Browser-based Constructive Solid Geometry for Anatomical Models

Final Report

Vikram Chandrashekhar, Nicole Ortega

Mentors: Alex Mathews and Param Shah

May 6, 2016

(EN.600.446) Computer-Integrated Surgery II

Dr. Russell Taylor

Goal

The goal of our project is to develop a *browser-based* constructive solid geometry application for the efficient creation of 3D anatomical models, specifically the creation of a modular orthosis.

Motivation and Background

One in 323 children in the US are born with cerebral palsy. Two out of every three children born with cerebral palsy could walk if they had proper orthotic devices to alleviate their condition. Currently, ankle foot orthoses are used to correct gait and prevent muscle deformities. The current process for making custom ankle foot orthoses is tedious and wasteful. This process begins by creating a mold of the foot of interest. After this mold is made, it is filled to create a cast of the foot. Once the cast hardens, the mold is removed, leaving just the cast of the foot. The orthosis is created around this cast and cut away when completed. Ultimately, the cast, mold, and scraps of material from the orthosis are thrown away. The cost to make a custom orthosis ranges from $400 - $600 (compared to $10 - $80 for a non-custom, off-the-shelf orthoses) and can take up to 3 weeks. Checkups every 6 months are required for adjustments. On average, these orthoses are replaced every six to eight months, which involves repeating the tedious and wasteful casting process.

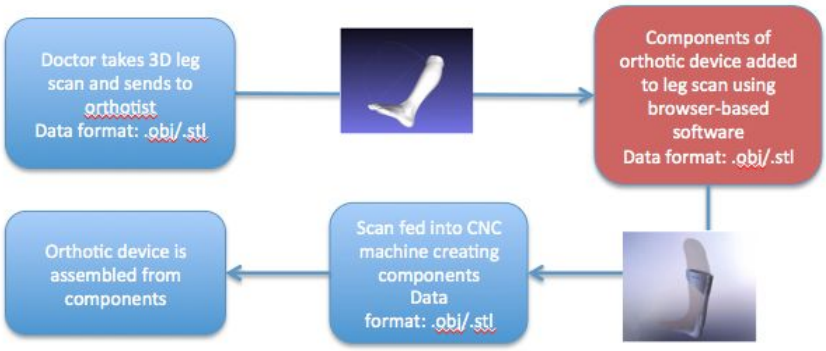

Figure 1. Workflow for clinical use. The red box is our contribution to the workflow.

Fusiform Medical Devices, a medical device company, has developed a process to reduce waste and time of designing and casting a custom orthosis as shown in **Figure 1**. The Fusiform process begins by using a structure sensor mounted on an iPad to take a 3D anatomical scan of the leg to a 1mm accuracy. The orthosis is then created in Solidworks using the scan and is fabricated by a CNC milling machine. This process reduces waste since it no longer requires the creation of a mold of the leg while also taking advantage of the CNC machine's subtractive production. While the end result of this process is still a custom cast, the individual pieces that comprise the cast can easily be replaced due to the modularity of the design of the components.

However, the Fusiform process has room for improvement. On average, it takes approximately 10 hours to design the orthosis on SolidWorks. The goal of our project is to develop a browser-based constructive solid geometry application for the efficient creation of 3D anatomical models, specifically the creation of a modular orthosis. This approach is unique in that no other browser-based interface like this exists nor has it been applied to anatomical models.

Technical Approach

Our browser-based software was developed using Javascript and HTML. In order to perform the 3D rendering and modeling we used an open-source Javascript library called three.js. Using three.js as our framework, we integrated multiple additional components to create an all-in-one constructive solid geometry software. The two key functionalities added were constructive solid geometry and mesh simplification/modification.

Constructive Solid Geometry

Constructive solid geometry (CSG) is the use of boolean operators to combine polyhedra in a d-dimensional space. For this project, we restrict the discussion to the three-dimensional case as it is the most relevant. The three boolean operations that were considered were: union, intersection, and subtraction; a union operation is the combination of two polyhedra that represents all the points contained in either the first or second polyhedron. The intersection of two polyhedra represents all of the points in both the first and second polyhedron. Lastly, the subtraction of two polyhedra represents all of the points in the first polyhedron but not in the second one. These three operations can be visualized in **Figure 2**. This functionality is extremely useful as complex polyhedra can then be represented by boolean combinations much simpler objects. For our project in particular, CSG will allow the orthotist to combine pre-designed orthoses components in various ways with a scan of the leg to customize the orthosis. In order to implement this functionality in our browser-based software we chose to use an open-source three.js-compatible CSG library called three-csg.

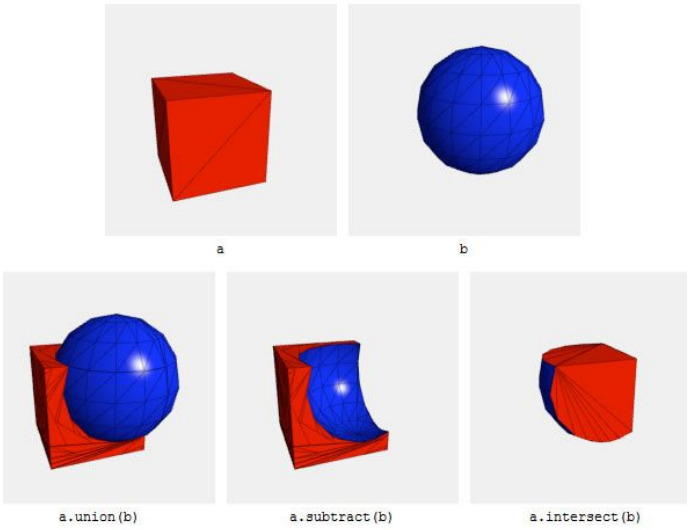

Figure 2: Visual representation of CSG operations
Source: http://codevisually.com/wp-content/uploads/2011/12/cvdec1_06.jpg

Mesh Simplification

Mesh simplification is the act of reducing the points on a mesh while attempting to preserve its topography and features. Several methods of mesh simplification exist including vertex/edge/face decimation, energy function optimization and vector clustering. For this project, we focused on quadric simplification. Quadratic Simplification computes the Q matrix for all the initial vertices on the mesh. The Q matrix is the quadric error, sum of fundamental error quadrics. Quadrics are created by the planes that meet at the vertex. The quadric error is calculated using the formula $\Delta v = v^T \mathbf{Q} v$. Valid vertex pairs are chosen if the pair is an edge or the Euclidian distance is less than a threshold. The optimal contraction of a valid pair is determined by the equation $v'^T(Q_1 + Q_2)v'$ and is designated as the cost of the contraction. Vertex pairs with minimal cost are contracted and the whole process is repeated. For our project, mesh simplification will allow the manipulation of a mesh in a browser environment. In order to implement this functionality in our browser-based software we chose to use an open-source c++ mesh simplification code converted to javascript using Emscripten.

Software Workflow

After developing these individual CSG and mesh simplification components, they were combined into a single package. The workflow for our software is shown in **Figure 3** and the

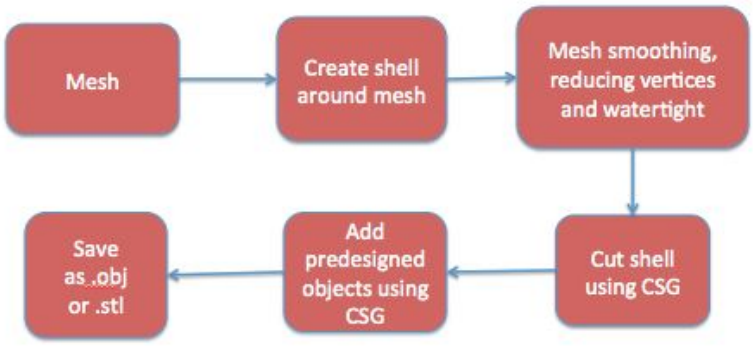

Figure 3. Workflow for our software.

functions implemented in our software are shown in **Figure 4**. This workflow is expanded from the red box in **Figure 1**.

The orthotist receives an annotated leg scan from the doctor and begins by creating a shell around the mesh. After performing mesh simplification and smoothing operations, the shell and cut to create a cast of the leg. Predesigned components of the orthosis are then intersected with the cast using various CSG operations to create custom fit components. The completed orthosis is then saved and sent to the CNC machine to be fabricated.

Input	Function	Output
2 meshes	union	Single combined mesh
2 meshes	subtract	Single mesh with the volume of the second mesh eliminated
2 meshes	intersect	Single mesh made up of the overlap between the two meshes
1 mesh	quadric simplication	Reduces number of vertices and faces in the mesh while preserving topology
1 mesh	remove duplicates	Remove duplicate vertices in mesh
1 mesh	exportOBJ	Downloads .obj file representing the selected mesh
1 mesh	scale	Scales mesh in x, y, and z directions by a given amount
1 (or more) meshes	remove	Removes the selected meshes from the scene
1 mesh	rotateX, rotateY, rotateZ	Rotates the selected mesh around an axis (x, y, or z) by a given amount

Figure 4. Software architecture with general overview of functions

Management Summary

Proposed Deliverables

Minimum

- three.js ”playground” in browser using constructive solid geometry algorithms for simple objects (sphere, cube, prism, etc)
    - sphere/cube addition algorithm
    - CSG union algorithm
    - CSG intersect algorithm
    - CSG subtract algorithm
- Mesh modification module for simple objects

Expected

- Mesh modification module for anatomical scans
    - Mesh cutting algorithm
    - Mesh smoothing algorithm

    - Mesh simplification algorithm
    - Watertight mesh algorithm
    - Mesh scaling

Maximum

- Test cast fabrication using a 3D printer and test “fits” on patients

Actual Deliverables

Minimum

- three.js ”playground” in browser using constructive solid geometry algorithms for simple objects (sphere, cube, prism, etc)
    - sphere/cube addition algorithm
    - CSG union algorithm
    - CSG intersect algorithm
    - CSG subtract algorithm
- Mesh modification module for simple objects
- Mesh importing (for .stl/.obj files)

Expected

- Mesh modification module for anatomical scans
    - Mesh cutting algorithm
    - Mesh smoothing algorithm
    - Mesh simplification algorithm
    - Mesh scaling

Maximum

- Playground with improved usability - sliders (to change parameters)
- Rotation of objects independently of axis
- Mesh Exporting (.obj files)

Removed Deliverables

- Mesh modification module for simple objects - Removed
- Test cast fabrication using a 3D printer and test “fits” on patients - Removed

Unmet Deliverables

- Watertight mesh algorithm

Division of Labor

The development of the software was split equally over the course of the semester. The CSG integration was completed by Vikram Chandrashekhar. Mesh modification algorithms were implemented by Nicole Ortega.

Dependencies

Dependency	Resolution
Three.js - Javascript software package that interfaces with WebGL to perform 3D rendering	Available online for free
The Visualization and Computer Graphics Library for mesh modification algorithms (in C/C++)	Available online for free
Emscripten to port C/C++ code to Javascript	Available online for free
ThreeBSP.js to perform CSG	Available online for free
Object (.obj/.stl) files of anatomical leg scans	Provided by mentors on 3/15
MeshLabJS - Javascript application ported from C++ used to obtain simplification and vertex removal algorithms	Available online for free

All dependencies have been resolved prior to starting development.

Milestones

Milestone	Expected Date	Completion Date
Create a browser-based three.js playground (add, move, and drag simple objects)	March 1	February 25
Implement Constructive Solid Geometry (CSG) Algorithms for simple objects (sphere, cube)	March 20	March 10
Expand the CSG algorithms to work for anatomical	March 25	March 20

objects		
Implement a mesh decimation/reduction algorithm for anatomical objects	March 30	March 30
Implement a mesh cutting algorithm for anatomical objects	April 2	April 5
Implement mesh smoothing algorithm for anatomical objects	April 10	April 15
Improve usability by adding parameter customization for various functions	April 30	April 25
Add functions to rotate objects independently of the world-axis	April 30	April 30

Lessons Learned and Future Plans

Through this project we learned the necessary skill set to develop a browser based application with Javascript. A major challenge we faced was quickly trying to learn Javascript and integrating different open source packages into one application. We also learned the importance of software design documentation.

This project will be ongoing over the summer and fall of 2016. The primary goals of this continuation will be to implement a more user-friendly user interface with our software and complete any unmet deliverables. Since this project has been a proof-of-concept for the development of a browser-based CSG application, the secondary stages will involve significant refining/modification to our software. The second iteration of this software should be completed by the end of August 2016.

Acknowledgements

We would like to thank our mentor Alex Mathews and Param Shah for providing us with guidance. As well as Dr. Russell Taylor and Alexis Cheng for helping us develop project management skills.