# CSE 167

DISCUSSION 1

# Announcements

- Private posts are for emergencies or a question that explicitly shows a part of your code
- Office hour schedule is up on Piazza
- Project 1 is due this Friday at 2pm
  - You may submit to TritonEd as many times as you want before the deadline
  - Grading will be done in B260 and B270
  - Write your name and station number on the board: you will be graded FIFO

# Contents

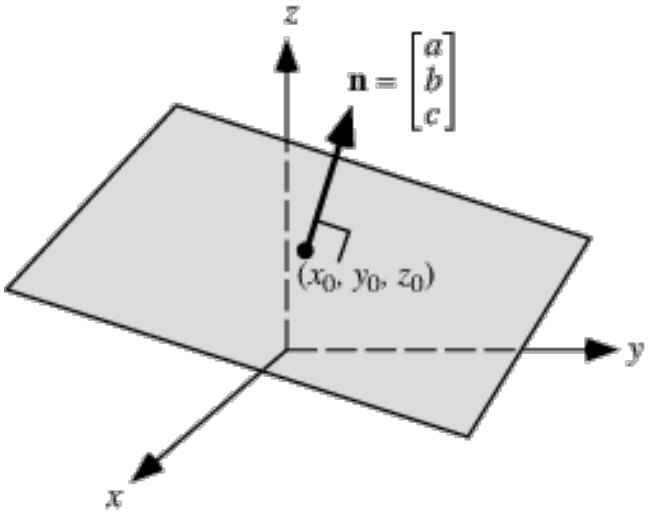
- Starter code overview
- Parsing .obj files
- Extra credit

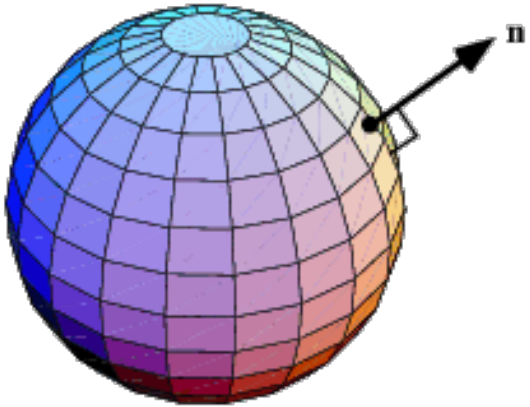
# Starter code overview: Window

- Window::initialize_objects()
  - DON'T parse an object every time you switch, parse them all at once in the beginning
    - bear.obj alone has 866,394 vertices, if you parse it every time you switch back to bear, it's going to take a lot of time
- Window::idle_callback()
  - You need to spin bunny/dragon/bear constantly; that means you need to do something here
- Window::display_callback()
  - Here, you will call the draw() function that renders the image ~60 times/second
- Window::key_callback()
  - You specify what to do when a certain key is pressed

# Normal Vectors

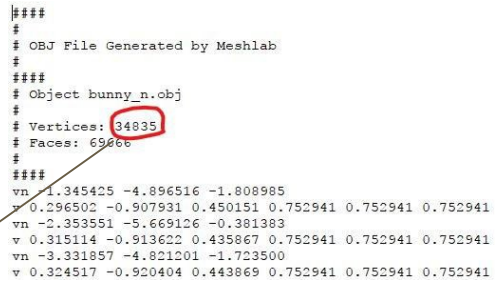
The normal vector, often simply called the "normal," to a surface is a vector which is perpendicular to the surface at a given point.



# Starter code overview: OBJObject

- What is an “object”?
  - A “thing” that you would like to place in your scene
- Where to create an object?
  - The same place where cube was created
- What functions need to be in OBJObject?
  - parse(): for parsing .obj files
    - Make sure you are NOT parsing vertex normals as vertices!!!
    - The number of vertices specified in .obj file should be the same as the number of parsed vertex lines
  - some_function(): for spinning the object



# OBJ Parser

MeshLab can load and display OBJ files: http://www.meshlab.net/

OBJ files are ASCII files and can be loaded **and edited** with a text editor.

Tip: create your own file with just a few points for testing while you develop your parser.

# OBJ File bunny.obj: Beginning

```
####
#
# OBJ File Generated by Meshlab
#
####
# Object bunny_n.obj
#
# Vertices: 34835
# Faces: 69666
#
####
vn -1.345425 -4.896516 -1.808985
v 0.296502 -0.907931 0.450151 0.752941 0.752941 0.752941
vn -2.353551 -5.669126 -0.381383
v 0.315114 -0.913622 0.435867 0.752941 0.752941 0.752941
vn -3.331857 -4.821201 -1.723500
v 0.324517 -0.920404 0.443869 0.752941 0.752941 0.752941
vn -1.349485 -5.093389 -2.572747
v 0.325879 -0.966814 0.485578 0.752941 0.752941 0.752941
vn -0.345145 -6.207325 -0.638455
```

# OBJ File bunny.obj: End

f 16601//16601 16546//16546 34835//34835
f 16546//16546 16324//16324 34835//34835
f 16324//16324 16323//16323 34835//34835
f 16323//16323 16322//16322 34835//34835
f 16322//16322 20914//20914 34835//34835
f 20914//20914 20916//20916 34835//34835
f 20916//20916 22065//22065 34835//34835
f 22065//22065 12707//12707 34835//34835
f 12707//12707 33423//33423 34835//34835
# 69666 faces, 0 coords texture

# End of File

# Extra Credit

1) Create a visual effect to bring up the points of a model when initially loaded: for instance, randomize their initial position within a cube the size of the model, then move them gradually to their final locations by linearly interpolating between their randomized positions and their locations in the model. (5 points) For inspiration see the first few seconds of this video and imagine watching it in reverse.

2) Alternative point coloring: implement a different way to color the points, and support a key to switch between normal coloring and your alternative coloring. (5 points) Examples:

- Color them based on their distance from the 3D model's coordinate origin (green is at origin, red is farthest from it, linearly interpolate between green and red for points in-between).
- Color them based on the point's x/y/z coordinates (x->red, y->green, z->blue).

3) Create transitions between models: morph the points from one model to the next when switching between models. (10 points)

4) Variable point size: render larger points when they are closer to the viewer for a more realistic 3D effect. Follow the instructions at: https://learnopengl.com/Advanced-OpenGL/Advanced-GLSL (10 points)