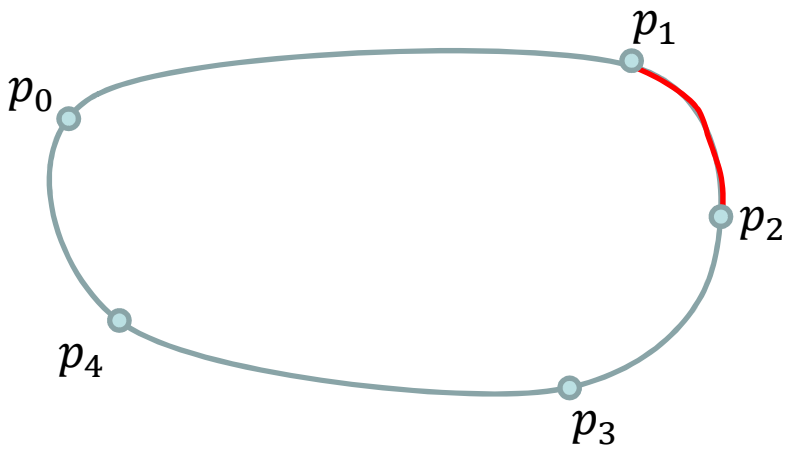
LEI / LCC
Departamento de Informática
Universidade do Minho

Animation with Catmull-Rom Curves

Cubic Curves – Catmull-Rom

- Matrix formulation
- $$P(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_o \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$



Cubic Curves – Catmull-Rom

- P(t) provides the position of an object "walking" along the curve
- P'(t) provides a vector tangent to the curve.
- Assuming an initial specification of an $\overrightarrow{up}$ vector, to place and align the object with the curve, we need to build a transformation matrix for the object:

$$\vec{d} = P'(t)$$
$$\overrightarrow{left} = \overrightarrow{up} \times \vec{d}$$
$$\overrightarrow{up} = \vec{d} \times \overrightarrow{left}$$

$$M = \begin{bmatrix} left_x & up_x & d_x & p_x \\ left_y & up_y & d_y & p_y \\ left_z & up_z & d_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

`glMultMatrix(float *m)`

- Current matrix gets multiplied by `m`

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

Assignment

- Complete the function

```
void getCatmullRomPoint(float t, int *indices, float *res) {

// 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}};

res[0] = 0.0; res[1] = 0.0; res[2] = 0.0;
// Compute point res = T * M * P
// where Pi = p[indices[i]]
// …
}
```

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 *res)
```

with `gt` in [0,1[.

- Apply the required transformations to have the teapot travelling along the curve oriented accordingly to the derivative.