Fall 2016: ELG5124/CSI5151
Assignment 2

Due: Wednesday, October 19th, 2016, 11:00pm in Virtual Campus
University of Ottawa - Université d'Ottawa

Jochen Lang

1 Penalty-Based Rendering

This assignment will be exploring penalty-based rendering as it commonly used for haptics in Virtual Environments. The specific algorithm that you will be partially implementing is the "god-object algorithm" by Zilles and Salisbury [2]. A simulator to integrate you implementation is provided using the JMonkey engine. You will have to extend the scene and implement the algorithm for convex obejcts.

1.1 Installation and Getting Started

This assignment will again require you to use *JMonkey 3.0* (`http://www.jmonkeyengine.org/`) and the starter project `Penalty`. Please download the starter project from BB learn.

1.2 Completing the Scene and Ray-Object Intersections [2]

Currently a single quad shape is shown as the object in the scene. Add five more quads to form a hexagonal cylinder shape. Add a corresponding mesh shape to the top and bottom of the hexagon. Keep the side facing the camera transparent but make the far end from the camera opaque. The example shows how to render a transparent quad. When you are done with this step, you should be able to move the red sphere symbolizing the haptic interaction point moving in and out of the hexagonal cylinder.

Verify that you receive the expected console output from the ray-object intersection calculated by the program.

1.3 Proxy (or God-Object) Sphere [1]

Add a yellow sphere (same diameter than the HIP sphere) representing the Proxy (or God-Object) to your scene. Also add a line connecting the HIP and the Proxy.

1.4 Constraint-based Proxy Placement [4]

Implement a repositioning of the Proxy every time the user hits the space bar. Your implementation must correctly work with the sides and bottom and top of the hexagon. It must also correctly identify the case when the HIP leaves not only the current quad but also the object.

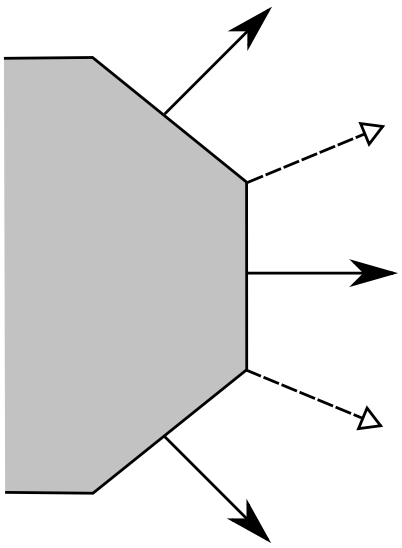

Figure 1: Normal Interpolation for Hexagon Faces in Force Shading

1.5 Force Shading [3]

Force shading is a technique as proposed by Ruspini et al. [1] and discussed in class. Calculate edge normals for the edges along the sides of the hexagon as the average between the two neighboring planes. Each quad has now two normals (we ignore the edges towards the botton and top), and the interpolated normal is found by the distance from each edge (see Figure 1). Implement Ruspini et al.'s force shading technique with these normals. Note that this is a three step procedure:

- Calculate the Proxy position as before without force shading.
- Find the interpolated normal for the Proxy position.
- Use the interpolated normal to update the Proxy and hence the force direction.

2 Submission

You will need to submit your solution (only the source directory and your assets directory along with the `*.xml` project files but no other files) to BB learn by the deadline. No late submissions are allowed, you can submit multiple times but only your last submission is marked.

References

[1] D.C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. In *Computer Graphics, Annual Conference Series*, pages 345–352, Los Angeles, USA, Aug 1997. ACM.

[2] C.B. Zilles and J.K. Salisbury. A constraint-based god-object method for haptic display. In *International Conference on Intelligent Robots and Systems*, Pittsburgh, Penn., USA, Aug 1995. IEEE/RSJ.