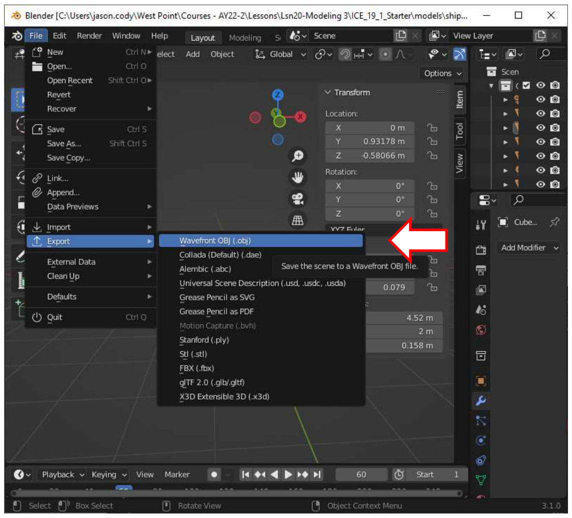
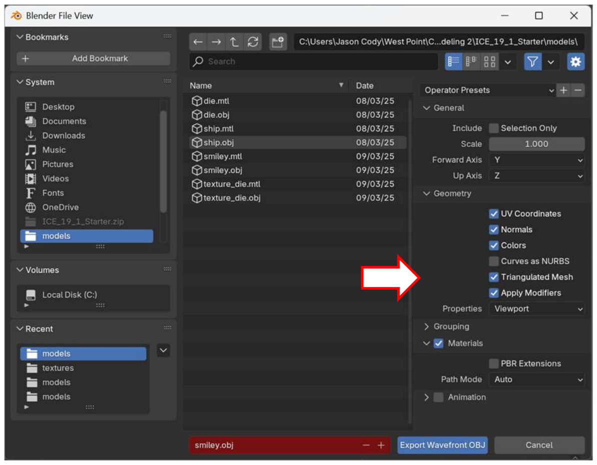
ICE 19.1: Importing Your Model

Start with the Lesson starter code (see new additions below).




Blender File View
C:\Users\Jason Cody\West Point\C...deling 2\ICE_19_1_Starter\models\
smiley.obj
Export Wavefront OBJ
Cancel

Operator Presets

General

Include: ☐ Selection Only
Scale: 1.000
Forward Axis: Y
Up Axis: Z

Geometry

☑ UV Coordinates
☑ Normals
☑ Colors
☐ Curves as NURBS
☑ Triangulated Mesh
☑ Apply Modifiers
Properties: Viewport

Grouping

☑ Materials

☐ PBR Extensions
Path Mode: Auto

☐ Animation

1. Review the starting code. Some important features are below:
    a. Shader programs have been reformatted to allow the input of a shader_state integer used to adjust the behavior of the shader depending upon what is being drawn. In main.cpp, an enumerated type (ObjectType) is used to set this value before each object is drawn. This is an example of how to intuitively change shader behavior.
    b. import_object.hpp/cpp: Review the header file and take careful note of the following member functions:
        i. The constructor for ImportOBJ facilitates model loading.
        ii. ImportOBJ::loadFiles takes a file name (no extension!) and a VAO with vertex position, normal, texture coordinate, color, and specular color defined and returns a BasicShape object. ImportOBJ::loadFiles calls private functions for reading the .mtl (material) and .obj (object or vertex data file). Examine these to see how it is done. Some cadet projects need to modify these.
        iii. The structure CompleteVertex is used to store all the data for a single vertex.
    c. main.cpp:
        i. Three VAOs are created (one for regular objects, one for textured standard objects, and one for imported objects).
        ii. Three BasicShape objects are defined using the ImportOBJ::loadFiles function.
        iii. A BasicShape object, the floor, is also defined.
        iv. The imported objects are drawn in the render loop. In the next lesson, you creating a class to manipulate this object instead of cluttering the render loop.
        v. After the render loop, the BasicShape::DeallocateShape function is used to delete VBOs.
2. Open the blender file you want to import into OpenGL. Then select File>Export>Wavefront(.obj). See image above (left side).
3. When the export screen appears typically use the following options (see image above right):
    a. UV Coordinates

   b. Normals (adds normal for lighting in the .obj file)
   c. Colors
   d. Triangulate Mesh (creates faces that our Importer class requires).
   e. Materials (creates the .mtl file)
4. Choose an appropriate location (./models) and press Export to create a .obj file and a .mtl file.
5. In your main program:
   a. Either create a new importer object (from class ImportOBJ)…or re-use the importer object already in the program to import your model using the ImportOBJ::loadFiles member function (which returns a BasicShape). Note that loadFiles must be called after the environment is initialized.
   b. IF your model has a texture applied to it, you will need to do the following:
      i. Open the models .mtl file and change the path to the texture file so that it is accurate (see the example die.mtl).
      ii. BEFORE loading another model with the same importer, make sure you store the generated texture (unsigned integer) for later binding.
6. Draw the imported object:
   a. Review the shader program. Note that the shader is set up to either use textures (die), materials (smiley), or a color you set (ship).
   b. Draw the imported object inside your render loop (Ensure that you set the coordinate transformations so that your object is rendered).
7. **CHALLENGE:** If you didn't use a textured object imported from blender, create one that does. If you did, create one that doesn't and import it – make sure you understand this process.