Tentacle - Requirements & Design

Requirements:

With this tool the user can create a line of points representing a tentacle, those can be used later to build a curve for example, and deform stuff along that curve, the scope of this tool is creating only the points though.

With some parameters the user can deform those points in order to make it look like a tentacle, with waves, swirls and that kind of stuff.

The user can define also other parameters, like the size and number of points.

Design General Description:

The tool will create several points aligned in the x axis, every point will inherit the transform of the previous point. So if one point rotates the next point will inherit that rotation (like a parenting). Every point will rotate using a sin function controlled by amplitude, frequency and offset values. Amplitude and offset can be controlled by a ramp as well to achieve interesting results.

Inputs:

- **Length (float):**
  This control the length of the tentacle along the x axis

- **Number of Points (integer)**
  The number of points along the tentacle, the more points you have the better the final result (more expensive too).

- **Amplitude (float)**
  Amplitude of the movement of the tentacle

- **Amplitude Ramp (float ramp)**
  Amplitude of the movement of the tentacle along its length

- **Offset (float)**
  Offset of the movement of the tentacle

- **Offset Ramp (float ramp)**
  Offset of the movement of the tentacle along its length

- **Frequency (float)**
  Frequency of the movement of the tentacle

- **Distance (float)**
  Distance traveled by the movement of the tentacle, the user can wire the Time here to make a procedural animation.

Process Steps:

- For every point to be created:
  - a transformation matrix is created (TM), it's used to transform the point,
  - a sin function is used to calculate a rotation value, the sin uses the *distance* input which is multiplied by the *frequency*, the *offset* is added too, that offset is multiplied by the *offset ramp*.
  - the value returned by the sin is multiplied by the *amount* input and the *amount* ramp.
  - Now the TM is rotated using the rotation value.
  - the TM position will be determined by the distance between the points (this is a local TM), only the first point will have zero position. The distance between the points is calculated using the *length* and *number of points* inputs.
  - the TM is multiplied by the TM of the previous point, if this is the first point the TM will be just the default matrix (TM=1).
  - the TM is used to create a point (with position, normal and upvector attributes).