

CS350/CSD3150 Project 1: Geometry Toolbox

Submission Details

Files (submit archive) due:

-  Week 3
-  By Thursday, 22 May 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

"The purpose of this project 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 (future project), and pair-queries."

You will:

- Construct basic geometric primitives.
 - Implement collision/intersection tests.
 - Practice integrating bounding volumes in spatial queries.
-

Topics Covered

The project covers the following core primitives:

- `Point3D`
- `Plane`
- `Triangle`
- `Bounding Sphere`
- `Axis-Aligned Bounding Box (AABB)`
- `Ray`

You are to implement operations for:

✓ Basic Intersection Tests

- Sphere vs Sphere
- AABB vs Sphere
- Sphere vs AABB
- AABB vs AABB

✓ Point-Based Tests

- Point vs Sphere
- Point vs AABB
- Point vs Triangle
- Point vs Plane

✓ Ray-Based Tests


- Ray vs Plane
- Ray vs AABB
- Ray vs Sphere
- Ray vs Triangle

✓ Plane-Based Tests

- Plane vs AABB
- Plane vs Sphere

Submission

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

```
1   foo-project-1      #  You're submitting Project 1
2  |   include          #  Header files - *.hpp and *.h files
3  |   src              #  Source files - *.cpp and .c files
4  |   shaders          #  Shader files - *.vert and .frag files
5  |  |   my-project-1.vert #  Vertex shader file
6  |  |   my-project-1.frag #  Fragment shader file
7  |  |   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-1**, 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-1`. 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-1` 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].

```
1 powershell if (Test-Path foo-project-1.zip) { Remove-Item foo-project-1.zip -Force
  }; Compress-Archive -Path foo-project-1 -DestinationPath foo-project-1.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
Task	25%
└ Task 1: Window class	5%
└ Task 2 & 3: Buffer class VBO and Attribute management	5%
└ Task 4: ECS	5%
└ Task 5: Light rendering	5%
└ Task 6: Interactivity (Camera)	5%
Collision Tests	70%

Component	Weight
↳ All pairwise tests	5% each
Misc. & README completeness	5%

"Remember: if your README is missing, the rest of your grade might be **voided**." 🤖

★ Extra Credit (Up to +30%)

- **+20%:** All tasks are completed on time with high-quality implementation. Demonstrates thoughtful design, clean code structure, and attention to detail in all areas.
- **+10%:** Dynamic PiP view (top-down mini-map of the scene) OR or Orbital camera view with zoom (switch FPS and Orbital Camera with key `C`)

🎯 References

- [OpenGL Extensions Viewer](#)

"If you're unsure about anything—ask your instructor, a tutor, or even your peers. Don't wait until the deadline." 🙋

🛡️ 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."

🔧 Geometry Toolbox Tips

Initial Engine Setup for Geometry Toolbox Suggestion

Refer to: Geometry Toolbox Engine lecture slides