Project 2

Due Feb 10 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip and rar
Available Jan 22 at 10am - Feb 13 at 6am 22 days

This assignment was locked Feb 13 at 6am.

See a demo of this project here: **Project 2 Demo** (No audio)

See screenshots of the .ply files here: **Project 2 Images**

Overview:

In this project, you will load a mesh stored in the .ply file format, render it as a 3D wireframe model using Vertex Buffer Objects and also add keyboard control that lets us move the .ply file around.

Part I: Parsing and Drawing

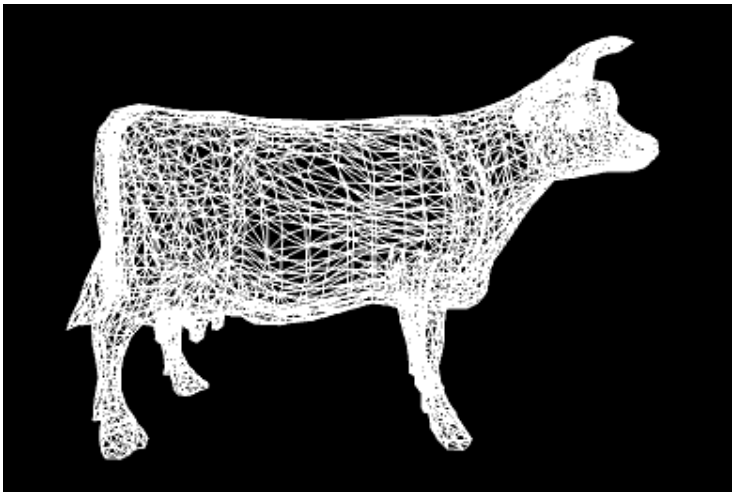
Begin this project by writing a simple WebGL program that can read in a .ply file and render the drawing to the screen. To read in a .ply file, you should use a file upload button like you did in Project 1. The format of a .ply file is as follows:

File Text	**What you should do**
Ply	If not present, exit
Format ascii 1.0	Skip line
element vertex 758	Read # of vertices (758)
property float32 x property float32 y property float32 z	Skip these lines
element face 1140	Read # of polygons (1140)
property list uint8 int32 vertex_indices	Skip line
end_header	End of header section; skip line
6.5 -7.2 1.1 0.5 0.8 -1.5 1.2 9.0 5.5	Coords of Vertex #0 Coords of Vertex #1 Coords of Vertex #2

etc.	And so on, until Vertex #757
3 1 9 8 3 5 10 5 3 7 0 9 etc.	First number is the number of vertices in the polygon. In our case, it's always 3. Then the next three numbers tell you which vertices make up that polygon. So, triangle #1 is made from vertices #1, #9, and #8._

You can get 44 .ply files to work with here: **cs4731_pjt2_ply_files.zip**

Your rendered drawing will be a mesh. Here's an example of what my program shows when I upload the cow.ply file:



Notes:

- Start testing your program with the cube.ply file. It's much simpler than the other meshes, so it will be easier for you to diagnose and debug problems.
- You should be able to use a fair amount of your Project 1 code for this part of the project.
- Note that the .ply files do NOT offer extent boundaries like the .dat files in Project 1. You will have to use the vertex information to figure out your own extents. Remember that you still need to ensure that all files observe the correct aspect ratio!
- Please use a perspective projection, not an orthographic one.
- After you have determined the extents of the mesh, you should use the lookAt() function to move your camera some fixed distance away from the mesh. The edges of the mesh should not be flush against the edges of your canvas!
- These meshes are 3D, so you will need to account for depth in your program.
- For ease of legibility, I recommend a white line color against a black background.

Part II: Pulsing

First, calculate the normal of each mesh face (i.e. each polygon) using the Newell method.

Then, create a pulsing animation by translating each face some fixed amount in its normal direction. By linearly interpolating the position of each vertex belonging to a given face between its original position and v+cn (where c is a constant) and then interpolating back in the opposite direction, we can make the mesh

bulge outward and then recede in a smooth fashion. This operation should make the meshes look like they are "breathing" back and forth.

Notes:

- Note that when the mesh breathes in, the faces of the mesh are in their original positions, and when the mesh breathes out, the faces move outwards and temporarily separate from neighboring faces.
- Make sure the faces move out enough for the bulging effect to be noticeable and make the face movements nice and smooth (Not too fast).
- You will need to use current transformation matrices (CTMs) on each face to translate it accordingly.

Part III: Keyboard Controls

Implement the following keyboard controls in your program:

- **User hits 'X' (Translate your wireframe in the positive x direction)** Continuously animate your wireframe some small units along the positive X axis. The ply file should continue to slide along until the user hits 'X' again. Essentially, the 'X' key acts as a toggle key. The speed is up to you, but it should not be too fast or slow.
- **User hits 'C' (Translate your wireframe in the negative x direction)**
- **User hits 'Y' (Translate your wireframe in the positive y direction)**
- **User hits 'U' (Translate your wireframe in the negative y direction)**
- **User hits 'Z' (Translate your wireframe in the positive z direction)**
- **User hits 'A' (Translate your wireframe in the negative z direction)**
- **User hits 'R' (Rotate your wireframe in an X-roll about it's CURRENT position)** Just like a rotisserie chicken, continuously animate your wireframe smoothly 360 degrees at a moderate speed (X-roll) about its CURRENT position (not about the center of the scene). This rotation is NOT the same as moving the wireframe in a wide arc. The rotation should be about the X axis.
- **User hits Key 'B': Toggle pulsing meshes ON/OFF.** When ON, the mesh faces pulse back and forth continuously as described above. When OFF the meshes do not pulse.

Notes:

- The camera position should remain fixed even if the mesh is moving.
- The model should be able to translate, rotate, and pulse at the same time. It only needs to translate in one direction at a time.
- If a translation, rotation, or pulse is stopped and started again, the model should pick right back up from where it stopped (i.e. do not "reset" the model).
- Your animation may run at different speeds depending on the complexity of the mesh you're using. This is normal given the amount of work that your CPU and GPU have to do for each frame. However, please make a point of minimizing the amount of redundant work your program does. For instance, you should only be calculating and passing your projection matrix once each time you select a file.

Extra Credit:

For this project, you are allowed to include additional features above and beyond the basic requirements. The instructor has sole discretion over the point value of each feature, and the student may earn up to 10

points total. To earn any credit, *you must list each feature in the comments at the top of your main *.js file.*

To be eligible for extra credit, the additional features must be graphical in nature. This means that you show thought behind how something is presented to the user on screen or how the graphics are being processed behind the scenes.

Here are some example features to consider:

- Increment/decrement the amount of shearing of the mesh along the x axis.
- Toggle on and off a drawing of each face normal. Each face normal is drawn as a short line starting from the middle of each face and extending outwards. When you draw all normals, it will look like the mesh has pins sticking out of each face

You are not limited to these ideas. If there is something else you would like to do but are not sure if it would qualify for extra credit, please contact me.

Submitting Your Work:

Make sure to double-check that everything works before submitting. Put all of your files (JavaScript, HTML, etc.) into a folder and zip it. Please include your *.ply files for ease of grading. Upload it to Canvas. Do not email me your program or submit it via a third-party cloud storage account.

Create documentation for your program and submit it along with the project inside the zip file. Your documentation can be either a pure ASCII text or Microsoft Word file. The documentation does not have to be long. Briefly describe the structure of your program and what each file turned in contains. Name your zip file according to the convention *LastnameFirstname_Pjt2.zip*.

Additional Notes:

- You are free to consult any resources you need to help you complete this assignment, including your classmates, the TA, the instructor, the class text, and any other books and websites. However, the code you turn in must be your own. ***Any evidence of plagiarism will result in an automatic 0 for this assignment.***
- You are welcome to use any class coding examples (posted under "Modules") in your program.

FAQ:

Q: How do I determine my extents? They're not in the .ply files.

A: Correct. Use your vertex information to calculate what your extents should be. Also, don't assume that your mesh is necessarily centered at the origin.

Q. How do I calculate my FOV and camera (eye) position?

A: Once you know your bounding box (your extents), you can use those to help you calculate FOV and camera location. Since the perspective projection function also takes in an aspect ratio, your model should not get distorted.

For your camera (eye) position, think about how you can use your extents to help you determine how far back you should pull your camera from the mesh. There's no one right way to do this, so please pick a

method that makes logical sense to you.

Once you know your camera position, you can calculate your FOV using your camera position, your extents, and some trigonometry. Take a look at the image below:

Q: I'm getting weird lines that connect my meshes. Each mesh looks like a marionette.

A: If you try to draw your entire vertex array at once, WebGL will treat it as one continuous polygon.Try making separate calls to `drawArrays()` for each polygon in your mesh. This will also be important for your pulsing animation, as each polygon will have a different normal and thus a different translation matrix.

Q: How fast should my meshes be drawn to the screen?

As shown in the demo video, the more complex animations will take a second or two. However, if the drawing process is taking more than about 5 seconds, please go back and try to reduce the number of computations your program makes. Are there things that can be pre-computed ahead of time?

If all else fails, try running your program on a newer and/or more powerful computer, especially if you are on a laptop that is more than 3 years old. This is a computationally expensive program, so it may overtax the hardware resources of some older machines.

Q: Should the model be able to translate across multiple axes simultaneously?

No. The model will only be able to move along one of the three axes at any given time. If I press "X" to translate along the positive x-axis and then press "Y", for instance, the model will simply stop translating. Pressing "Y" again will cause it to translate along the positive y-axis.

Project 2 Rubric

Criteria	Ratings		Pts
File Parsing - Program has a place to upload *.ply files (1 pt) - Program parses *.ply files (3 pts) - Ignores lines where appropriate (3 pts) - Parses out, converts, and stores numbers correctly (3 pts)	**10.0 pts** **Full Marks**	**0.0 pts** **No Marks**	10.0 pts
File Drawing (4 pts each) - Entire picture drawn to screen immediately - Aspect ratio preserved - Appropriate, mesh-specific perspective projection matrix - Enough contrast in colors between foreground and background for the image to be visible - Appropriate view matrix	**20.0 pts** **Full Marks**	**0.0 pts** **No Marks**	20.0 pts
Pulsing Animation - Normals calculated using Newell method (9 pts) - Interpolation used to calculate animation path for each face (8 pts) - Pulsing animation is visible but not extravagant (8 pts)	**25.0 pts** **Full Marks**	**0.0 pts** **No Marks**	25.0 pts
Other Animation - Pulsing, rotation, and translation can all occur at the same time (5 pts) - Rotation occurs around the x axis (2 pts) - Radius of rotation does not change when object translates (3 pts) - Camera does not move (5 pts) - Program avoids redundant calculation where possible (5 pts) - Animation can stop and start again from the same state (5 pts)	**25.0 pts** **Full Marks**	**0.0 pts** **No Marks**	25.0 pts
Keyboard (5 pts each) - All keyboard commands implemented - Each key toggles its respective animation on and off	**10.0 pts** **Full Marks**	**0.0 pts** **No Marks**	10.0 pts
Documentation and Organization (5 pts each) - Code is clean, well-commented, and easy to read - Documentation file describing structure of program and what it contains	**10.0 pts** **Full Marks**	**0.0 pts** **At least one of documentation file and code comments is missing**	10.0 pts
Extra Credit	**0.0 pts** **Full Marks**	**0.0 pts** **No Marks**	0.0 pts
Late Submission	**0.0 pts** **Full Marks**	**0.0 pts** **No Marks**	0.0 pts
			Total Points: 100.0