**# PointToMesh**

Find the closest point to a 3D triangle mesh surface from a given point

**## Algorithm Used**

Algorithm detailed in paper [Fast and Accurate Closest Point Search on Triangulated Surfaces and its Application to Head Motion Estimation](https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.5.152&rep=rep1&type=pdf) and a naive algorithm to iterate through all triangles.

**## Build**

make all would create release build. make release and make debug are valid too. Can run it as

./release/Point2Mesh symphysis.obj 23 -278.22 99. ./release/Point2Mesh <filename> <3D point>

This runs both the naive and sphere tree algorithm printing out the time elapsed and details about the faces considered.

**## Tests**

cd tests

make all

./TestPoint2Mesh

This runs both perf tests and unit tests between the naive vs sphere tree algorithm

**## Challenges**

The paper does not describe the way threshold is picked and I spent a lot of time fine-tuning it. The algorithm is very sensitive to it. With a bad threshold, the resulting tree traversal takes a lot of time. Also, it does not describe how two sphere nodes are picked from the ***working list***  and merged. Instead of finding the sphere with the best ratio, I went ahead and took the first sphere that met the criteria for speed gains.

For unit tests, Float is **double,**  this is because some tests failed randomly with precision issues.

The next big challenge I faced was while writing unit tests for the routine to find the closest point on a Triangle. I used the code found here : [***https://www.gamedev.net/forums/topic/552906-closest-point-on-triangle/***](https://www.gamedev.net/forums/topic/552906-closest-point-on-triangle/)as the base reference against my algorithm. However, it would fail for some points. I spent quite some time debugging and found out that the base reference is incorrect. One such sample is in debug\_PointToTriangle()

To test end to end, I have used the naive implementation as the reference. Perf tests are also provided and show quite a lot of speedup with the implemented algorithm.

**## Assumptions**

1) works with a 3d triangle mesh

2) input mesh should be in obj format with no texture information

3) compiled and tested with C++17

4) clang formatted.

**## Sample Output**

Using mesh symphysis.obj

Read in mesh with 60792 vertices and 51087 faces.

Working list size before building tree 51087

Working list size after building tree 1

Elapsed time in milliseconds for building tree : 169 ms

Elapsed time in microseconds for naive : 3015 micros

Faces to consider : 158

Elapsed time in microseconds for sphere : 65 micros

[ 16.2766, -66.534, 47.0163 ]

[ 16.2766, -66.534, 47.0163 ]

Cleanup.. Deleting all nodes now..

**The two points outputted are from Naive and sphere tree implementation.**

**## Meshes tried with (present in directory):**

bunny.obj

teapot.obj

symphysis.obj