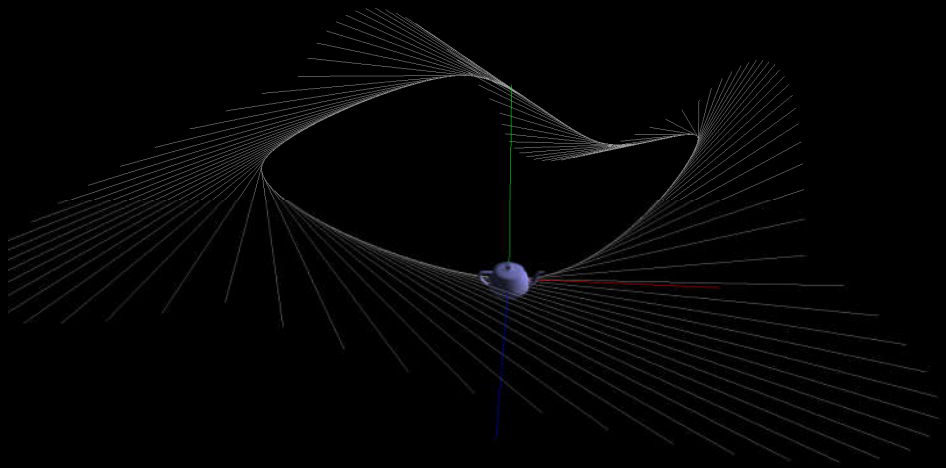


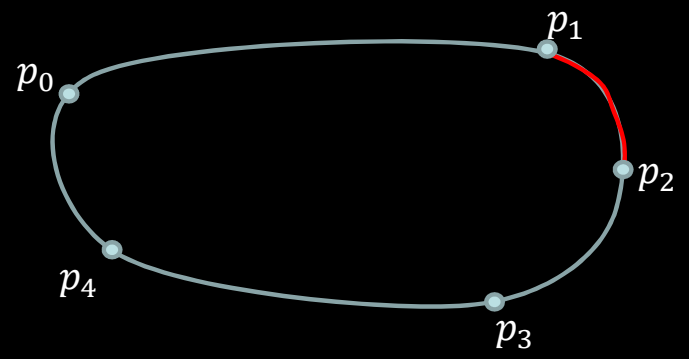
LEI / LCC
Departamento de Informática
Universidade do Minho

# Animation with Catmull-Rom Curves



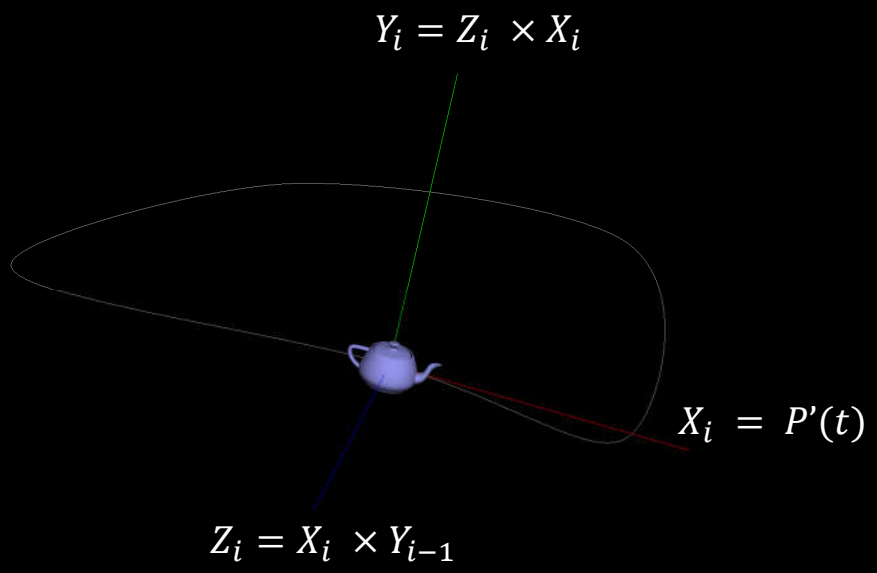
# Cubic Curves – Catmull-Rom

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



# 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 $\overrightarrow{Y_0}$ vector, to align the object with the curve, we need to build a rotation matrix for the object:

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

*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}};

    // For each component i: // x, y, z
    //     Compute vector A = M * P // use function multMatrixVector
    //         in component i P is the vector (p0[i], p1[i], p2[i],p3[i]
    //     Compute pos[i] = T * A
    //     compute deriv[i] = T' * A
}
```

# Assignment – cont.

- Write the function

```
void renderCatmullRomCurve() {

// draw the curve using line segments - GL_LINE_LOOP (see slides near the end)
}
```

– To get the points for the full curve call

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

with `gt` in [0,1[.

– To draw the curve select a tessellation level, for instance 0.01. This will get a curve with 100 line segments.

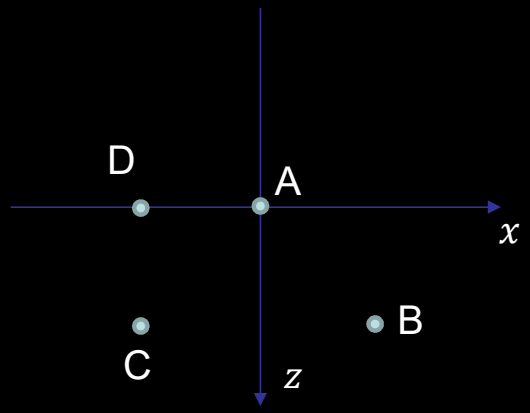
# Assignment – cont.

- In function `renderScene`, apply the required transformations to have the teapot travelling along the curve oriented accordingly to the derivative.
  - Get the position and derivative in the curve for current `t` value - `getGlobalCatmullRomPoint(t,pos, deriv)`
  - Use `glTranslate` to position the teapot along the curve
  - Build the rotation matrix to align the teapot with the curve
    - Compute the axis
      - functions, `cross` and `normalize`, are provided in the code.
    - Use `buildRotMatrix` provided in the source code
  - Apply the matrix using `glMultMatrix`

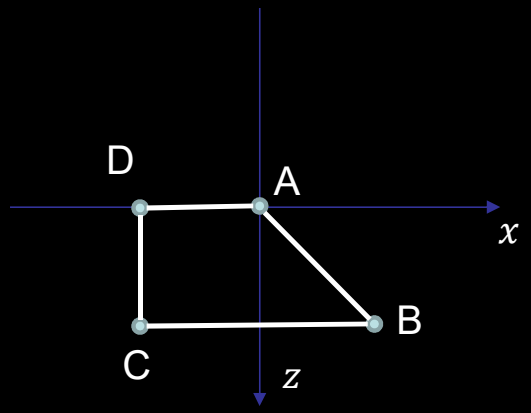
# Drawing with line loops

Consider the following points:

- A = (0,0,0), B = (1,0,1), C = (-1,0,1), D = (-1,0,0)




A line loop receives all the points and draws a line between each two points and connects the last to the first point:



# Drawing with line loops

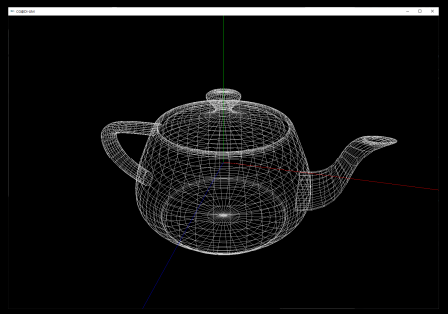
- Line loops in OpenGL

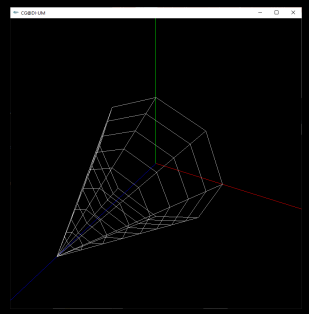
```
float p[4][3] = {{ 0.0f,  0.0f, 0.0f},   // A
                 { 1.0f,  0.0f, 1.0f},   // B
                 {-1.0f,  0.0f, 1.0f},   // C
                 {-1.0f,  0.0f, 0.0f}};  // D

glBegin(GL_LINE_LOOP);
for (int i = 0; i < 4; ++i) {
    glVertex3f(p[i][0], p[i][1], p[i][2]);
}
glEnd();
```

# Questions

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





- What would be required to use Bezier or Hermite curves?
  - What parts of the code would require change?