Assignment in Computer Graphics II
– Assignment 5 –

Computer Graphics and
Multimedia Systems Group
David Bulczak, Christoph Schikora

Assignment 1 [2 Points] Evaluation of TP-Bézier-surfaces

Given the bi-quadratic Bézier-surface with control points

$$\mathbf{C}_{00} = \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix}, \quad \mathbf{C}_{10} = \begin{pmatrix} 4 \\ 2 \\ 6 \end{pmatrix}, \quad \mathbf{C}_{20} = \begin{pmatrix} 12 \\ 2 \\ 2 \end{pmatrix}, \quad \mathbf{C}_{01} = \begin{pmatrix} 0 \\ 6 \\ 6 \end{pmatrix}, \quad \mathbf{C}_{11} = \begin{pmatrix} 4 \\ 6 \\ 10 \end{pmatrix}$$

$$\mathbf{C}_{21} = \begin{pmatrix} 12 \\ 6 \\ 6 \end{pmatrix}, \quad \mathbf{C}_{02} = \begin{pmatrix} 0 \\ 14 \\ 2 \end{pmatrix}, \quad \mathbf{C}_{12} = \begin{pmatrix} 4 \\ 14 \\ 6 \end{pmatrix}, \quad \mathbf{C}_{22} = \begin{pmatrix} 12 \\ 14 \\ 2 \end{pmatrix}$$

1. Evaluate twice the surface using the two-stage de Casteljau algorithm for the parameters (0.5,0.5):
   - First in u and then in v direction
   - Then first in v and then in u direction.
2. Determine the surface normal $\hat{n}$ for the parameters (0.0) and (1.1).
3. Determine the surface normal $\hat{n}$ for the parameters (0.5,0.5).

Assignment 2 [2 Points]

In this task you will extend the curve framework introduced with assignment 02. This time you have to implement B-Spline curves.

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 B-Spline curves in `Curve/BSplineCurve.hpp` and `Curve/BSplineCurve.cpp`.
   - `basisFunction`: In this function you have to implement recursively the actual basis functions $N_i^n(u)$ that are used for B-Spline curve representation. Thus the parameter n represents the B-Spline degree, i the i-th basis function. u is the parameter at which the basis functions has to be evaluated. Additionally this functions provides a parameter k that represents the knot vector (`std::vector`).
   - `evalBasisfunction`: In this function you will have to evaluate the B-Spline curve for parameter u by using the basis functions.
   - `eval`: In this function you have to evaluate the B-Spline curve by using de Boor's algorithm. You can assume that `controlPoints_` contains all de Boor points set by the user and that `knotVector_` contains the currently set knot vector. The class member variable `degree_` represents the current set B-Spline degree.
   - `evalConstruct`: In this function you have to compute and store the intermediate results of the de Boor's algorithm. The results have to be stored in the point hierarchy `contructedPoints_`. It is a C++ `std::map` of point containers where the key value represents the algorithm level e.g. `contructedPoints_.at(0)` returns a `PointContainer` that contains the input points of a segment.
   - `evalCurve`: In this function the curve has to be evaluated for the whole interval depending on a given curve `_resolution`. Push the resulting points into the `std::vector curvePoints_`.

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

Hand in: 23.11.2015, 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.