^2\\ \boldsymbol{v}\\ \mathbf{1}\end{vmatrix}$$

Where C00 to C33 are the 16 pre-multiplied control points stored in the map files in SSX and SSX tricky.

In **BezierSurface.py**:

CalcBezierUsingMatrixMethod(Points, isPremultiplied, numSegments) – if isPremultiplied=True will return the Bezier surface from a pre-multiplied set of control points.

Also useful is **ReversePrecomputedBezier**(PreComputedPoints) – which will calculate the normal Bezier control points from the pre-multiplied control points which then could be used in a program that accepts normal Bezier control surfaces (or a map modder if someone makes one)