Prof. G. Zachmann
H. Meißenhelter

University of Bremen
School of Computer Science
CGVR Group
June 1, 2021

Wintersemester 2020/21

Assignment on Advanced Computer Graphics - Sheet 2

Due Date 17.06.2021, 23:59 Uhr

Exercise 1 (Hierarchical Collision Detection — Framework: CollisionDetection-Framework, *20 Credits*)

In the framework `CollisionDetectionFramework`, you are expected to implement the algorithm *The General Hierarchical Collision Detection Algo* given on slide 63 in the lecture about *Collision Detection*. You will need to perform collision detection between objects that are approximated from the inside with non-overlapping spheres. The sphere hierarchy on top of the inner spheres for each object is already built and provided to you in a file. To load each inner sphere tree (IST), the constructor `SphereTree::SphereTree(char *filename)` can be used. During the collision detection, you will need to simultaneously traverse the two ISTs, similar to how bounding volume hierarchies (BVHs) have to be traversed.

Normally, the collision detection step is performed before drawing the objects. You can observe this in the method `GLwidget::paintGL()`. To perform the collision detection, the method `SphereTree::checkOverlap(SphereTree &other, Data *data)` is called, i.e., a check for an overlap between the ISTs of the two objects is performed. The input parameters to this method are the IST of the other/second object and a pointer to an instance of the class `Data`. The class `Data` contains the fields `Data::m12` and `Data::m21`, which describe the transformations between the local coordinate systems of the two objects (i.e., `Data::m12` describes the transformation from the local coordinate system of the first object to the local coordinate system of the second object, and analogous for `Data::m21`). In addition, the class `Data` contains the fields `Data::obj1Spheres` and `Data::obj2Spheres`, which are used to store the overlapping innner spheres by calling the method `Data::addSphereIntersectionData()`.

The method `SphereTree::checkOverlap(SphereTree &other, Data *data)` internally calls the method `SphereNode::checkOverlap(const SphereNode *other, Data *data)`, which is the actual recursive function that starts with a check for an overlap between the root nodes of the ISTs of the two objects and should simulatenously traverse the two ISTs. The end of the recursion should be reached when the first pair of intersecting leaf-nodes is found. This is the main method that you need to implement. In this method, you need to make use of the method `SphereNode::overlap(const SphereNode *other, const QMatrix4x4 m)`, which checks whether two spheres overlap. Furthermore, you need to make use of the method `SphereNode::isLeaf()` in order to check if a sphere node is a leaf or not. For visualization purposes, you will need to store the overlapping leaf sphere nodes by making a call to the method `Data::addSphereIntersectionData()` with the right input parameters.

Please have a look at the method `GLwidget::keyPressEvent`, in order to learn how to interact with the OpenGL scene using your keyboard. Note that you need to click inside the OpenGL widget, before any keyboard events are detected.

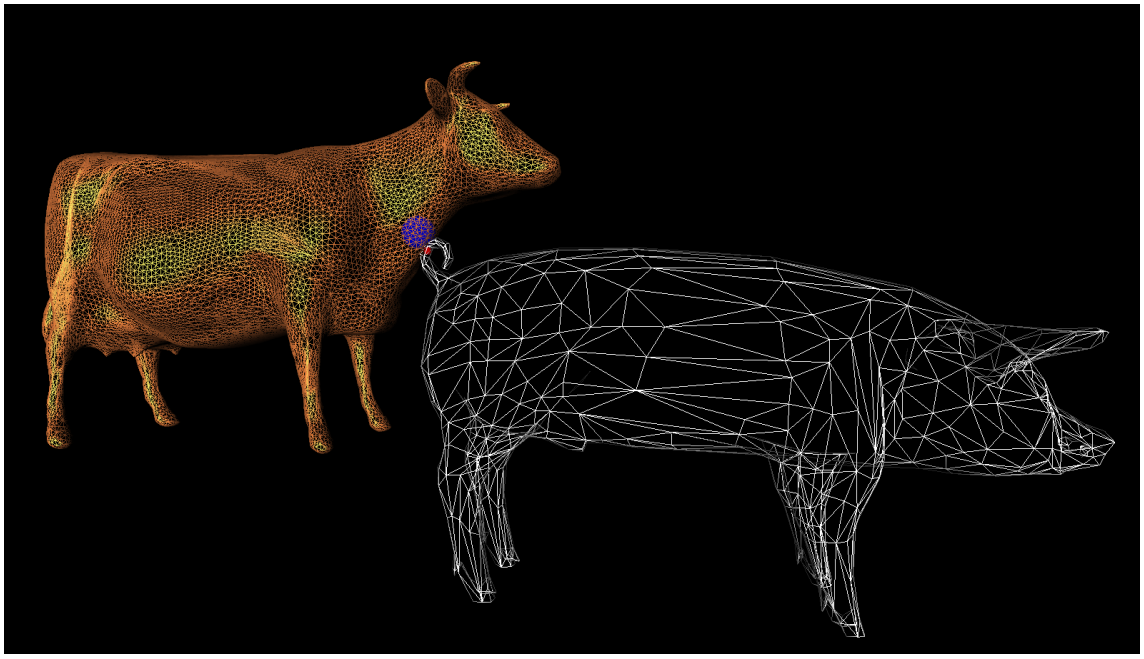

Figure 1: Example of two objects colliding, with the first pair of collding inner spheres visualized.

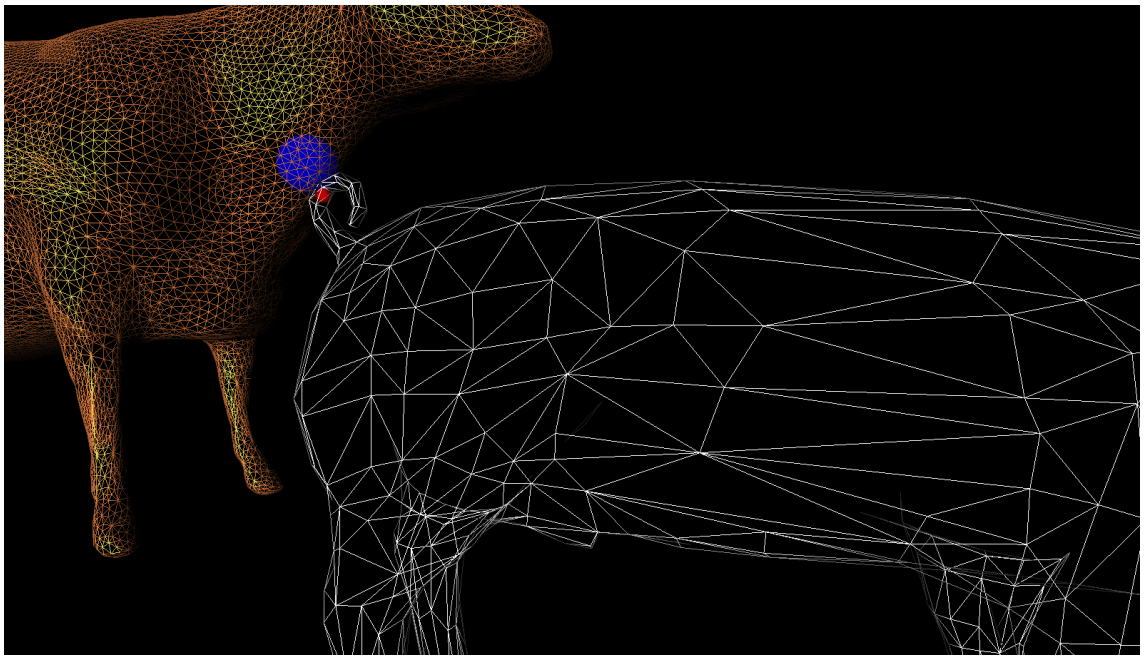

Figure 2: Example of two objects colliding, with the first pair of collding inner spheres visualized. (Zoomed in.)