Assignment in Computer Graphics II

– Assignment 2 –

Computer Graphics and Multimedia Systems Group

David Bulczak, Christoph Schikora

Assignment 1 [2 Points] Interpolation with squared polynomials

Given polynomials:

$$f_0(u) = 2u^2 - 3u + 1, \quad f_{\frac{1}{2}}(u) = -4u^2 + 4u, \quad f_1(u) = 2u^2 - u$$

and the defintion of a curve:

$$\mathbf{P}(u) = f_0(u)\mathbf{P}_0 + f_{\frac{1}{2}}(u)\mathbf{P}_{\frac{1}{2}} + f_1(u)\mathbf{P}_1$$

Show that $P(u)$ has following interpolation properties:

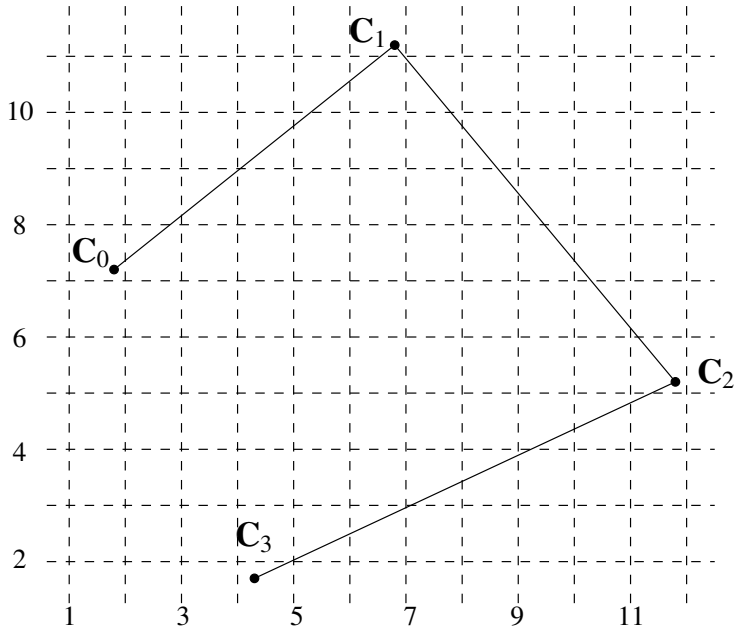
$$\mathbf{P}(0) = \mathbf{P}_0, \quad \mathbf{P}(\tfrac{1}{2}) = \mathbf{P}_{\frac{1}{2}}, \quad \mathbf{P}(1) = \mathbf{P}_1$$

Assignment 2 [2 Points] Example de Casteljau-Algorithm

Evaluate the cubic Bézier-curve with control points

$$\mathbf{C}_0 = \begin{pmatrix} 1.8 \\ 7.2 \end{pmatrix} \quad , \quad \mathbf{C}_1 = \begin{pmatrix} 6.8 \\ 11.2 \end{pmatrix} \quad , \quad \mathbf{C}_2 = \begin{pmatrix} 11.8 \\ 5.2 \end{pmatrix} \quad , \quad \mathbf{C}_3 = \begin{pmatrix} 4.3 \\ 1.7 \end{pmatrix}$$

graphically and mathematically with the **de Casteljau-Algorithm** for $u = 0.4$! Denote all the points!



Assignment 3 [2 Points] Polynomial curves & Bezier curves

In this task we introduce a curve framework that supports several types of curves presented in the lecture. You will begin with an initial curve type and extend this tool during the next weeks. Download the framework `curve-framework.zip` and take an initial look on the code.

All relevant files for this and future programming tasks, related to curves, can be found in the `Curves` folder. In `Curve/Curve.hpp` you can find the abstract base class for all further curve classes. It provides three abstract member functions `eval, evalCurve, evalConstruct` which you will have to implement for all derived classes at least. Please study this class, read the comments and try to understand it.

To build the project in your preferred development environment use the included CMake project ("CMakeLists.txt"). CMake can be downloaded from the following website: http://www.cmake.org/. Use the instructions on the page
http://www.cmake.org/cmake/help/runningcmake.html and the tutorial page to create the project.

1. Implement the polynomial curve type in `Curve/PolynomialCurve.hpp` and `Curve/PolynomialCurve.cpp`. This curve should be evaluated by using the straight forward evaluation using the monomial basis $B = \{1, x, x^2, x^3, \ldots\}$.
   - `eval`: Computes the curve value for parameter u.
   - `evalConstruct`: Can be ignored for this curve type.
   - `evalCurve`: In this function the curve has to be evaluated for the whole interval $[0, 1]$.
2. Implement Bézier-Curves in `Curve/BezierCurve.hpp` and `Curve/BezierCurve.cpp`. This curve should be evaluated by using Bernstein Basis polynomials and by using the de Casteljau algorithm.
   - `eval`: Computes curve value for given parameter by evaluating the de Casteljau algorithm.
   - `evalBernstein`: Computes curve value for a given parameter by evaluating Bernstein polynomials.
   - `calculateBernstein`: Evaluates a Bernstein basis polynomial.
   - `evalConstruct`: Evaluates curve by the de Casteljau algorithm and stores intermediate steps.
   - `evalCurve`: In this function the curve has to be evaluated for the whole interval $[0, 1]$.

Further explanations can be found in the comments of the code.

Hand in: 02.11.2015, at beginning of the lecture or until 10:00 in the mailbox of the chair (next to room H-A 7107) and send files corresponding to the programming task to johnfr93@gmail.com.