Assignment in Computer Graphics II

– Assignment 7 –

Computer Graphics and Multimedia Systems Group

Marcus Kluge, Dmitri Presnov

Assignment 1 [2 Points] Knot Insertion (Böhm's Algorithm) (non-uniform knot vector)

Give a cubic B-Spline curve with $m = 7$, knot vector $T = \{0,0,0,0,1,2,3,3,4,5,5,5,5\}$ and control points

$$\mathbf{D}_0 = \begin{pmatrix}1\\4\end{pmatrix} \quad , \quad \mathbf{D}_1 = \begin{pmatrix}-3\\-1\end{pmatrix} \quad , \quad \mathbf{D}_2 = \begin{pmatrix}-4\\-9\end{pmatrix} \quad , \quad \mathbf{D}_3 = \begin{pmatrix}2\\3\end{pmatrix} \quad ,$$
$$\mathbf{D}_4 = \begin{pmatrix}2\\7\end{pmatrix} \quad , \quad \mathbf{D}_5 = \begin{pmatrix}6\\3\end{pmatrix} \quad , \quad \mathbf{D}_6 = \begin{pmatrix}9\\11\end{pmatrix} \quad , \quad \mathbf{D}_7 = \begin{pmatrix}8\\13\end{pmatrix} \quad ,$$

- Insert three times a knot with $u = 2.5$ using Böhm's algorithm.
- Name the knots and control points of the resulting curve.

Assignment 2 [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 (0.0,0.0) and (1.0,1.0)!
3. Determine the surface normal $\hat{n}$ for the parameters (0.5,0.5).

Assignment 3 [2 Points] Surface Programming task

In this task you will implement Bezier and B-Spline surfaces analogously to previous assignments related to curves. Download the framework `surface-framework.zip` and take an initial look on the code.

All relevant files for this programming tasks, related to surfaces, can be found in the `Surface` folder. In `Surface/Surface.hpp` you can find the abstract base class for all further surface classes. It provides three abstract member functions `eval, evalSurface, 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.

File `vertices.txt` contains the initially loaded vertices. Each row represents one vertex. Feel free to use it for experiments.

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 Bezier surfaces in `Surface/BezierSurface.hpp` and `Surface/BezierSurface.cpp`.
   - `eval`: In this function you have to evaluate the Bezier surface by using the dual-step de Casteljau algorithm algorithm. You can assume that `controlPoints_` contains all control points. This time `controlPoints_` is an `std::vector<std::vector<glm::vec4>>`. This reflects the two dimension of the surface definition. Thus to access control point C_{01} you have to use `controlPoints_.at(0).at(1)`.
   - `evalConstruct`: In this function you have to compute and store the intermediate results of the dual-step de Casteljau algorithm. The results have to be stored in `contructedPoints_`. It is a C++ `std::map` of two dimensional vectors. The key value represents (as in our curve implementations) the level of the algorithm iteration. Since we have to deal with two dimensions the structure of the map is the following one:
     - key=0: This element should contain all input points.
     - key $\in \{1,\ldots,n_u\}$: These elements should contains the 1st, 2nd, ..., n_u-th iteration steps of the algortihm in the first direction.
     - key $\in \{n_u+1,\ldots,n_u+n_v\}$: These elements shoudl contain the 1st, 2nd, ..., n_v-th iteration step in the algrithm second direction.

     In your implementation you have to determine the degrees in both direction (see `nU` and `nV` in the code). Based on this the the constructedPoints_ map is initialized properly. Further, as in `eval` you have to perform both steps of the dual-step de Casteljau algorithm, but additionally add the interation results to the corresponding map elements e.g. constructedPoints_.at(2) to access all intermediate results from the second step of the algorithm in first direction. To get the result for a specific step in one direction you can use contructedPoints_.at(2).at(step). To access them for the second direction you have to use an offset that is already set in the code e.g. constructedPoints_.at(2+offset).
   - `evalSurface`: In this function the surface has to be evaluated for the whole interval depending on a given `_resolution`. Push the resulting points into the `std::vector<std::vector<glm::vec4>> surfacePoints_`.
2. Implement BSpline surface in `Surface/BSplineSurface.hpp` and `Suface/BSplineSurface.cpp`.
   - `eval`: In this function you have to implement B-Spline surfaces by using the basis functions. You can assume that `controlPoints_` contains all initial de Boor points and that `knotVectorS_`, `knotVectorT_` contains the currently set knot vectors.

Total points after sheet 7: 34 of 70.

Hand in: Until 31.05.2018 12:00 o'clock in mailbox of our chair (next to room 7115) and the programming assignment via e-mail (jan.mussmann@student.uni-siegen.de**)..**