24783 Advanced Engineering Computation: Problem Set 7

In this assignment, you will write:

- Flood-fill on a mesh
- Rendering slices
- Using a lattice for fast intersection checking

(*) 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 ps7 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
       └─ps7
```

3 Make a CMake Project

Write CMakeLists.txt. This time you need only one CMakeLists.txt (top-level only). In the top-level CMakeLists.txt,

- Make sure to enable C++11
- Add subdirectory "../../public"
- Add a library called glutil, which uses glutil.cpp and glutil.h, and links ysclass library.
- Add a library called meshlattice, which uses lattice.h, meshlattice.h, and meshlattice.cpp, and links geblkernel library.

- Add a library called ps7lib, which uses ps7lib.cpp and ps7lib.h and links geblkernel, glutil, meshlattice, and fssimplewindow libraries.
- Add an executable called ps7, which uses main.cpp and links ps7lib.

You can compile the base code once you write your CMakeLists.txt. You need to start the program with a command-line parameter, which is an STL mesh file. The program will render the mesh in blue color.

4 Paint on Mesh

The main program calls:

```
Paint(mesh,plHd,YsPi/9.0);  // PI/9=20 degree
```

in RunOneStep function when the user clicks on a polygon. This function is written in ps7lib.cpp.

This function takes three parameters, mesh, polygon handle from which the flood-fill starts, and the angle tolerance.

Write this function so that it runs a flood-fill algorithm and change color of polygons to red (You can use an alias function YsRed()) for all contiguous polygons (you can exclude point contact) that has normal vector within the given tolerance from the starting polygon.

5 Show Slices of the Mesh

The program needs to take one of three modes:

- NORMAL mode: Render STL triangles
- SLICE mode: Show slices of the STL mesh
- SCANNER mode: Show intersecting points of the columnes of lines

You will work on the SCANNER mode after the SLICE mode.

Add lines in RunOneStep function so that the program switches to the SLICE mode when the user presses 'S' key, and NORMAL mode 'N' key. When the user presses 'S' key, your program must also call MakeSliceVertexArray function in ps7lib.cpp so that it caches the vertex array for rendering the slices.

Then, write MakeSliceVertexArray function in ps7lib.cpp. This function should calculate slices of the mesh with 100 planes that evenly divides between minimum-Y and maximum-Y of the mesh and parallel to the XZ-plane, and return a vertex array that can be drawn with GL_LINES.

The function in the base-code also prints the time taken for calculating slices. The program needs to complete the calculation within 1 second for c172r.stl in the course_files/data/binstl, or you lose 5 points.

Once the function is finished, fill code in the Draw() function in main.cpp so that slices are rendered in black color in the SLICE mode.

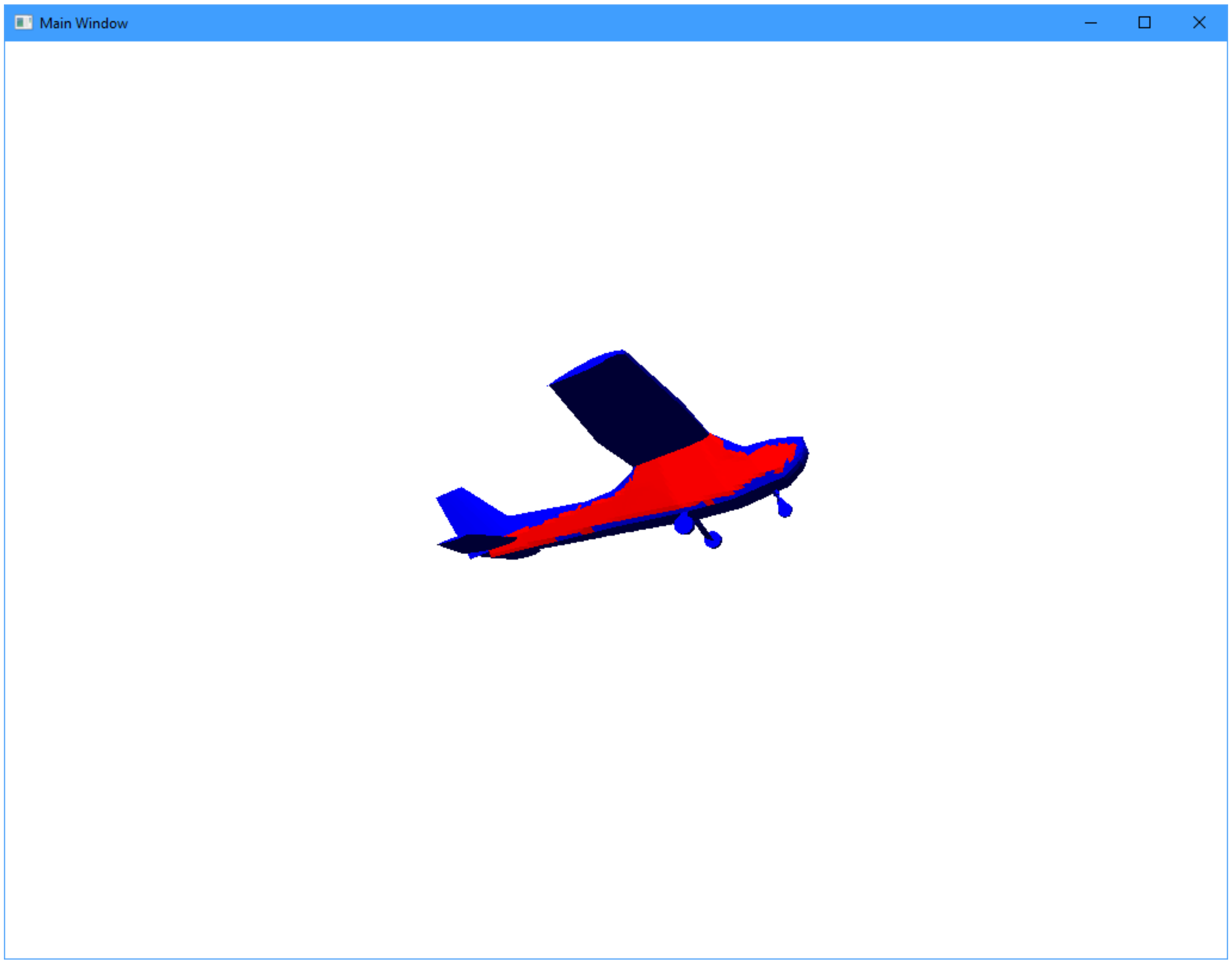

Fig. 1: Normal Mode

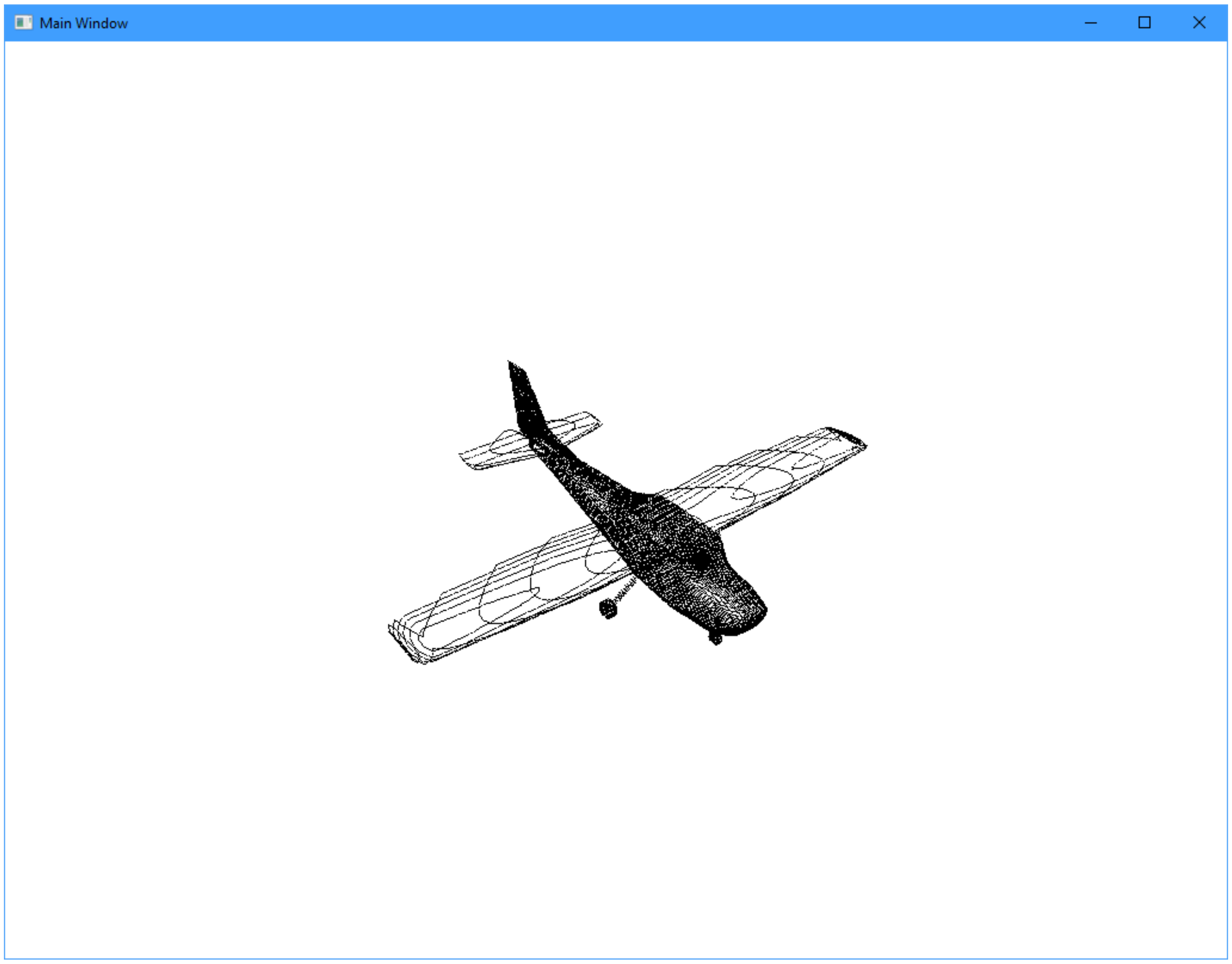

Fig. 2: Slice Mode

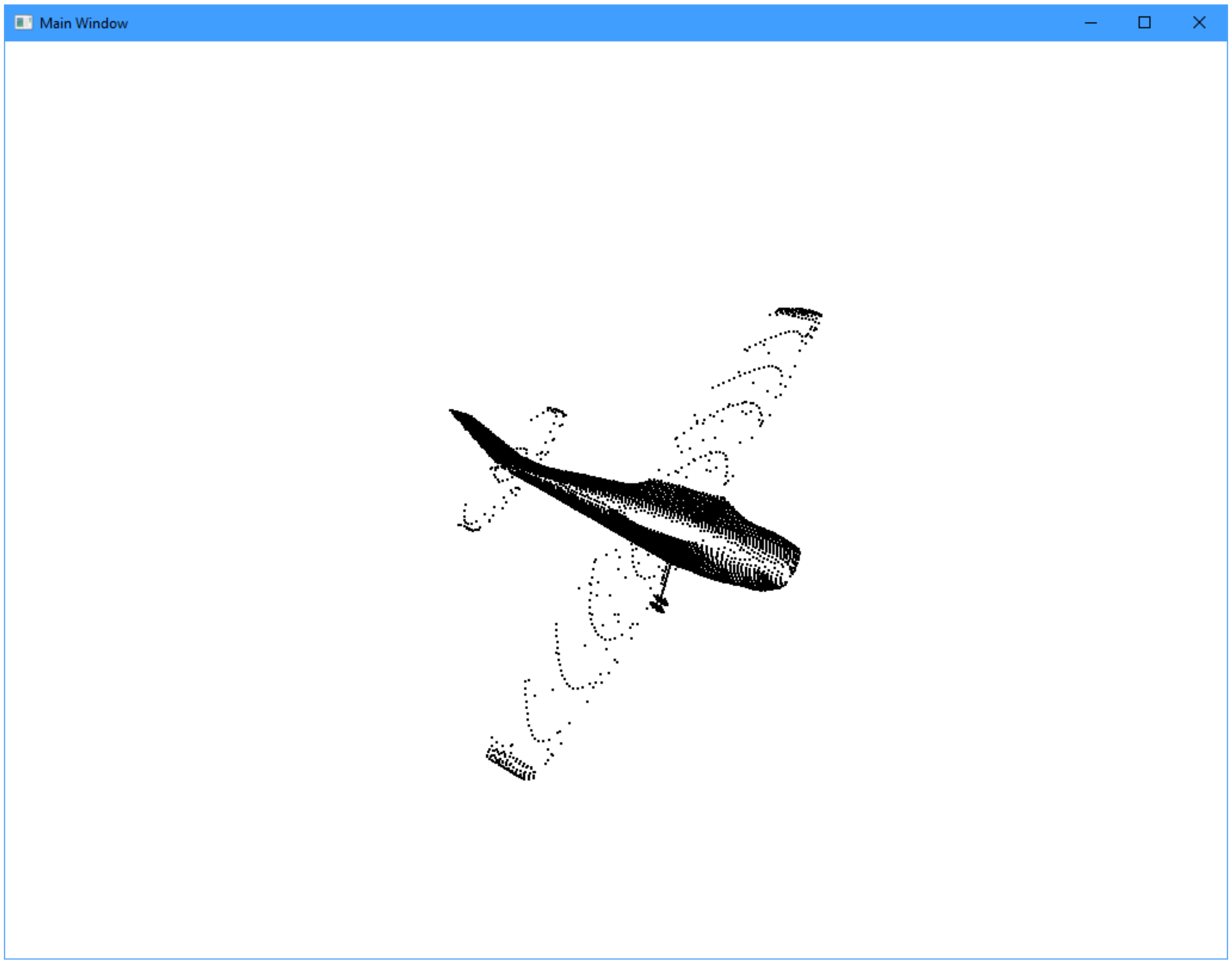

Fig. 3: Scanner Mode

6 Virtual Range-Scanner Mode

When the user presses 'L' key, the program must enter the SCANNER mode.

When the user presses 'L' key, the program should call MakeRayIntersectionVertexArray function in ps7lib.cpp and cache a vertex array for rendering intersections between columns of lines and the mesh with GL_POINTS.

Fill MakeRayIntersectionVertexArray in ps7lib.cpp. This function must return a vertex array that can ge rendered with GL_POINTS. Imagine you have a 100 rows and 100 columns of rays that are parallel to Z axis. Rays are evenly spaced between minimum and maximum X- and Y-coordinates of the mesh. In this function, calculate intersections with the rays and the mesh, and build a vertex array. Real laser range scanners can only take the intersection that is closest to the laser emitter, but in this program, you render all intersections.

One of the purposes of this assignment is to practice using a structured lattice for faster intersection detection. Use MeshLattice class in meshlattice.h and meshlattice.cpp. You can use SetDomain member function to register vertices and polygons to the lattice. For calculating intersections, see the function for finding a picked polygon.

Then, add your code in the Draw() function in main.cpp so that intersection points are rendered in the SCANNER mode.

7 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:24780
- http://freefood2.andrew.cmu.edu:24780
- http://freefood3.andrew.cmu.edu:24780
- http://freefood4.andrew.cmu.edu:24780

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.

8 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.

8.1 Add Files to git's control

In this case, you want to add all the files under ps7 subdirectory. To do so, type:

```
git add ~/24783/yourAndrewID/ps7
```

This command will add ps7 directory and all files under the subdirectories.

8.2 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 1 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.

9 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.