

CS350/CSD3150 Project 2: Bounding Volumes and View Frustum Culling

Submission Details

Files (submit archive) due:

-  By Friday, June 6, 2025, 8:59 AM

Reminder

"The major point of this project is for you to learn how to submit your homework correctly. You should strive to follow all of the directions exactly as specified in the handouts. The syllabus that was handed out during the first day of class also contains important information on how to submit your homework. If you are still unsure of how to do something, you should ask for help—either from me, a tutor in the lab, or another student."

Purpose of This Project

"This project focuses on building bounding volumes and using them to implement view frustum culling."

You will:

- Load a scene with multiple models
- Compute various types of bounding volumes (AABB, BSphere, OBB)
- Use the bounding volumes to perform **View Frustum Culling (VFC)**
- Visualize visibility status using debug display

Requirements

Scene Setup (10%)

- Load a scene with at least **4 objects** (e.g. bunny, cube, cup, rhino)
- Use **Assimp** (or similar library) to load `.obj` files (ignore `.mtl`)
- Implement:
 - A **First-Person Camera** (WASD)
 - A **Top-Down Debug Camera Or Orbital Camera**

Bounding Volume Calculations (60%)

Compute and implement:

- **AABB (Axis-Aligned Bounding Box)**
- **Bounding Spheres (BSphere)** using:

- Ritter's Method
- Modified Larsson's Method
- PCA-based Method (using covariance and eigen vectors)
- **OBB (Oriented Bounding Box)** using PCA

Bounding Volume Display (25%)

- Wireframe rendering of bounding volumes
- Toggle visibility of actual object (wireframe or hidden)
- Allow switching between volume types
- Use distinct colors for:
 - Inside frustum
 - Outside frustum
 - Intersecting frustum

Submission

1. Create a copy of project directory **project-2** named `<login>-<project-2>`. That is, if your Moodle student login is foo, then the directory should be named **foo-project-2**. Ensure that directory **foo-project-2** has the following layout:

 foo-project-2	#  You're submitting Project 2
 include	#  Header files - *.hpp and *.h files
 src	#  Source files - *.cpp and *.c files
 shaders	#  Shader files - *.vert and *.frag files
 my-project-2.vert	#  Vertex shader file
 my-project-2.frag	#  Fragment shader file
 README.txt	#  [IMPORTANT] Don't forget to add this

2. Make sure you are not using an absolute path for shader files, it will not work on the instructor's machine.
3. Inside **foo-project-2**, 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-2**. 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-2** 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-2.zip) { Remove-Item foo-project-2.zip -Force };
Compress-Archive -Path foo-project-2 -DestinationPath foo-project-2.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 Setup	10%
└ Object Load & Diffuse Light Rendering	5%
└ Simple Diffuse Shading Code Implementation	5%
Bounding Volume Calculations	60%
└ AABB	5%
└ Sphere – Ritter's	7.5%
└ Sphere – Larsson's	12.5%
└ Sphere – PCA	15%
└ OBB – PCA	20%
Display & Interactivity	25%
└ Toggle bounding volume types	5%
└ Show volumes in wireframe	5%
└ Debug + First Person Cameras	15%

Component	Weight
Miscellaneous	5%
└ Missing README	-2%
└ Compile Errors	-1%
└ Execution Errors	-1%
└ Scene Incorrect	-1%
Total	100%

 *Note: If your application does not load models correctly or fails to show frustum culling, your submission cannot be graded.*

References

- Use [OpenGL Extensions Viewer](#) to find driver and GPU info.

Legal Notice

Copyright © 2019 DigiPen (USA) Corp.

No part of this project may be copied, transmitted, or distributed without explicit permission from DigiPen (USA) Corp., 9931 Willows Road NE, Redmond, WA 98052.

"This project builds on what you've learned in A1, but now we're visualizing culling — which is essential for any modern renderer or game engine."