Project 5: Subdivision and Geodesic Spheres

Due Dec 6 by 11:59pm   **Points** 100   **Submitting** a file upload
Available Nov 22 at 12:01pm - Dec 11 at 11:59pm

This assignment was locked Dec 11 at 11:59pm.

Objective

This project is about modifying surface meshes. In particular, you will write a program in the Python version of Processing that reads in polyhedral models and creates geodesic surface using triangle subdivision. Your program will also demonstrate the "corners" representation by visualizing corner operations such as next, opposite and swing. All of the mesh objects that you create and modify will be made of only triangles.

More Flexible Late Policy

For this project only, you can turn in the project up to five days late. Each late day will incur a 3-point penalty instead of the usual 5-point penalty. This late policy does not apply to any of the other projects in the course.

Project Description

The provided code draws a single triangle on the screen. This triangle can be rotated by clicking and dragging the mouse in the window. The provided code also reads a mesh from a file and prints this information on the screen.

Your finished program will be able to read a mesh from a file and display that mesh. More importantly, with a keystroke your program will create a subdivided version of the mesh, replacing the old mesh in memory. Your program should also show corners adjacency information by visualizing a current corner as a small sphere. Your program will respond to the following keystroke commands:

1-4: Read in a mesh file (tetrahedron, octahedron, icosahedron, star).

d: Create the triangle subdivided current mesh (you should be able to do this more than once).

i: Inflate the points of the mesh so that they lie on the unit sphere.

r: Toggle between white and randomly colored faces.

c: Toggle between showing and not showing the current corner as a sphere.

n: Change the current corner using the "next" operator.

p: Change the current corner using the "previous" operator.

o: Change the current corner using the "opposite" operator.

s: Change the current corner using the "swing" operator.

The subdivision that you will implement does not actually produce what is known as a subdivision surface. What you will implement is slightly less complex than making a subdivision surface, because the rule for placing the new vertices on edges is simple, and because you will not be moving the original vertices. You will place the new vertex for an edge at exactly the midpoint of the edge. For each original triangle, you will use its three original vertices together with the three new vertices at the edge midpoints as the vertices of four new triangles that replace the original triangle.

A large part of this project is being able to perform triangle subdivision of the surface more than once, so **make sure** your program can do this.

You should initialize the "current corner" to be zero. The display of the current vertex should be toggled by the "c" key. Other keys will change the current vertex to be other corners, through the next, previous, opposite and swing operators. Be sure that the sphere that represents the current corner is clearly at the **corner** of one particular triangle. It is **not sufficient** to draw a sphere at a vertex, since this doesn't show which triangle the corner belongs to. Remember that the swing operator (key press "s") should move the current corner around a vertex, and that this motion should be visible as you travel between corners.

When random colors are turned on, the faces of your model should be colored at random. These random colors should not flicker, but instead should remain steady from one frame to the next.

Please make sure your project has the ability to rotate the current mesh by clicking and dragging with the mouse.

Provided Code

Use the **PROVIDED CODE** **(https://gatech.instructure.com/courses/264908/files/37197747/download)** ↓ **(https://gatech.instructure.com/courses/264908/files/37197747/download?download_frd=1)** to start on this project. The provided code shows how the mesh files are read in line-by-line, but you will need to take the mesh data and store it in the "corners" mesh data structure.

File Format

The file format we will use for this project is a simple text format for meshes. The file format is basically an indexed face set. The first two lines specify the number of vertices and faces of the model, respectively. Then all of the vertex (x,y,z) values are listed, one per line. This is followed by the list of faces, one face per line. The first number that describes a face is simply the number of sides of that face, and for this project, that number will always be 3. Following this, indices into the vertex list are given. The vertices are indexed starting from the value zero. The mesh files can be found in the "assets" folder that is give together with the provided code. You can open these files in any text editor.

Suggested Approach

A key part of this project will be using a representation of the polygon mesh that will easily allow you to move from a face to other adjacent faces. One good representation of the polygon mesh is the "corners" representation. You can find Jarek Rossignac's slides of this representation **HERE (https://gatech.instructure.com/courses/264908/files/36989775?wrap=1)** . (Only the first 10 slides of these notes are relevant to this assignment.) Note that all of the meshes for this project contain only triangles, so the corners representation is appropriate. Keep in mind that you will need mesh adjacency information to create the triangulated dual mesh and also to calculate per-vertex normals.

First, modify the skeleton code to read in the .txt file format into your own polyhedral model data structure. Next, modify the "draw" routine to draw the current polyhedral mesh. Next, add a toggle that switches between a white model and randomly colored faces.

The key to this program is to use the "corners" representation for triangle meshes. Make sure you store the triangles as one long list of references to the vertices. The length of this list should be three times the number of faces. Once you have done this, create a "current corner", and display that corner as a small sphere. Then implement the "next" and "previous" corner operators, and cause the keys "n" and "p" to move the current corner using these operators. The next task is to implement the "opposite" operator on a corner. This requires that you find out which corners are paired by virtue of being opposite each other across an edge. Once you have opposites implemented, adding the "swing" operator should be easy.

The most challenging part of this assignment is being able to subdivide the triangle faces into four sub-triangles. You should do this by creating one new vertex along each edge of the mesh, and then replace each triangle by four smaller triangles that use both old and new vertices. There are several steps to performing this subdivision. The first step is to create one new vertex for each edge of the mesh. You need to avoid creating two new vertices per edge, but instead share such edge vertices between the two polygons that meet at a given edge. You don't need to make a new list of vertices, but instead can simply add the edge-associated vertices to your current list of vertices. Next, you should create a new list of triangles, replacing each original triangle with four new, smaller triangles.

Note that you should be able to subdivide a given mesh more than once. Each time you calculate the subdivision, this will result in a mesh that has more triangles than the previous one.

Finally, you should implement the "inflate" mesh operation. This is a fairly easy task, since all of the objects are centered at the origin. Each vertex of the mesh should be moved along a line that connects its current position with the origin (0, 0, 0). Move each vertex to be on the surface of a sphere with a radius of one. When coupled with subdivision, this will allow you to make geodesic spheres.

Results

Below are images that show how you should visualize the corners operators. In the top row, one corner is illustrated by placing a small sphere in the corner. The remaining four images show where this corner would be moved by each of the following operators: next, previous, opposite, and swing, respectively.
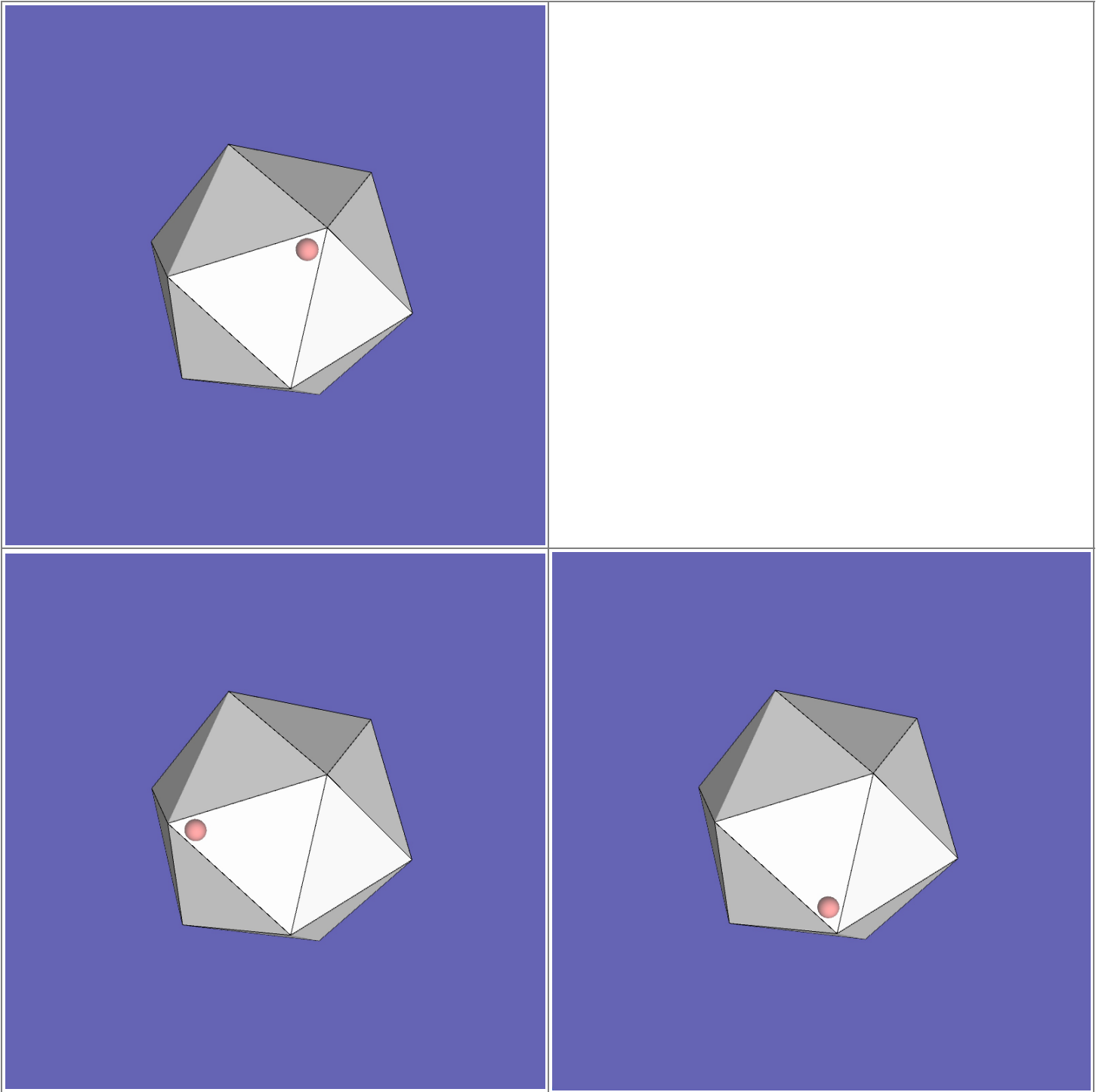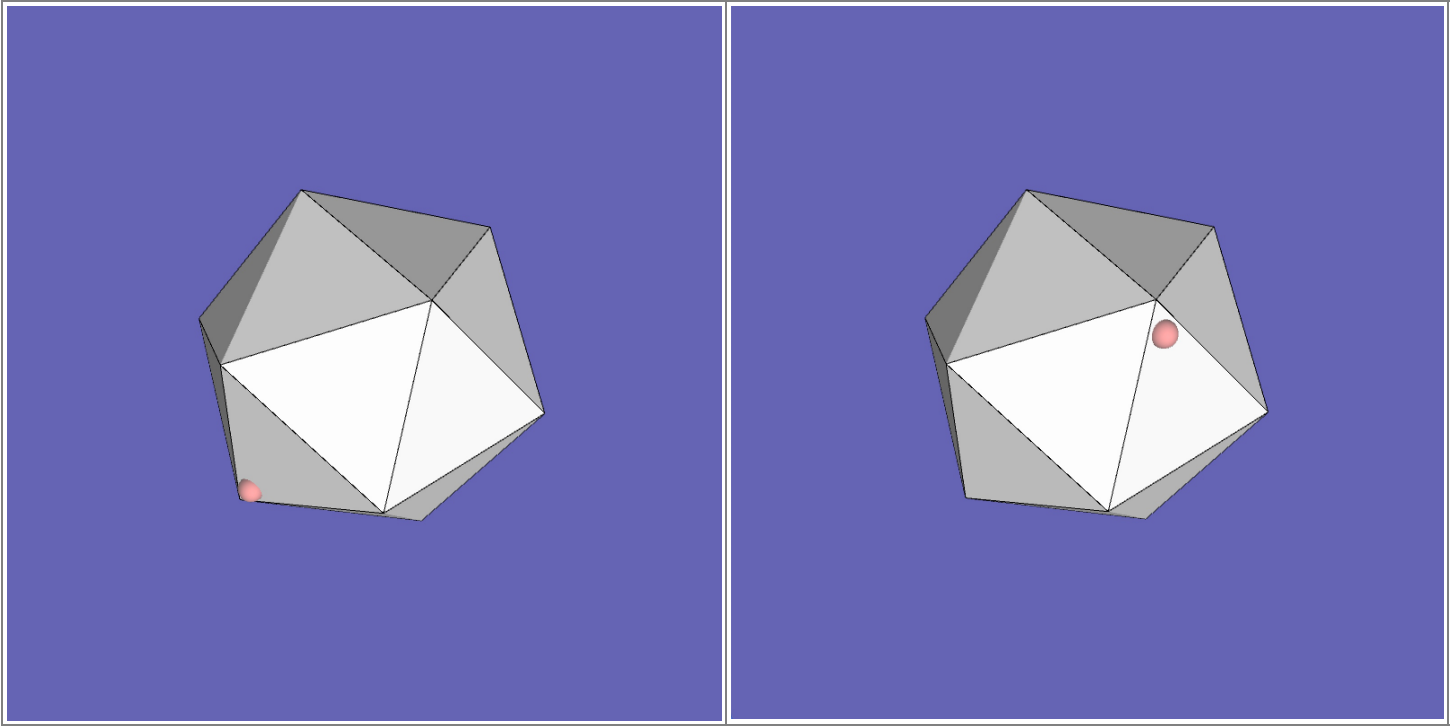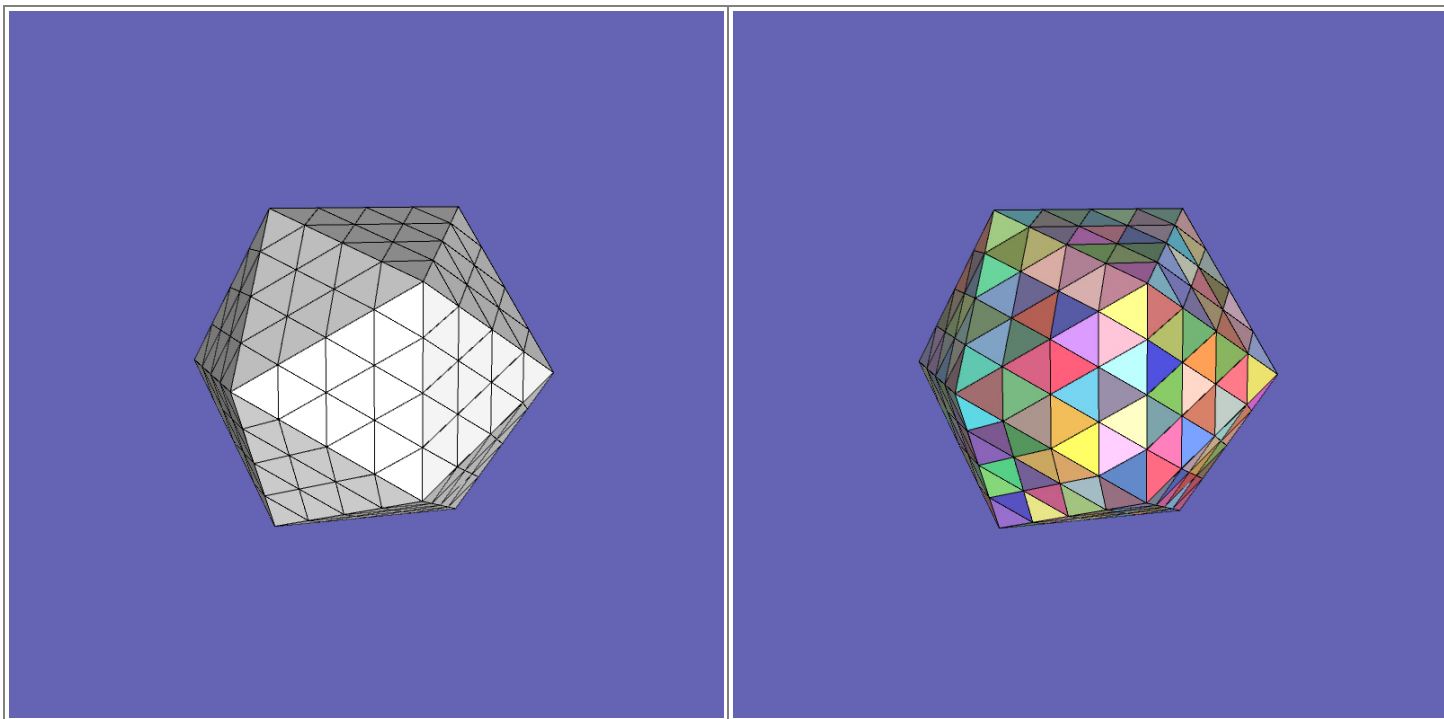
Each line below shows an original object, the object subdivided once, subdivided twice, and subdivided twice and inflated.
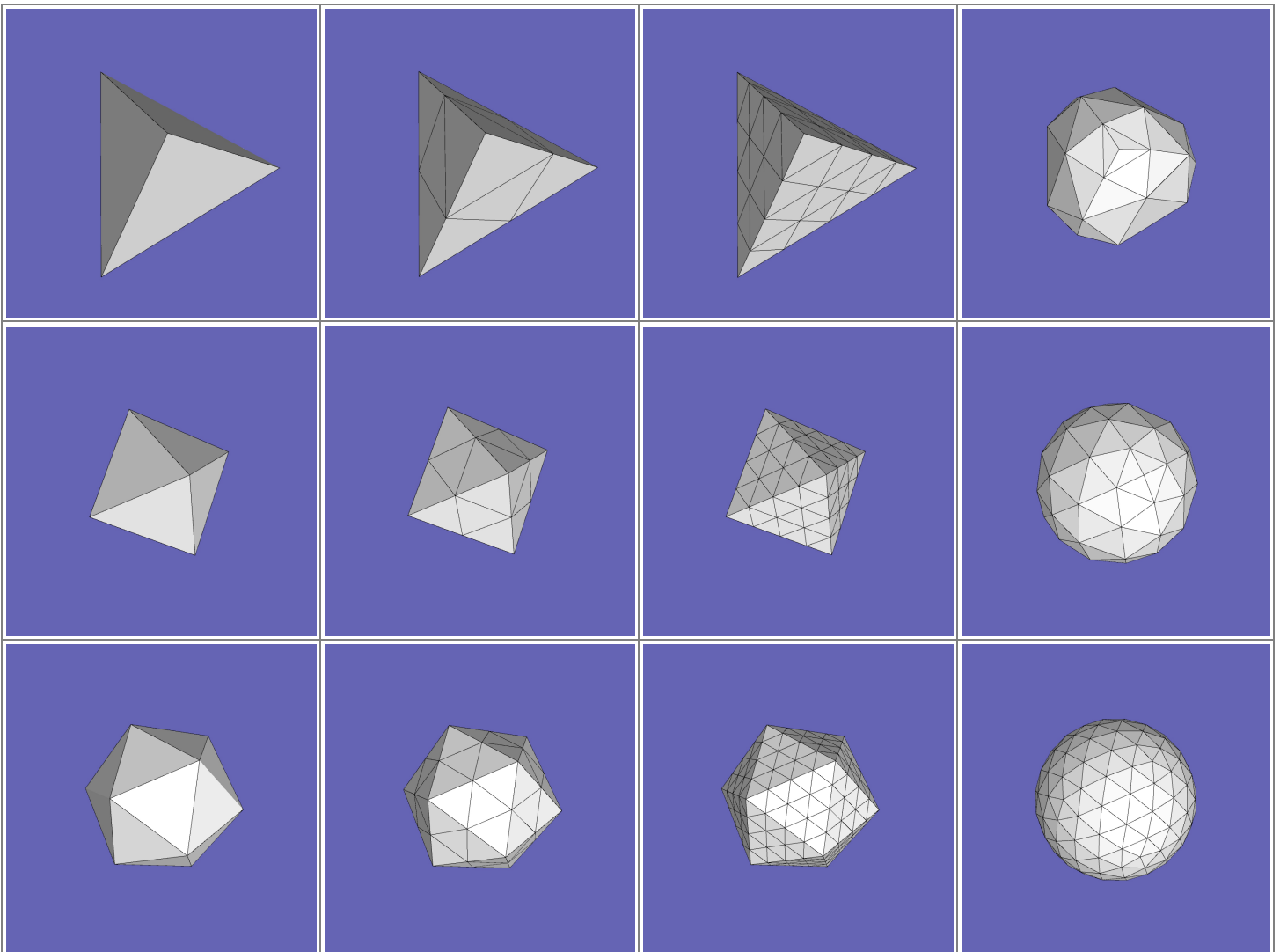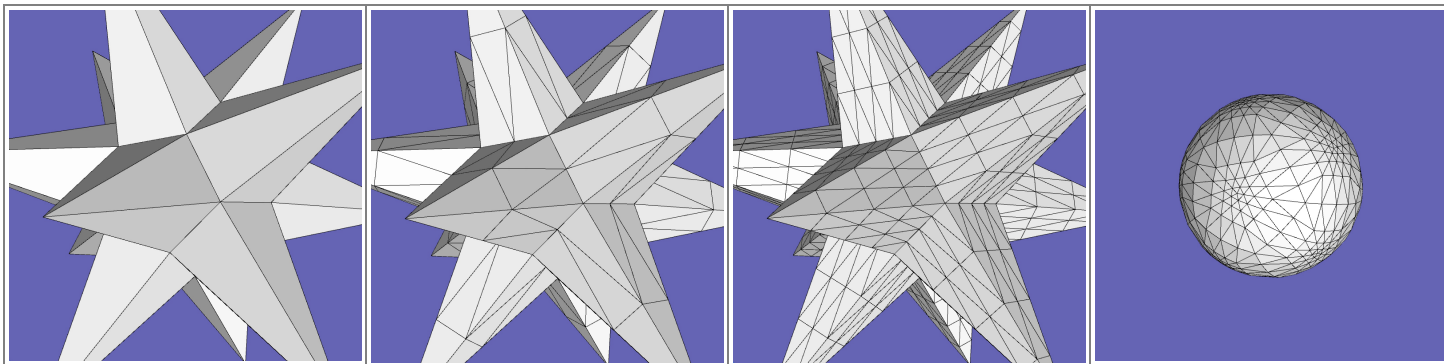
Below shows a subdivided icosahedron, without and with randomly colored faces.

Authorship Rules

The code that you turn in entirely your own. The instructor and the TA's are available to answer your questions about this project. You are also allowed to talk to other students the class about high-level implementation strategies. It is also fine to seek the help of others for general python and Processing programming questions. You may not, however, use code that anyone other than yourself has written. The only exception is that you should use the example source code that we provide for this project. Code that is explicitly **not** allowed includes code taken from the Web, github, from books, from the py.processing.org web site, from other assignments, from other students or from any source other than yourself. You should not show your code to other students. Feel free to seek the help of the instructor and the TA's for suggestions about implementing the project and debugging your code.

Submission

In order to run the source code, it must be in a folder named after the main file. Please also include the provided mesh files in the data sub-directory. When submitting any assignment, leave it in this folder, zip

it and submit via Canvas. Please do **not** use tar or rar to turn in your files.