24783 Advanced Engineering Computation: Problem Set 8

(*) In the following instruction (and in all of the course materials), substitute your Andrew ID for where you see *yourAndrewId*.

START EARLY!

1 Check Out or Update Base Code and Libraries

Please make sure you have up-to-date libraries and course files before starting an assignment.

If you have not done working-directory set up as described in the first assignment (like in case you need to work from a different computer), please see Problem Set 1 and set up the working directory.

I assume you created the working directory called *24783* under you home directory and you checked out your Git repository in there.

Home directory is typically *C:\Users\username* in Windows, */Users/username* in macOS, and */home/username* in Linux, where *username* is the user name in your local computer.

First, open command-line (Developer PowerShell or Terminal), and move to your working directory by typing:

```
cd ~/24783
```

You need to check out (or clone) Git repositories once. If you have not checked out yet, do the following:

```
git clone https://yourAndrewId@ramennoodle.me.cmu.edu/Bonobo.Git.Server/course_files.git
git clone https://yourAndrewId@ramennoodle.me.cmu.edu/Bonobo.Git.Server/yourAndrewId.git
```

You need to replace "yourAndrewId" with your Andrew ID. You'll be asked to type in credentials.

Also we are going to use two additional repositories:

```
git clone https://github.com/captainys/MMLPlayer.git
git clone https://github.com/captainys/public.git
```

If you are successful, you should have the following directory structure under your home directory.

```
Your User Directory
├── (Other files and directories)
└── 24783
    └── course_files
```

```
└── yourAndrewID
```

If you already have checked out these repositories (most likely you did for Problem Set 1), you need to update (or git pull) in those repositories. By change directory to the location where you checked out repositries and then type:

```
git pull
```

To update all four repositories, you can type the following commands in a sequence:

```
cd ~/24783/course_files
git pull
cd ~/24783/yourAndrewID
git pull
cd ~/24783/public
git pull
cd ~/24783/MMLPlayer
git pull
```

2 Copy Base Code and Add to Git's Control

Copy ps8 subdirectory from course_files to your directory. The directory structure must look like:

```
Your User Directory
├── (Other files and directories)
└── 24783
    ├── public
    ├── course_files
    └── yourAndrewID
        └── ps8
            ├── ps8_1
            └── ps8_2
```

3 Make CMake Project Files

In this problem set, you write three CMakeLists.txt scripts. One is the top-level CMakeLists.txt, which needs to enable C++11 features, include public libraries, and include ps8_1 and ps8_1 sub-directories.

The other two are for two sub-directories. CMakeLists.txt in ps8_1 sub-directory needs to have an executable target called ps8_1 which uses all files in ps8_1 sub-directory. CMakeLists.txt in ps8_2 needs to have an executable target called ps8_2 which uses all files in ps8_2 sub-directory.

4 Highlighting Small Dihedral Angle

In ps8_1, you complete function called MakeSmallDihedralAngleHighlight. This function is empty in the base code. The function takes two references to std::vector of GLfloats and a constant reference to a PolygonalMesh class.

The function needs to make a vertex and color arrays that can be given to glVertexPointer and glColorPointer, and drawn as GL_LINES.

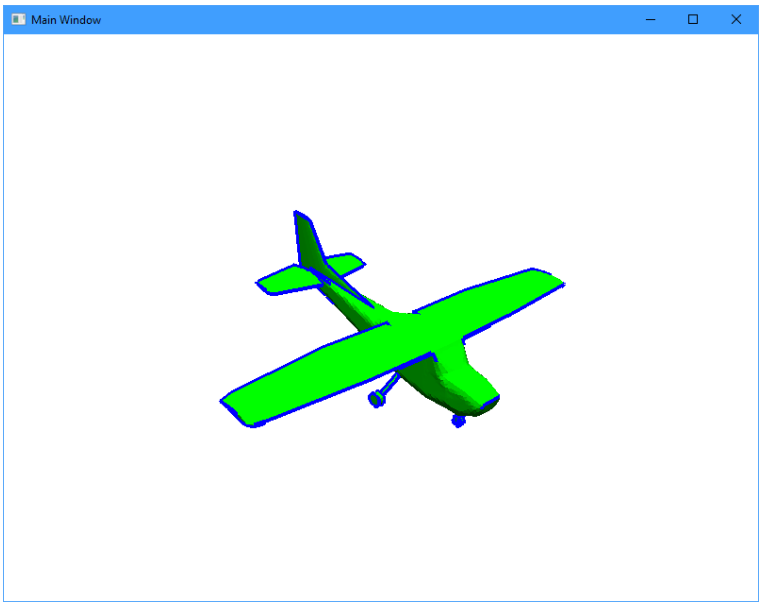

Fig. 1: PS8_1 Example

The function prototype is as follows:

```
void MakeSmallDihedralAngleHighlight(
    std::vector <GLfloat> &vtx,
    std::vector <GLfloat> &col,
    const PolygonalMesh &mesh)
{
// Write this function.
}
```

In this function, measure normal vectors of two neighboring polygons, and the two normal vectors make greater than 60 degrees, add the edge to the vertex and color arrays.

The base-code PolygonalMesh class has a convenient function called GetNeighborPolygon, which takes a polygon handle plHd, and edge number i. This function returns a polygon handle that is connected to plHd on edge i. If the polygon uses vtHd[0] to vtHd[n], edge i of the polygon is connecting vtHd[i] and vtHd[(i+1)%n].

The base code takes an STL file name as parameter. You can start the program as:

```
ps8_1 c172r.stl
```

If you implement correctly and if you open c172r.stl in the sample data directory, you will see a screenshot as shown in Fig. 1.

5 Accelerating Particle Simulation using a Lattice

In ps8_2, you practice to use a lattice structure to accelerate particle simulation. The base code is a multi-threaded version of the implementation of the paper:

Mathias Müller, David Charypar, and Markus Gross, *Particle-Based Fluid Simulation for Interactive Applications*, Proceedings of SIGGRAPH 2003, pp. 154-159

(1) First, add a member variable called maxY in FluidSimulator class, and give the initial value of 80.0.

(2) Then, in BounceOnWall function, make particle bounce on maxY.

(3) Add a member variable called ltc, which is a 2D lattice class (use the templated class defined in lattice.h). A lattice cell of ltc needs to be std::vector <FluidParticle *>.

(4) Add a member variable called YsVec2i ltcIdx; in FluidParticle class. This will be used for caching where in the lattice the particle is registered to.

(5) Add a member function called ResetLattice in FluidSimulator class. This function needs to first create a lattice that covers (minX,minY) to (maxX,maxY). The resolution of the lattice needs to be calculated so that the block dimension matches h. You can assume h is always non-zero. Then, clear all the cells of the lattice, and register particles to the lattice cells. For registration, you will need to calculate lattice index from particle position. Make sure to cache lattice index in ltcIdx member of the particle.

(6) Call ResetLattice function immediately after MakeTestCase.

(7) Add a member function called UpdateLattice in FluidSimulator class. This function must calculate lattice index for each particle, and if the index is different from the cached index, the particle needs to be removed from the previous cell (pointed by ltcIdx), and added to the new cell.

(8) Call UpdateLattice function immediately after moving particles.

(9) Modify FindProximityParticle function so that it uses lattice to find proximity particles. With this modification, your frame rate must get a boost.

6 Test Your Code on the Compiler Server

Test your source files (.cpp and .h files) on the compiler server. Some assignment may not require .h files. You do not have to test files that you don't make modifications. The files you need to test are the ones you write or modify.

We have four compiler servers:

- http://freefood1.andrew.cmu.edu
- http://freefood2.andrew.cmu.edu
- http://freefood3.andrew.cmu.edu
- http://freefood4.andrew.cmu.edu

Make sure you don't see red lines when you select your files and hit "Compile Test" button on the server.

We have multiple servers to make it less likely that all of them need to shut down for maintenance. If do not have to test on all of the servers. You need to make sure that your code passes on one of the servers.

7 Submit

Lastly, you need to submit using git. What you need to do are two things: (1) add files to git's control, and then (2) send to the git server.

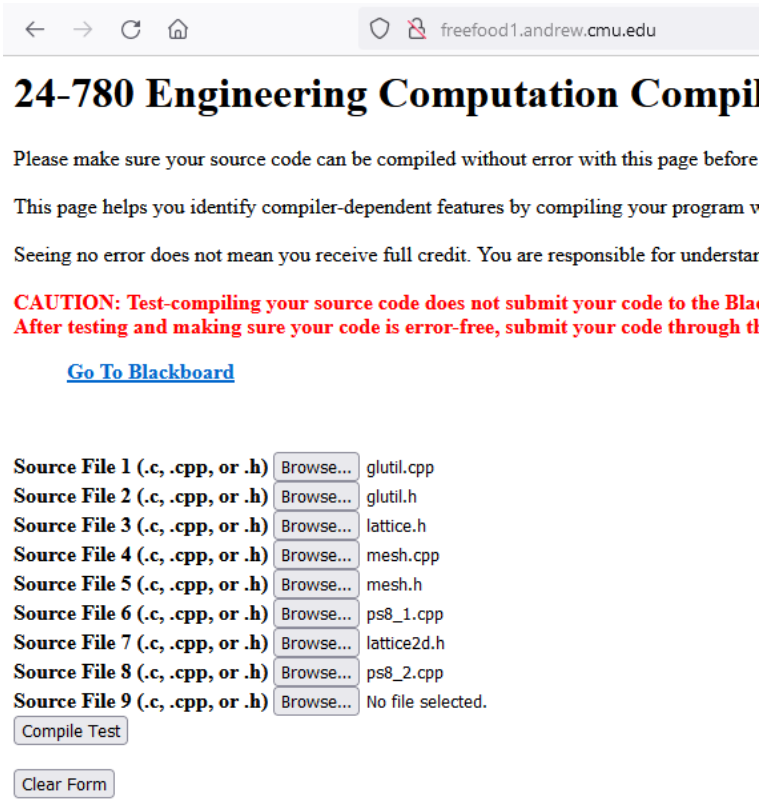

Fig. 2: Make sure to test your code with one of the compiler servers!

7.1 Add Files to git's control

In this case, you want to add all the files under ps8 subdirectory. To do so, type:

```
git add ~/24783/yourAndrewID/ps8
```

This command will add ps8 directory and all files under the subdirectories.

7.2 Send to the Git Server

In Git, sending files to the server is a two-step process. The first step is local commit. You can do it by:

```
git commit -m "Problem Set 8 solution"
```

The message can be anything, but it is recommended to type something meaningful, at least you can see what changes you made to your repository.

Local commit is just local. Git server does not know about any local commit unless the commit is sent (or pushed) to the server. To do so, type:

```
git push
```

Make sure to do it in the CMU network. If you are working from home (probably most likely), use VPN to connect to the CMU network.

You can re-submit (commit and push) your solution as many times as you want with no penalty before the submission due.

8 Verification

It is recommended to clone your repository to a different location and make sure that all of your files have been sent to the Git server.

You can do the following:

```
cd ~
mkdir 24783Verify
cd 24783Verify
git clone https://yourAndrewID@ramennoodle.me.cmu.edu/Bonobo.Git.Server/yourAndrewId.git
```

Once you made sure all the files have been submitted, you can delete files and directories under 24783Verify directory.