# Assignment in Computer Graphics II
## – Assignment 8 –

**Computer Graphics and**
**Multimedia Systems Group**
David Bulczak, Christoph Schikora

**Assignment 1** [2 Points] Complex numbers

1. Given two complex numbers $p = 3 + 2i$ and $q = -3 + 3i$. Calculate $p + q$ and $p \cdot q$.
2. Specify the real and imaginary parts for the following terms:

$$\frac{3 + i\sqrt{7}}{4}$$

$$e^{1+i\pi}$$

3. Simplify the expression $i + i^2 + i^3 + i^4 + i^5$as much as possible.
4. Find all the (complex) solutions of the following quadratic equation:

$$z^2 - 2z + 10 = 0, \quad z \in \mathbb{C}$$

5. Transform $1 + i$ to polar coordinates.

**Assignment 2** [2 Points] Quaternion

Given the following quaternions

$$\mathbf{q_1} = \tfrac{5}{13} - \tfrac{12}{13}k \qquad \mathbf{q_2} = \tfrac{4}{5} + \tfrac{4}{5}j \qquad \mathbf{q_3} = \tfrac{1}{17} - \tfrac{12}{17}i + \tfrac{12}{17}j$$

and let be $s_{ij} := q_i + q_j$ and $p_{ij} := q_i q_j$ the sum and the product of quaternions.

1. **Addition:** Calculate $s_{12}, s_{23}$ and $s_{13}$.
2. **Multiplication:** Calculate $p_{12}$ and $p_{13}$.
3. Determine if $q_1, q_2, q_3 s_{12}, s_{23}, s_{13}, p_{12}$ and $p_{13}$ correspond to rotations in 3D.

**Assignment 3** [2 Points]

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 `PointContainer` that has been extended to an `std::vector<std::vector<Point3D>>`. 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 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.
   - `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 surfacePoints_`.
2. Implement BSpline surface in `Surface/BSplineSurface.hpp` uns `Suface/BSplineSurface.cpp`.
   - `eval`: In this function you have to evaluate the B-Spline surface by implementing the dual-step de Boor algorithm. You can assume that `controlPoints_` contains all de Boor points and that `knotVectorS_`, `knotVectorT_` contains the currently set knot vectors. The class member variables `degreeU_` and `degreeV_` represent the current set B-Spline degrees.
   - `evalConstruct`: In this function you have to compute and store the intermediate results of the dual-step de Boor algorithm. The results have to be stored in the point hierarchy `contructedPoints_`.
   - `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 surfacePoints_`.

**Hand in: 14.12.2015, at beginning of the lecture or until 12:00 in the mailbox of the chair (next to room H-A 7107) and send files corresponding to the programming task to johnfr93@gmail.com.**