Computer Science and Applied Cognitive Science
High Performance Computing Group
http://hpc.uni-due.de/teaching

Winter Term 2021/2022

Prof. Dr. Jens Krüger
Andrea Iannella

Computer Graphics

Assignment 12 (?? Points)

All assignments are to be uploaded to Moodle on **January 30, 2022 at 11:55pm**. Assignments that are not uploaded to Moodle before the deadline will not be graded (0 points). Assignments that do not compile also receive 0 points. Often example inputs and outputs are provided for you to test your programs; assignments that do not process these examples correctly will receive very few points. Please do not add any additional output to your program than what is requested. Submit your solution as a **single** zip-archive named as follows: **`a12_FirstName_LastName.zip`**.

Include the entire project folder in the archive and for assignments also consisting of theoretical tasks include in addition a text file or a scan/photo with the corresponding answers. Please see assignment sheet 1 for details.

1 Growth Simulation

The provided code in this assignment simulates a growing dendrite-like shape using particles. This simulation process takes place in the `simulate` function of the `main.cpp` file. The first particle is placed in the origin. In each subsequent step, a random position between (-1, -1, -1) and (1, 1, 1) of a new particle is first calculated and then it is moved in random directions until it collides with an existing particle. If the particle moves too far from the origin a new random position is calculated. Without an acceleration data structure each existing particle needs to be checked for a collision. To speed up this process the provided code uses an octree for space space partitioning. The program also shows a visualization of the current octree. You can switch between a 2D and 3D view by commenting out line 2 of the `main.cpp` file.

2 Octree as acceleration data structure (20 Points)

The provided `Octree` class consists of a hierarchical structure of `OctreeNode` objects. The member variable `root` points to the root node. The octree further defines two parameters `maxElemCount` and `maxDepth` which represent the maximum number of elements that one node can hold until it splits into new nodes and the maximum depth of the octree. The provided example program already implements a nearest neighbour search of the octree to figure out the minimum distance to the new particle position in `Octree::minDist`. Your task is to implement the addition of new elements to the octree and splitting nodes into new children if necessary. Complete the member functions `OctreeNode::add` and `OctreeNode::split`. They are annotated with TODO comments and provide further information for the implementation.

You can compare your solution with the clip of the growth simulation on Moodle. Keep in mind that the simulation is based on random positions.