E15: Recursive Grid

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: Single zipped file of Node.h and Node.cpp

**Objective:**

The objective of this exercise is for students to implement a **recursive subdivision** of a previously subdivided grid. Starting with the root grid cell, the space will be recursively divided, and each new subdivision will continue dividing until a predefined **maximum level of subdivisions** is reached. This recursive process will help students better understand how tree data structures work and how recursion can be applied to spatial partitioning problems.

By completing this exercise, students will:

* Learn how to recursively subdivide grid cells, applying the same subdivision logic from the previous exercise but with the added complexity of recursive depth.
* Understand the role of **recursion** in data structures like trees, and how it can be used to break down a problem into smaller, more manageable sub-problems.
* Implement the recursive subdivision logic that divides the world space into smaller sections, ensuring that each level of subdivision continues until the **maximum level of subdivisions** is reached or until the number of entities in any grid cell exceeds a given threshold.
* Gain practical experience in working with **tree-like structures**, understanding how each grid cell (node) can have children (subdivided grid cells) that represent smaller sections of space.
* Understand how the recursive subdivision process can be used to improve performance in simulations by creating a hierarchical spatial structure that can be used for efficient collision detection and entity management.
* Learn how **Octrees** operate similarly, with subdivisions happening recursively on the **X**, **Y**, and **Z** axes, where each node is subdivided into 8 child nodes when necessary. This is similar to how the subdivision will occur, but with the added note that subdivisions are triggered when the maximum level is not yet reached or when a grid cell contains more than the ideal number of entities.

**Note:** An **Octree** is a structure that follows a similar behavior, where each subdivision of the space (on the X, Y, and Z axes) splits the node into **8 smaller nodes**. An Octree only continues to subdivide if it has not yet reached the maximum subdivision level, or if a grid cell contains more entities than the optimal threshold. This concept introduces a more efficient way to partition space when dealing with large, dynamic worlds, and forms the basis for many real-time simulations and game engines.

This exercise will deepen students’ understanding of recursion and spatial partitioning, providing a foundation for more advanced hierarchical structures like **Octrees**, which are crucial for optimizing space queries, collision detection, and efficient entity management in large-scale simulations.

**Instructions:**

This exercise follows lecture D13

1. Your code will start like this:

A screenshot of a computer

Description automatically generated

1. Under \_Binary look for the example solution. It will look like this:

A screenshot of a computer

Description automatically generated

1. Out of the box the subdivision on the grid will be already functional, and the connection between the entities and the spaces is as well. What you must do in this convert the functionality to be a recursive method. Both grids have the same level of subdivisions, but the algorithm for this new one will divide the space in 8 and then it will divide the new ones in 8 and lastly once again each one of them in 8. The old way of doing it will take the space and divide it in equal parts (in this case 3 parts instead of 3 levels)
2. This exercise only requires the Node.h and Node.cpp file for submission. You may or may not have modified the Node.h file, just to be sure, include it as well.

As a heads up to what is following, this is what an Octree would look like, this is your next homework assignment.

A red circle in a cage

Description automatically generated

An octree will only divide the space when its necessary, based on two conditions, how many obects I ideally want per space/node/octant and what is the maximum level I want to reach to