Prof. G. Zachmann
R. Fischer
N. M. Jadid

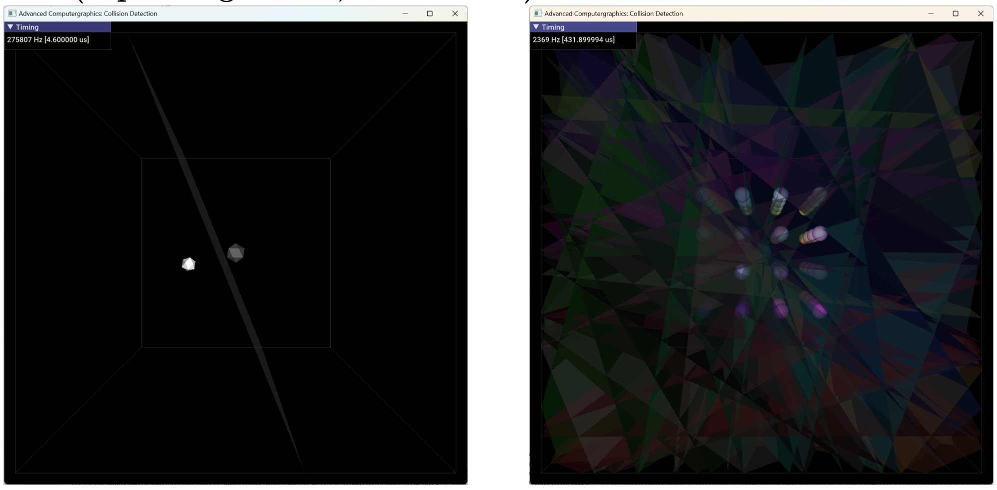
University of Bremen
School of Computer Science
CGVR Group
May 23, 2024

Summer Semester 2024

Assignment on Advanced Computer Graphics - Sheet 4

Due Date 05.06.2024

Exercise 1 (Separating Planes, *10 Credits*)


Advanced Computergraphics: Collision Detection
Timing
Advanced Computergraphics: Collision Detection
Timing

```
275807 Hz [4.600000 us]
```

```
2369 Hz [431.899994 us]
```

Figure 1: The correct results of the separating planes algorithm. **Left**: The result for `SUBDIVISIONS=1` and `OtherBodyDim=0`. **Right**: The result for `SUBDIVISIONS=4` and `OtherBodyDim=4`.

In this assignment we want to implement an acceleration algorithm that uses spatio-temporal coherence to accelerate the broad-phase of collision detection between convex objects.

The algorithm is the separating planes algorithm that was presented in the lecture on Collision Detection. The algorithm maintains a plane for each object pair such that it cleanly separates them, without intersecting either object, if possible. For collision detection, this means an object with a valid separating plane can not intersect, and cosequently, do not need to be considered in the narrow phase.

Use the provided framework **CollisionDetectionFramework**. Implement the algorithm in `PlaneCube::updateSeparatingPlanes` according to the lecture notes (s. slides 19-22 from lecture *Collision Detection*). Each `Body` in `Scene::oribitingBodies` should be tested against `Scene::centerObject`, which is also a `Body`. In `main.cpp`, you can adjust the number of oribiting bodies by setting `OtherBodyDim` to a value between 0 and 4; it will create k^3 (k=`OtherBodyDim`) many orbiting bodies. Each of those k bodies should update its separating plane in the fastest time possible. You can store the separating plane as a homogenous 4D vector in `Body::m_sepPlane` and the flag for its validity in `Body::m_sepPlaneValid`. That means the flag is `false` if the objects (potentially) collide and no separating plane can be constructed, and `true` otherwise.

Note: Make sure that you transform the vertices and planes to a common space. The rendering portion of the framework expects the planes in world space, I suggest you stay with this convention.

Therefore, you need to transform a vertex v_{Model} in model space to world space v_{World}, by using $H_{W \leftarrow M}$=`Body::getModelToWorld()` like $v_{World} = H_{W \leftarrow M} \cdot v_{Model}$ with the fourth component 1 for points and 0 for vectors.

You can modify the application settings at the beginning of `main.cpp`. Test your implementation with the settings `SUBDIVISIONS=1` and `OtherBodyDim=0` to verify correct behaviour. Afterwards, accelerate your algorithm and benchmark with `SUBDIVISIONS=4` and `OtherBodyDim=4`. For reference, the sample solution completes the benchmark setting in around 500 µs.

Spatio-temporal coherence involves using the most recent separating plane for the Body-pair as the starting candidate, as well as using steepest descent to quickly determine the next "offending" vertex ("offending" meaning that when a vertex is on the wrong side of the plane and thus forces an update of the plane to be valid)