PROJECT’S DETAIL

Let’s go into more detailed of the method and technique required for creating the Free Form Deformation. Once the program has determined the current location of the shark, a function will be called to form the local space, namely the P zero and the vectors S,T,U mentioned in the introduction. Vector S was created by subtracting vertex at P_zero from the vertex at S-location (lower, right-most corner), and so forth for the T and U vectors. Online documentations use U, V, W to represent S, T, U respectively. Once the frame work above is obtained, the program calls another function to obtain the local “s,t,u” coordinates (not vectors) of the shark using the formula below. Again, online documentations refer to these (s,t,u) coordinates as (u,w,v) coordinates respectively.

$$u = \frac{(\vec{v} \times \vec{w}) \cdot (\mathbf{P} - \mathbf{O})}{(\vec{v} \times \vec{w}) \cdot \vec{u}}$$

$$v = \frac{(\vec{u} \times \vec{w}) \cdot (\mathbf{P} - \mathbf{O})}{(\vec{u} \times \vec{w}) \cdot \vec{v}}$$

$$w = \frac{(\vec{u} \times \vec{v}) \cdot (\mathbf{P} - \mathbf{O})}{(\vec{u} \times \vec{v}) \cdot \vec{w}}$$

The cross product of the U,V,W was self-explanatory, and $\mathbf{P}$ is representation of a vertex of the shark, thus this formula iterates through every single vertex of the shark and creates as many (s,t,u) coordinates as the shark’s vertices. $\mathbf{O}$ is the representation for P_zero. The shark has the total of 2560 (x,y,z) coordinates thus the function above will form 2560 (s,t,u) coordinates corresponding to each shark’s vertex. This calculation considered vital since it introduces a set of local coordinates to the shark along with the existing global (x,y,z) coordinates. Once this set of local coordinates is determined, each vertex ($\mathbf{P}$) on the shark is recalculated using the this formula…

$$\mathbf{P} = \mathbf{O} + u\vec{u} + v\vec{v} + w\vec{w}$$

If no transformation is yet applied to the control points, these vertices of the shark are identical to the original vertices. So far, the control points have not been touched in order to maintain its S,T,U vector, however once the steps above are performed, the program needs to determine the position of the 64 control points (4x4x4) once again so that they are all relative to the U,V,W vectors (in this case S,T,U) in case those get transformed. This step is required in order to preserve the volume of the box, in turn preserving the volume of the shark.

$$\mathbf{F}^{pp}_{i,j,k} = \mathbf{O} + \frac{i}{l}\vec{u} + \frac{j}{m}\vec{v} + \frac{k}{n}\vec{w}$$

The formula above uses i,j,k as indices of the control points where 0 < i,j,k < l,m,n (4x4x4)

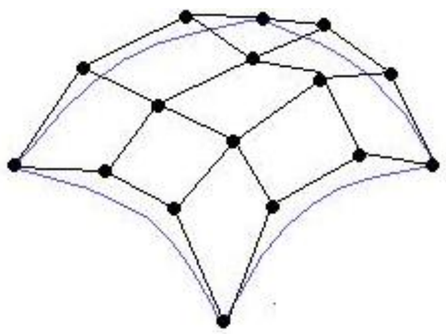
For every global transformation on the control points, $\mathbf{F}^{pp}_{i,j,k}$ (coordinates of the control points) will be recalculated thus after every frame, the function which calculating this gets call after the call the transformation. The program will, based on all the values acquired above, transform all 2560 vertices on the shark using the *trivariate Bézier function* to accommodate the changes in the position of the control points. In another word, the program will iterate through the whole set of shark's coordinates $\mathbf{P}$ and reconstruct them using the formula below…

$$\mathbf{F}(u,v,w) = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n}\mathbf{F}_{i,j,k}B_i^l(u)B_j^m(v)B_k^n(w)$$

The formula utilize the *Bézier curve* $B_k^n(w)$ to obtain the deformation of $\mathbf{P}$. The amount of deformation to be applied to the shark is represented by $\mathbf{F}(u,v,w)$ thus the new position $\mathbf{P}$ of the shark is the sum of itself and the difference between $\mathbf{F}(u,v,w)$ and $\mathbf{P}$. The formula to calculate the *Bézier curve* uses iteration on each point of this curve to create a new formation (in the case of 2D line).

$$Q(u) = \sum_{i=0}^{n} p_i B_{i,n}(u) \quad \text{------->} \quad B_{i,n}(u) = {}^nC_i u^i (1-u)^{n-i} \quad \text{------->} \quad {}^nC_i = \frac{n!}{i!(n-i)!}$$

However the shark is in form of a 3D model thus it uses the Tricubic Bezier Hyperpatch form of the Bézier *curve.* Once above calculation done, the program render the shark with the new value of $\mathbf{P}$, and this will have the affect of transforming the shark as the control points lying on the cubic frame are moving.



$$\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} p_{ijk} B_i(u) B_j(v) B_k(w)$$

Tricubic Bezier Patch