CS350
Assignment 2: Bounding Volumes and Spatial Partitions

Submission:

Zip up and submit the entire project folder (the .sln and the CS350Framework folder). Make sure to not include any build artifacts! See the syllabus for submission specifications. Due October 16th (Sunday) by 11:55 pm.

Topics:

The purpose of this assignment is to be able to construct and use core bounding volumes (Spheres and Aabbs) for simple optimizations. This includes using them for bounding volume spatial partitions to optimize ray-casting, frustum-casting, and pair-queries. Simple debug drawing is also required as debugging large scenes almost necessitates a basic debug drawing library.

Testing:

You will be given a txt file containing sample results for the assignment. To test your project for this assignment run with the command line arguments of *"2* 0". The argument "0" runs all of the tests at once.
Do note that the graded tests will not be limited to the ones given to you!

Implementation Details:

Debug Drawing:

Use the GetNewShape function to create a DebugShape that you fill out with line segments. Also make sure to return this shape from your draw function. For DrawSphere you must do the horizon disc calculation for full points. DrawRay must have an arrow head of some kind. DrawPlane must draw the plane normal (draw it at the center of the plane's quad). Note: GetNewShape returns a reference, make sure you don't copy this. Also make sure to not create more than one debug shape per draw call.

ComputeRitter:

Some rules to help avoid ambiguities. Whenever visiting points make sure to do so in array order (0 to size). Whenever there is a tie, only pick a value if it is greater than the previous min/max. For example, when picking extremal points only pick a point if it is more "extreme" not tied. When it comes to picking the largest axis pick from x then y then z with the same update rules (only pick y if it is > x, not >=).

ComputePCA:

3 helper functions are provided that you must implement: ComputeCovarianceMatrix, ComputeJacobiRotation, and ComputeEigenValuesAndVectors. These should be used in the inner algorithm of PCA and are exposed to help you debug.

When picking the extreme points on each axis you should visit points in order (0 to size) and pick the first extreme point (in case of a tie). Also when choosing the largest axis pick x then y then z in the case of ties (y must be > x to be picked). Point expansion should be in the same order.

Aabb::Transform:

Just update the aabb in place by applying the transform as described in class. Do not transform all 8 points and then compute the resultant aabb, you must use an optimized method!

SpatialPartitions:

Do not assume anything about SpatialPartitionData::mClientData, you should do nothing more than storing and returning it. In particular, do not assume that a value of 0 can be used to denote an empty item (in your internal storage). Also you should not do a linear search to find an object during update and removal, this operation must be constant time! All cast operations should be linear time and SelfQuery should be no worse than quadratic time.

When implementing FilloutData on your spatial partition make sure to only fill out the array with valid data (don't add empty items) and just ignore mDepth. For this assignment only, the order that you fill out this array doesn't matter.

When implementing DebugDraw just use the passed in transform, color, and bit mask to set each debug shape (use ".Color()", ".SetMaskBit()", and ".SetTransform()"). Make sure modify the actual debug shape and not a copy! For this assignment 'level' is not used.

Code Quality:

Code quality is a percentage modifier applied to your grade up to a -20%. This is mostly to ensure that your code is easy for graders to look at and understand what you were doing.

Grade Breakdown:

	Points	Percentage
Shapes.cpp:	**40**	**45%**
Sphere:	*24*	*27%*
ComputeCentroid	4	5%
ComputeRitter	8	9%
ComputePCA	12	14%
Aabb:	*16*	*18%*
GetVolume	2	2%
GetSurfaceArea	2	2%
Contains	2	2%
Transform	10	11%
SimpleNSquared.cpp:	**30**	**34%**
BoundingSphereSpatialPartition	*30*	*34%*
Insert/Update/Remove	15	17%
RayCast	5	6%
FrustumCast	5	6%
SelfQuery	5	6%
DebugDraw.cpp:	**18**	**20%**
DrawPoint	2	2%
DrawLine	2	2%
DrawRay	2	2%
DrawSphere	2	2%
DrawAabb	2	2%
DrawTriangle	2	2%
DrawPlane	2	2%
DrawQuad	2	2%
DrawFrustum	2	2%
Total	88	100%

Notes:

Avoid creating new files in the framework. I will be copying the relevant files for the assignment into a clean project for testing. For this assignment that is:

1. Geometry.hpp/cpp
2. Shapes.hpp/cpp
3. SimpleNSquared.hpp/cpp
4. DebugDrawing.hpp/cpp

The full framework should still be submitted though.

To make it easier to find missing functions they contain the comment "/******Student:Assignment2******/" that you can search.