Geometric Processing Tools 几何处理工具

A1. Directed_Edges 有向边处理工具

Functionality 功能

1. 将 .tri 文件转换为 .face 文件
2. 将 .face 文件转换为 .tri 文件
3. 检查模型是否为流形并计算属数

Notes 注意事项

en:

1. Please place the files face2faceindex.cpp, faceindex2directededge.cpp, and MeshManifoldCheck.cpp in the same directory as the handout_models folder so that the code can run correctly.
2. The time complexity and space complexity of the algorithm are documented in the header comments at the beginning of each .cpp file.
3. This code will process all .tri files within the handout_models folder in one go. The progress will be displayed in the command line.
4. When executing MeshManifoldCheck, the command line will output the first item (edge or vertex) that does not satisfy the non-manifold condition. It will also display the number of vertices, faces, and edges in the manifold file.

zh:

1. 请将文件 face2faceindex.cpp、faceindex2directededge.cpp 和 MeshManifoldCheck.cpp 放在与 handout_models 文件夹相同的目录中，以便代码能够正确运行。
2. 算法的时间复杂度和空间复杂度记录在每个 .cpp 文件开头的注释中。
3. 该代码将一次性处理 handout_models 文件夹中的所有 .tri 文件，进度将在命令行中显示。
4. 在执行 MeshManifoldCheck 时，命令行将输出第一个不满足非流形条件的项目（边或顶点），并显示流形文件中的顶点、面和边的数量。

How to Run My Code 使用方法

en:

1. Start by using the `make` command to execute the Makefile and compile the three C++ files.
2. Execute the `./face2faceindex` command to run the `face2faceindex` executable, which converts .tri files to .face files.
3. Execute the `./faceindex2directededge` command to run the `faceindex2directededge` executable, which converts .face files to .tri files.
4. Execute the `./MeshManifoldCheck` command to run the `MeshManifoldCheck` executable, which checks whether the model is a manifold and calculates the genus.
5. If you need to recompile and run, first execute the `make clean` command to clean the file directories, and then proceed with the first step again.

zh:

1. 首先使用 `make` 命令执行 Makefile，编译这三个 C++ 文件。
2. 执行 `./face2faceindex` 命令运行 `face2faceindex` 可执行文件，该文件将 .tri 文件转换为 .face 文件。
3. 执行 `./faceindex2directededge` 命令运行 `faceindex2directededge` 可执行文件，该文件将 .face 文件转换为 .tri 文件。
4. 执行 `./MeshManifoldCheck` 命令运行 `MeshManifoldCheck` 可执行文件，检查模型是否为流形并计算属数。
5. 如果需要重新编译并运行，首先执行 `make clean` 命令清理文件目录，然后再次从第一步开始。

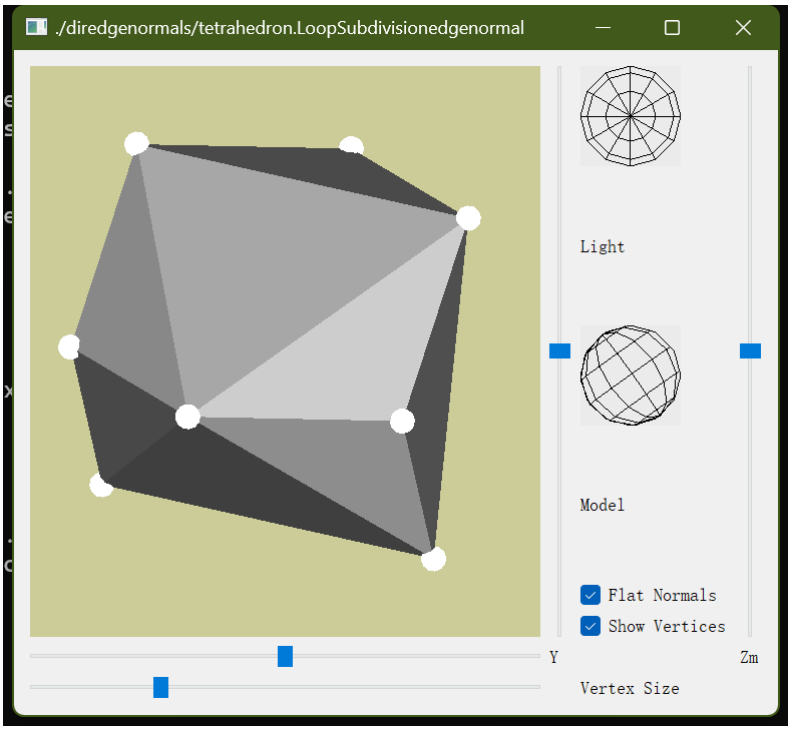
A2. Subdivision_Surfaces 曲面细分工具

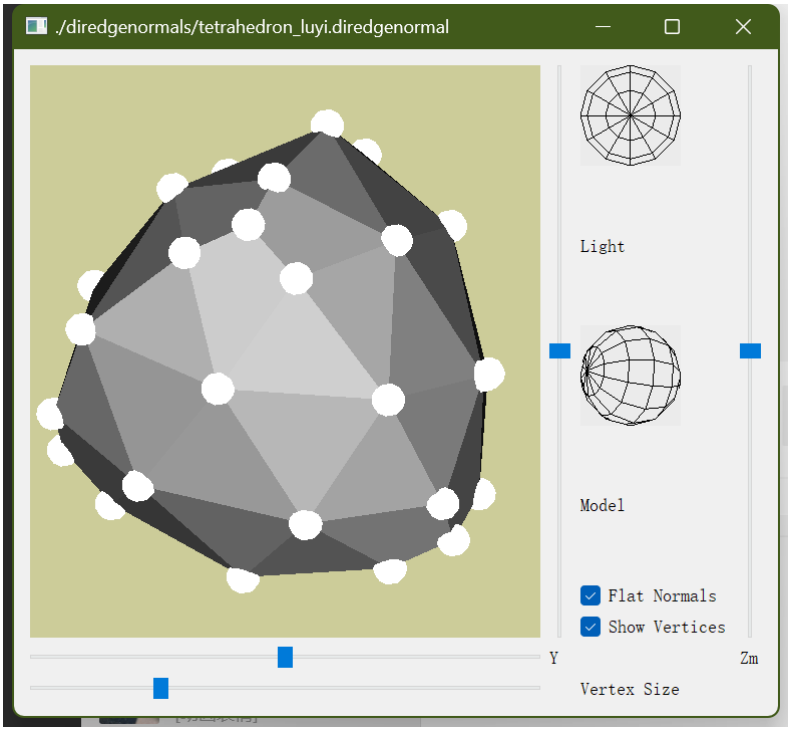
Functionality 功能

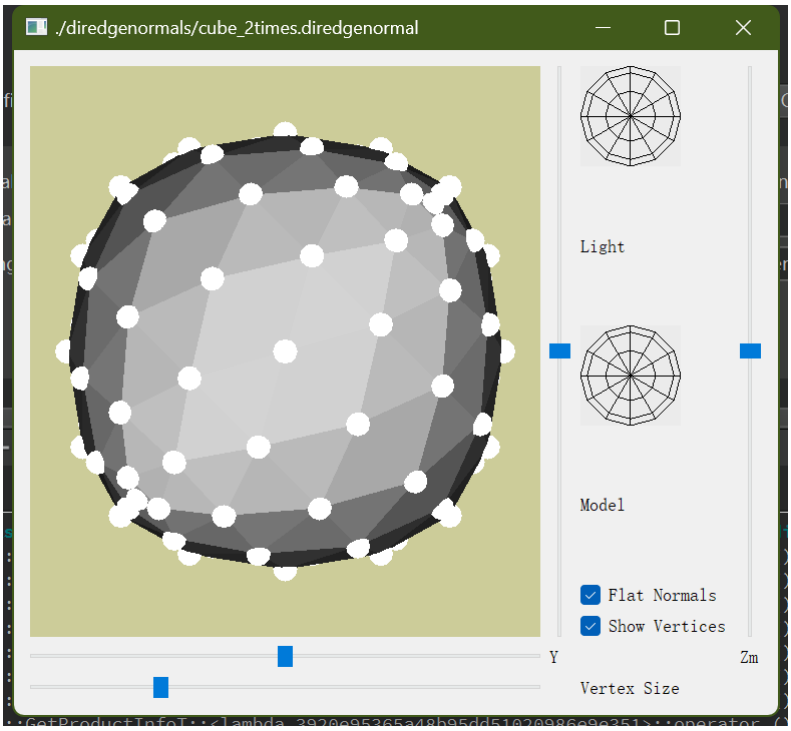
曲面loop细分:

1. 连接每条边的中点，得到新的顶点
2. 将所有顶点分为两类: 新顶点和旧顶点
3. 新顶点的坐标更新为: (A+B)*3/8 + (C+D)*1/8
    - 新顶点所在边的两个旧顶点的坐标: A, B
    - 新顶点相邻两个三角形的不共边的两个旧顶点的坐标: C, D
4. 旧顶点的坐标更新为: (1-n*u) * 旧顶点坐标 + u * 相邻旧顶点坐标之和
    - n = 旧顶点相邻的新顶点数
    - u = 3/16 if n = 3, u = 3/8n otherwise

Screenshots 截图


./diredgenormals/tetrahedron.LoopSubdivisionedgenormal
Light
Model
Flat Normals
Show Vertices
Y
Zm
Vertex Size


./diredgenormals/tetrahedron_luyi.diredgenormal
Light
Model
Flat Normals
Show Vertices
Y
Zm
Vertex Size



Notes 注意事项

en:

1. Please place the files that need to be subdivided in the same level directory as this code.
2. This code allows you to specify the number of subdivisions. During multiple subdivisions, there is no need to output files; only the final result will be output.

zh:

1. 请将需要细分的文件放在与此代码同一级目录中。
2. 此代码允许指定细分次数。在多次细分过程中，无需输出文件，最终结果才会输出。

How to Execute My Code 使用方法

en:

1. Compile the program by executing the following command in the terminal: `make`
2. Run the compiled program by executing the following command: `./main`
3. The program will prompt you to enter the filename (excluding the file extension) of the `.bridgenormal` file that needs subdivision.
4. The program will prompt you to enter the number of surface subdivisions you desire.
5. The program will generate a new `.diredgenormal` file containing the subdivided surface geometry.

zh:

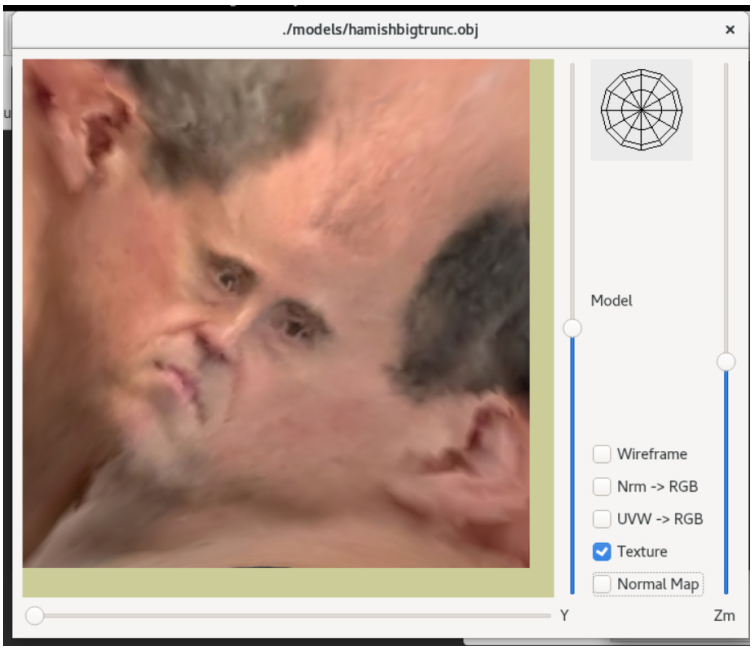
1. 在终端中执行以下命令编译程序：`make`
2. 通过执行以下命令运行已编译的程序：`./main`
3. 程序将提示您输入需要细分的 `.bridgenormal` 文件的文件名（不包括文件扩展名）。
4. 程序将提示您输入所需的表面细分次数。
5. 程序将生成一个新的 `.diredgenormal` 文件，其中包含细分后的表面几何体。

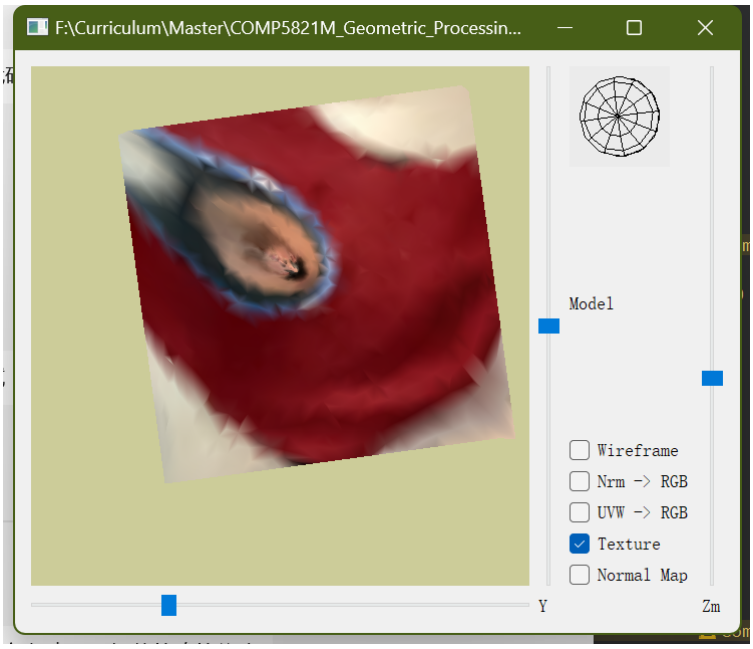
A3. Texture_Mapping 纹理映射工具

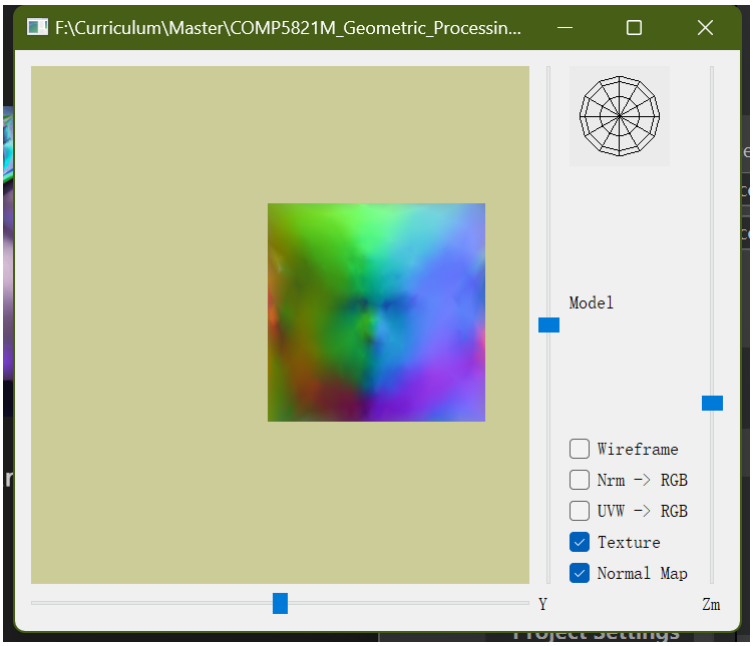
Functionality 功能

1. 建立半边数据结构和邻接点信息
2. 识别边界: 找到没有配对的半边，从该边开始，沿着边界边的两个顶点遍历，直到回到起始边界
3. 创建正方形纹理: 根据边界点生成纹理坐标，处理非边界点的默认坐标
4. 使用Floater算法为网格中的每个顶点分配纹理坐标: 迭代更新每个非边界点的纹理坐标为其邻接点的平均值
5. 计算法线: 根据三角形的法线计算每个顶点的法线，并进行归一化处理

Screenshots 截图







en:

If compilation is needed, please use the "make" command for compilation. After compilation, enter "./TextureProcessing" followed by the path to the model's obj file and press Enter to run the program. For example: "./TextureProcessing ./models/hamishtrunc1.obj".

1. When Texture is selected, the model in the window will be tiled into a square texture map; when Texture is not selected, all points will be displayed in the window in the state of model coordinates.
2. Besides Texture, when only Wireframe is selected, the model or texture map in the window will be presented in the form of a grid.
3. Besides Texture, when only UVM -> RGB is selected, the display in the window will show the color obtained by converting the coordinates of the point on the texture map.
4. Besides Texture, when only Normal Map -> RGB is selected, the display in the window will show the color obtained by converting the normals of each point in the model state (Color = (Normal + 1) * 0.5).

zh:

如果需要编译，请使用 "make" 命令进行编译。编译完成后，输入 `./TextureProcessing`，后接模型的 obj 文件路径，然后按回车键运行程序。例如：`./TextureProcessing ./models/hamishtrunc1.obj`。

1. 当选择 Texture 时，窗口中的模型将被平铺到方形纹理贴图上；未选择 Texture 时，所有点将在模型坐标状态下显示在窗口中。
2. 除了 Texture 之外，当只选择 Wireframe 时，窗口中的模型或纹理贴图将以网格形式呈现。
3. 除了 Texture 之外，当只选择 UVM -> RGB 时，窗口中的显示将是通过转换纹理贴图上点的坐标得到的颜色。
4. 除了 Texture 之外，当只选择 Normal Map -> RGB 时，窗口中的显示将是通过转换模型状态下每个点的法线得到的颜色（颜色 = (法线 + 1) * 0.5）。