Physikbasierte Modellierung und Simulation  
https://graphics.tu-bs.de/teaching/ws1819/PBM

ASSIGNMENT 6

Present your solutions for this sheet in the exercise on Thursday, December 20, 2018.
In this exercise, you will implement an efficient collision detection algorithm using axis-aligned bounding boxes (AABBs). Your collision detection algorithm will be used in a simple space flight simulator.

6.1 AABB sorting (30 points)

One possible algorithm for detecting collisions between (possibly moving) AABBs is to maintain – for each spatial dimension ($x$, $y$, and $z$) – a sorted list of their start and end points and a set of pairs of AABBs whose coordinate ranges overlap along this dimension. Whenever a bounding box changes, the list of start and end points is resorted. While resorting the list, it may happen that the end point of one box and the start point of another box are swapped. In this case, these boxes start or cease to overlap, and the set of overlapping pairs has to be updated. Implement this algorithm – for a single spatial dimension – in the `sort` function in `CollisionDetector.cpp`.

6.2 Detect overlapping AABBs (20 points)

When the set of pairs of AABBs with overlapping coordinate ranges is known for each dimension, it is easy to find those pairs of AABBs that actually overlap in three dimensions: those are simply the pairs that overlap along each dimension, or the set intersection of all the sets of overlaps. Compute the set of colliding bounding boxes in the `detect_possible_collisions` function in `CollisionDetector.cpp`.

6.3 AABB update (20 points)

In order to keep the bounding box of a moving space ship up to date, implement the function `update_bb` in `Spaceship.cpp`. You will have to transform the vertices of the triangles (which are given in a local coordinate system) to the global coordinate system through which the ship moves, and find the minimal and maximal $x$, $y$ and $z$ values.

6.4 Collision of a triangle with a sphere (30 points)

As soon as a possible collision is detected by finding overlapping bounding boxes, the two corresponding bodies are tested for an actual collision. In our case, the only type of collision we are concerned about are collisions between (spherical) asteroids and the triangles of the space ship geometry. In `Spaceship.cpp`, implement the function `collides`. It should return true whenever the distance between the center of the asteroid and the closest point on some triangle drops below the size of the asteroid. Since you implemented the collision detection in quite a generic way, you can also modify the constructor of the `Spaceship` class in order to create a more realistic space ship geometry. Your collision detector should then be able to detect collisions with your much more beautiful space ship without any changes to the code.