Assignment in Computer Graphics II
– Assignment 4 –

Computer Graphics and Multimedia Systems Group

Markus Kluge, Dmitri Presnov, Jan Mußmann

Assignment 1 [2 Points] Catmull-Rom Approach

Calculate according to the Catmull-Rom approach, all control points for a cubic Bezier spline through the points $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ whose tangents at the beginning and end are set by additional points $\mathbf{P}_{-1}$ and $\mathbf{P}_3$.

$$\mathbf{P}_{-1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad , \quad \mathbf{P}_0 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad , \quad \mathbf{P}_1 = \begin{pmatrix} 10 \\ 2 \end{pmatrix} \quad , \quad \mathbf{P}_2 = \begin{pmatrix} 20 \\ 5 \end{pmatrix} \quad , \quad \mathbf{P}_3 = \begin{pmatrix} 22 \\ 8 \end{pmatrix} \quad .$$

Assignment 2 [4 Points]

Program Catmull-Rom

This time you have to implement Catmull-Rom 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 Catmull-Rom curves in `Curve/CatmullRomCurve.hpp` and `Curve/CatmullRomCurve.cpp`.
   - `generateCatmullRomControlPoints`: In this function you have to compute the tangent points for all segments of the Catmull-Rom Curve. The input PointContainer `points` contains all points that were set by the user in the GUI of this curve framework. Your task is to fill the `catmullRomControlPoints_` container s.t. it includes **all** Catmull-Rom tangent points and the corresponding control points e.g. elements 0 and 1 represent the first control point and the corresponding tangent control point, elements 2, 3 and 4 represent the first tangent control point, the corresponding control point and the second tangent control point and so on.
   - `eval`: For a given parameter u you have to compute and return the corresponding curve value. You can assume that the member variable `controPoints_` contains the points computed in `generateCatmullRomControlPoints`. 4 successive points (beginning with a control point) define a Catmull-Rom segment that should be evaluated in deCasteljau manner.

- `evalConstruct`: In this function you have to compute and store the intermediate results of the deCasteljau algorithm for the segment corresponding to parameter u. Again, you can assume that `controlPoints_` contains all points computed in `generateCatmullRomControlPoints`. 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: Until 02.05.2019 12:15 o'clock in mailbox of our chair (next to room 7115) and the programming assignment via e-mail (`jan.mussmann@student.uni-siegen.de`**).**