CONTENT

The following describes the content found in this design document

Folders: describes the layout of our repository (folders and files)
Files: details the content of each file, including inputs/outputs and a brief description.
Compiling Cpp Code: details how to compile the cpp code to make MeshLabCppCore.js
User-Interface Manual: describes how to use the interface

FOLDERS

The repository contains one html file to open the web-browser environment and three folders: js, MeshCpp and Meshes. Folder js contains .js files. Folder MeshCpp contains .cpp files. Folder Meshes contains three .stl files. Folder Vcglib contains an open source library.

- Threejs_Playground
    - playground.html
    - js
        - meshModification.js
        - three.js
        - ThreeBSP.js
        - TrackBallControls.js
        - MeshLabCppCore.js
    - MeshCpp
        - CppMesh.cpp
        - Decorator.cpp
        - Meshing.cpp
        - Open.cpp
        - miniz.c
    - Meshes (bunny5k_asc.stl, bunny 5k_bin.stl, Leg.stl)
- Vcglib (apps, docs, eigenlib, img, vcg, wrap)

FILES

Note: if a function does not contain either inputs, outputs, or both, only a description is provided. Additionally, open source functions are only briefly described.

Playground.html
Dependencies: three.js, trackballControls.js, Three.BSP, OBJLoader.js, MeshLabCppCore, meshlab.js

- init()
    - Description: initializes the html document, sets up the scene (including the camera, light and background colors), and renders the axes and mouse
- buildAxes()

    - Inputs: length - the length of the axes
    - Outputs: axes - the axes created using several axis
    - Description: takes in all axes and adds them to scene in appropriate location. Adapted from http://soledadpenades.com/articles/three-js-tutorials/drawing-the-coordinate-axes
- buildAxis()
    - Inputs: src- beginning location (Vector3), dst - end location (Vector3) , colorHex - color of axis, dashed- true if dashed, false if solid
    - Output: axis - the axis created
    - Description: builds a single axis
- onWindowResize()
    - Description: updates rendering frame size when window size changes
- onDocumentMouseMove()
    - Inputs: event- contains information regarding mouse movement (especially coordinates)
    - Description: Keeps track of the mouse's position and whether the mouse is positioned over an object in the scene
- isObjectInArray()
    - Inputs: object - object of interest, array - array to check
    - Outputs: index1 - index of location of object in array (or -1 if not found)
    - Description: searches for an object in an array. If the object is found in the array, its index is returned. If the object is not found, -1 is returned
- checkSelected()
    - Inputs: intersects - array of intersected objects
    - Description: checks if the object in the scene was selected by the mouse
- onDocumentMouseDown()
    - Input: event- event- contains information regarding mouse movement (especially coordinates)
    - Description: Allows one to move an object or rotate the scene upon mouse move
- onDocumentMouseUp()
    - Input: event- event- contains information regarding mouse movement (especially coordinates)
    - Description: completes selection of object if mouse is above an object.
- animate()
    - Description: animates the scene
- render()
    - Description: displays the scene in the web-browser
- removeFromArray()
    - Input: orig- array where object is found, remove- objected to be removed
    - Description: removes an object from the array.
- search()
    - Input: nameKey - object key name , myArray - array to search in
    - Output: the object

    - Description: searches an array for a specific object.

meshModification.js

- HandleFileSelect()
    - Inputs: evt- change event, meshFromFile- mesh, meshLab- mesh module, callback- function
    - Description: opens, parses and reads the mesh file. It then creates the mesh by calling on createMesh function. The function calls back two variables in the function → the mesh file and the module that stores the mesh.
- createMesh()
    - Inputs: mesh, meshLab- mesh module
    - Outputs: mesh- constructed mesh
    - Description: creates the mesh using a mesh and and mesh module. The function creates vertices and face geometry (along with their normals) using a pointer to the mesh file. If the mesh already exists, only the geometry is updated.
- QuadricSimp()
    - Inputs: mesh, meshLab- mesh module
    - Description: performs quadric simplification on a mesh
- ClusteringSimp()
    - Inputs: mesh, meshLab- mesh module
    - Description: performs clustering simplification on a mesh
- removeDupVert()
    - Inputs: mesh, meshLab- mesh module
    - Description: removes duplicate vertices on mesh
- laplSmooth()
    - Inputs: mesh, meshLab- mesh module
    - Description: performs mesh smoothing
- buildFaceNormals()
    - Inputs: meshLab- mesh module
    - Outputs: array of face normals
    - Description: creates an array of all the face normals
- buildVertexNormals()
    - Inputs: meshLab- mesh module
    - Outputs: array of vertex normals
    - Description: creates an array of all the face normals

Three.js

Open source library that makes WebGL (3D in the browser) easy to use. Website: http://threejs.org. Author: mrdoob, http://mrdoob.com/

ThreeBSP.js

Open source binary space partitioning tree that performs constructive solid geometry applications in three.js

TrackballControls.js
Allows the user to rotate the camera about an object and drop/drag objects. Authors: Eberhard Graether http://egraether.com/, Mark Lundin http://mark-lundin.com, Simone Manini http://daron1337.github.io, Luca Antiga http://lantiga.github.io

MeshLabCppCore.js
Mesh modification code written in cpp compiled into asm.js (an extraordinarily optimizable, low-level subset of JavaScript) using emscripten. The following code was compiled: cppMesh.cpp, decorator.cpp, meshing.cpp, and open.cpp

CppMesh.cpp
Contains functions that allow access to mesh information, ie. mesh-pointer, number of vertices, number of faces, vertex vectors, face vectors, etc. Also allows functions to open and save mesh.
https://github.com/cnr-isti-vclab/meshlabjs/tree/master/sources

Decorator.cpp
Contains functions to build face normals and vector normals.
https://github.com/cnr-isti-vclab/meshlabjs/tree/master/sources

Meshing.cpp
Contains quadratic simplification, clustering, and removing duplicate vectors function.
https://github.com/cnr-isti-vclab/meshlabjs/tree/master/sources

Transform.cpp
Contains the laplacian smoothing function.
https://github.com/cnr-isti-vclab/meshlabjs/tree/master/sources

Open.cpp
Contains opening mesh function.
https://github.com/cnr-isti-vclab/meshlabjs/tree/master/sources

Miniz.c
Needed for compiling using emscripten.
https://github.com/cnr-isti-vclab/meshlabjs/tree/master/sources

VcgLib
Contains the Visualization and Computer Graphics Library, an open source portable C++ templated library for manipulation, processing and displaying with OpenGL of triangle and tetrahedral meshes (http://vcg.isti.cnr.it/vcglib/).

COMPILING CPP CODE

To compile the cpp code found in the MeshCpp folder and make the MeshLabCppCore.js file, emscripten must be installed and dowloaded by following this link
http://kripken.github.io/emscripten-site/

When compiling, place all cpp and h files in the emscripten folder. Compile using:
./emcc --bind -I../../../vcglib CppMesh.cpp Meshing.cpp open.cpp Decorator.cpp Transform.cpp -o MeshLabCppCore.js

USER-INTERFACE MANUAL

Buttons:

Choose File - allows one to choose and upload a .stl file for loading onto the screen.
Add Sphere - adds a sphere of radius 50, 30 horizontal segments and 30 vertical segments
Add Cube - adds a cube of dimension 50x 50x 50
Add Plane - adds a plane of dimension 0x500x500
Union - unions the mesh of one object with that of another object
Intersect - intersects the mesh of one object with that of another object
Subtract - subtracts the second mesh from the first mesh
Remove Duplicates - removes duplicate vertices from a mesh
Quadric Simplification - applies quadric simplification algorithms to a mesh
Smoothing - applies a Laplacian smoothing algorithm to a mesh
Scale Object - scales an object by 1.5 in the x, y, and z direction
Remove Object(s) - removes meshes that are currently in the scene
RotateX - rotates the object by a given amount along the x-axis
RotateY - rotates the object by a given amount along the y-axis
RotateZ - rotates the object by a given amount along the z-axis

Directions to prepare a mesh for CSG:

1) Click the “Choose File” button to upload an .stl mesh.
2) Select the mesh (it should turn red) and then click the “Remove Duplicates” button
3) Select the mesh again and click the “Smoothing” button (optional)
4) Select the mesh again and then click the “Quadric Simplification” button. Do this as many times as needed until you reach the desired number of faces or vertices
5) To close the mesh, create a plane by clicking on the “Add Plane” button. Move the plane so that it touches the mesh and also covers the hole on the mesh. Select the plane and then the mesh and click “Intersect”. Do this for as many holes as needed.
6) If you wish to scale the mesh, select the mesh and then click the “Scale Object” button.
7) Your mesh is now ready for CSG.

Directions to perform CSG:

Union: select both meshes and then click the “Union” button
Intersect: select both meshes and then click the “Intersect” button

Subtraction: select a mesh and then the mesh you would like to subtract from the first mesh and then click the “Subtract” button