E9: Axis Aligned Bounding Box

Course: IGME 309 – Real Time Simulations for Games II

Golisano College of Computing and Information Sciences

School of Interactive Games and Media

Rochester Institute of Technology

Due: Check in MyCourses

Deliverable: RigidBody.cpp file (single file, unzipped)

**Objective:**

The objective of this exercise is to help students understand how to generate the smallest possible Axis-Aligned Bounding Box (AABB) that tightly fits a 3D model, while ensuring that the box is oriented to the global coordinate space. By computing the correct **Minimum**, **Maximum**, **Center**, and **Halfwidth** vectors for the bounding box, students will learn to create efficient bounding boxes for collision detection, spatial partitioning, and other computational geometry tasks.

Through this exercise, students will:

* Learn how to compute the **Minimum** and **Maximum** points of an AABB, ensuring the box completely encloses the 3D model in local space.
* Understand how to calculate the **Center** of the AABB, which is the midpoint between the **Minimum** and **Maximum** vectors, representing the center of the bounding box.
* Discover how to compute the **Halfwidth** of the bounding box, which defines the half-length of the box along each axis and is useful for calculations like collision checks and spatial queries.
* Implement a method for transforming these local space values into the global space, ensuring that the bounding box is oriented correctly in world coordinates.
* Learn how to compute the smallest-sized bounding box that encloses a model, which is critical for applications such as collision detection and visibility culling in 3D environments.
* Understand the challenges of maintaining both tightness (i.e., the box should not be too large) and correct orientation in global space.

This exercise will provide students with practical skills in geometry and transformations, particularly in the context of 3D modeling and game development. They will also gain insights into optimizing the bounding boxes for performance and accuracy in real-time applications.

**Instructions:**

This exercise follows lecture D09

1. Under \_Binary look for the example solution. It will look like this when the creeper is moved:

A screenshot of a computer

Description automatically generated

1. Out of the box the yellow box the collision does work, it will default to true (its always colliding even if its not) for this exercise you need to fix the collision. Assume all code in Rigid body is working except for the collision.
2. For this you will need to follow the comments found in the method IsColliding under RigidBody.cpp:

//TODO: Check if they are not colliding if at least

//one of the following 6 conditions are true

//This is to the Left of Other

//This to the Right of Other

//This Below Other

//This Above Other

//This Behind of Other

//This In front of Other

1. You are only modifying the RibigBody.cpp so this is the only file you need to submit

