📘 CS350/CSD3150 Project 4: Spatial Partitioning using Adaptive Octree and KD-Tree

---

📅 Submission Details

> **Files (submit archive) due:**
>
> - 📅 **Wednesday, July 24, 2025

📝 Reminder

> *"The major point of this project is for you to learn how to submit your homework correctly. Follow all instructions exactly. If unsure, ask your instructor, a TA, or another student for clarification."*

🎯 Purpose of This Project

> *"In this project, you will build an Adaptive Octree and KD-tree for the provided UNC PowerPlant sections, implementing different resolution methods, termination conditions, and display options."*

You will:

- Load and render sections of the UNC PowerPlant model.
- Build an **Adaptive Octree** spatial partition.
- Build a **KD-tree** spatial partition.
- Implement visual toggling of trees and their levels.

🧩 Requirements

🧱 Scene Setup

- Continue using the previous project's scene setup.
- Load three models: `ppsection4`, `ppsection5`, and `ppsection6`.
- Populate and render the scene ignoring `.mtl` files.

🌳 Adaptive Octree Creation (40%)

- Build the octree using **top-down** fashion.
- Create subtrees only if objects exist within the parent node.
- Handle boundary (straddling) objects using any two of the following methods:
    - Associate based on **object center**.

    - Associate to **all overlapping cells**.
    - Associate to **current level's cell**.
    - **Split the object** across cells.
- Allow toggling between the two straddling methods.
- Terminate subdivision based on **number of objects** per cell (default 10, user-adjustable).
- **Display cells** with different colors per level.

🌲 KD-Tree Creation (40%)

- Build the KD-tree in **top-down** fashion.
- Split **one axis per split** in the order: X ➔ Y ➔ Z.
- Implement two of the following split-point strategies:
    - Median of BV centers
    - Median of BV extents
    - K-even splits along an axis
- Allow toggling between split methods.
- Terminate subdivision based on **number of objects** per node (default 10, user-adjustable).
- **Display nodes** with different colors per level.

🎨 Display Requirements

- Toggle display of:
    - Adaptive Octree levels (different colors).
    - KD-tree levels (different colors).

⚙️ Submission

1. Create a copy of project directory project-4 named <login>-<project-4>. That is, if your Moodle student login is foo, then the directory should be named foo-project-4. Ensure that directory foo-project-4 has the following layout:

```
📘 foo-project-4          # 🖋️ You're submitting Project 4
├─ 📁 include                # 📇 Header files - *.hpp and *.h files
└─ 📁 src                    # 🌟 Source files - *.cpp and .c files
└─ 📁 shaders                # 🌟 Shader files - *.vert and .frag files
|  └─ 📝 my-project-4.vert  # 📝 Vertex shader file
|  └─ 📝 my-project-4.frag  # 📝 Fragment shader file
└─ 📝 README.txt             # 📝 [IMPORTANT] Don't forget to add this
```

2. Make sure you are not using absolute path for shader files, it will not work on instructor's machine.
3. Inside foo-project-4, add README.txt file. README must Include:

    - UI usage instructions (especially if not described in the project brief)
    - Assumptions and crash conditions
    - Completed parts
    - Incomplete/buggy parts with explanation
    - File paths, function names, and line numbers of key logic
    - Test platform details (e.g., `Windows 11, NVIDIA 3070, OpenGL 4.6`)
    - Weekly time breakdown
    - Any other useful notes
        - Add **key mappings**, assumptions, and known issues.
        - Track your weekly effort hours.
        - Be explicit about:
            - What is completed
            - What is not working, and why
            - Platform and GPU details (lab machine or your home setup)

4. Re-run the CMake command to build the new project named foo-project-4. If you're unsure how to run the CMake command, please refer to the <your-sample-framework-location>/README.md file.
5. Build and execute project foo-project-4 by opening the Visual Studio 2022 solution in directory build. Test it, make sure it works good.
6. Use File Explorer to open directory <your-sample-framework-location>/projects. Open the command-line shell by typing cmd [and pressing Enter in the Address Bar]. Execute the following PowerShell command to zip it with name [by typing the script's name in the shell and then pressing Enter].

   ```
   powershell if (Test-Path foo-project-4.zip) { Remove-Item foo-project-4.zip -Force };
   Compress-Archive -Path foo-project-4 -DestinationPath foo-project-4.zip
   ```

   [IMPORTANT] Please use only the command provided above for zipping, as it generates the archive in the specific format required by the automation tool for grading. Using any other method may result in incompatibility or failed evaluations.
7. Submit this zip file on Moodle.

🧪 Grading Breakdown

Component	Weight
Scene Creation	15%
└ Scene created with specified objects	15%
Adaptive Octree	40%
└ Creation of adaptive octree	10%

Component	Weight
└ Two straddling resolution methods	10%
└ Termination criteria implementation	10%
└ Colored level rendering	10%
KD-Tree	40%
└ Creation of KD-tree	10%
└ Two split-point experiments	10%
└ Termination criteria implementation	10%
└ Colored level rendering	10%
Miscellaneous Issues	5%
└ Missing README	-2%
└ Compile/Execution/Scene Errors	-3%
Total	100%

📷 Sample Output

- Adaptive Octree visualization (reference only)
    - Adrian Peter Togeskov - XNAGameEngine, Oct- and BSP-Trees
- Different levels represented using color coding.(see above reference)

🔗 Notes

- You can use **OpenGL** or **DirectX** or **Vulkan**.
- Be able to **explain and derive every line** of your code.
- A **README.txt** must describe:
    - Key mappings
    - Choices for straddling methods
    - Choices for termination conditions
    - Observations and notes

> *Refer to the course syllabus for submission guidelines.*

🛡️ Legal Notice

> *"Spatial partitions like Octrees and KD-Trees are fundamental for efficient rendering, collision detection, and real-time simulation. Master them well!"*

🛠️ Geometry Toolbox Tips

Implement Adaptive Octree and KD-Tree

(Project 4: Spatial Partitioning)

- Build an **Adaptive Octree**:
  - Subdivide based on number of objects per cell.
  - Handle straddling objects:
    - Associate to center / all overlapping / split object
- Build a **KD-Tree**:
  - Split alternately along X ➔ Y ➔ Z.
  - Terminate based on number of objects.
- Provide UI toggles:
  - Choose termination criteria.
  - Choose split strategies.
  - Switch tree type display (Octree or KD-Tree).
- Visualize each level of the tree with distinct colors.

🛠️ *Tips*:

- Build trees offline during loading if scene is static.
- For dynamic scenes, rebuild or update tree incrementally.