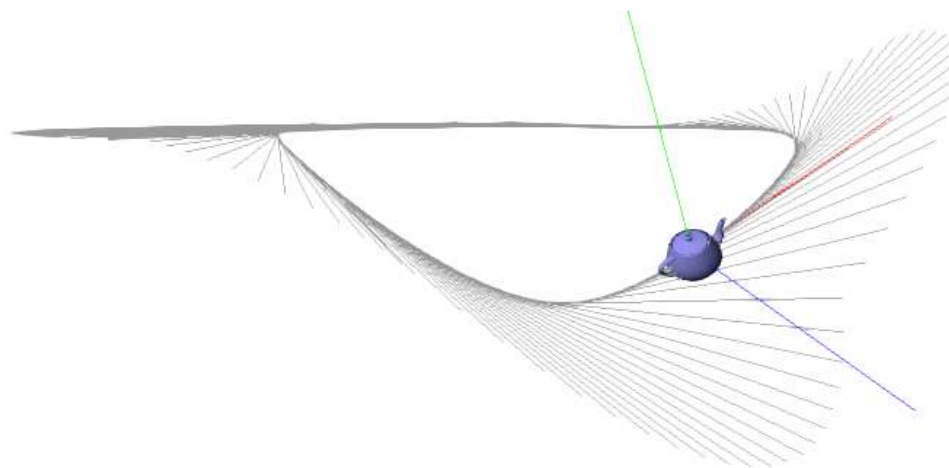




Animation with Catmull-Rom Curves

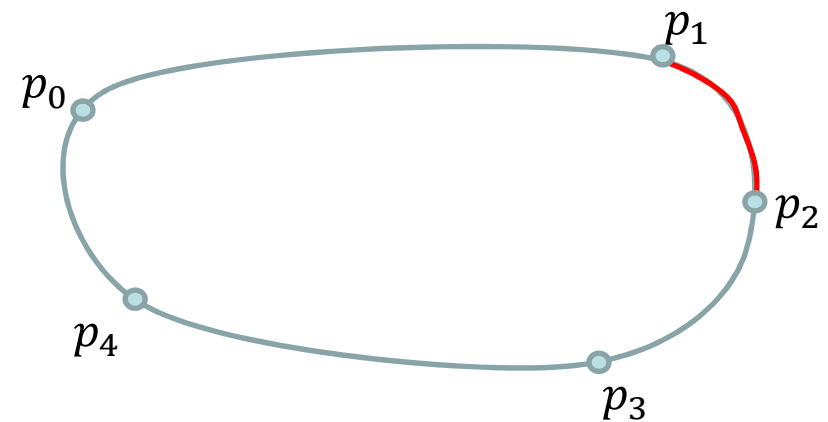




Cubic Curves – Catmull-Rom

- Matrix formulation

- $$P(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$
- $$P'(t) = [3t^2 \quad 2t \quad 1 \quad 0] \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

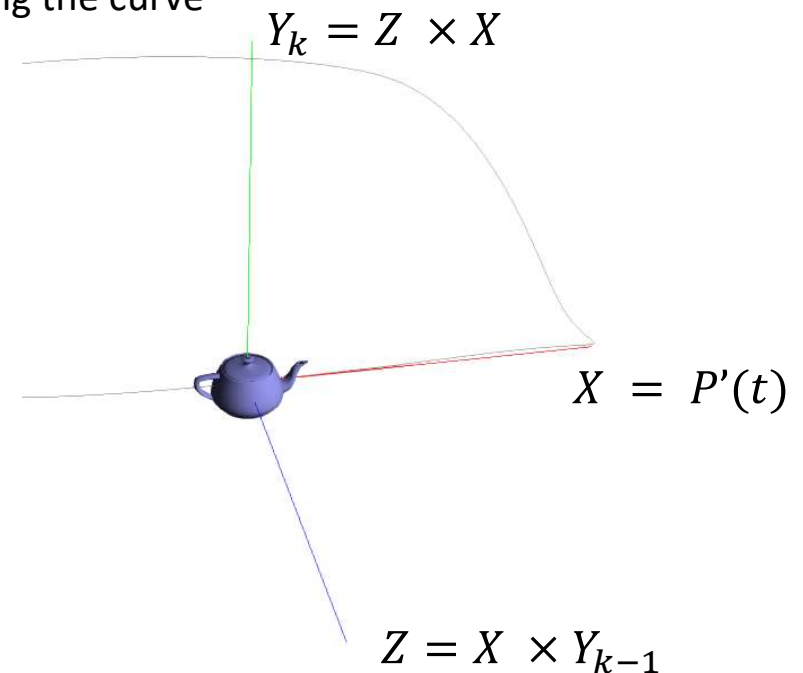




Cubic Curves – Catmull-Rom

- Axis for Rotation Matrix
 - Available data at instant t
 - $P(t)$ - position of an object “walking” along the curve
 - $P'(t)$ - vector tangent to the curve

- Transform for teapot
 - Translation to place teapot
 - Rotation to align with curve
- $Y_0 = (0,1,0)$





Cubic Curves – Catmull-Rom

- Assuming an initial specification of an \vec{Y}_0 vector, to align the object with the curve, we need to build a rotation matrix for the object:

$$\begin{aligned}\vec{X}_i &= P'(t) \\ \vec{Z}_i &= X_i \times \vec{Y}_{i-1} \\ \vec{Y}_i &= \vec{Z} \times \vec{X}\end{aligned} \quad M = \begin{bmatrix} X_x & Y_x & Z_x & 0 \\ X_y & Y_y & Z_y & 0 \\ X_z & Y_z & Z_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note: All vectors need to be normalized

```
glMultMatrixf(float *m)
```

- Current OpenGL MODEL_VIEW matrix gets multiplied by m

Note: OpenGL matrices are column major => compute the transpose instead



Assignment

- Complete the function

```
void getCatmullRomPoint(float t,
                        float *p0, float *p1, float *p2, float *p3,
                        float *pos, float *deriv) {

    // catmull-rom matrix
    float m[4][4] = { {-0.5f, 1.5f, -1.5f, 0.5f},
                      { 1.0f, -2.5f, 2.0f, -0.5f},
                      {-0.5f, 0.0f, 0.5f, 0.0f},
                      { 0.0f, 1.0f, 0.0f, 0.0f}};

    // Compute A = M * P
    // for component x P is the vector (p0[0], p1[0], p2[0], p3[0])
    // Compute pos = T * A
    // compute deriv = T' * A
    // ...
}
```



Assignment

- Write the function

```
void renderCatmullRomCurve() {  
  
    // draw the curve using line segments - GL_LINE_LOOP  
}
```

To get the points for the full curve call

```
void getGlobalCatmullRomPoint(float gt, float *pos, float *deriv)
```

with gt in $[0,1[$.

- Apply the required transformations to have the teapot travelling along the curve oriented accordingly to the derivative.
 - Use `buildRotMatrix` provided in the source code



Questions

- Replace the teapot with the cone from GLUT.
 - What happens?
 - How to fix it?

