

Constructive Solid Geometry

Adam Leslie: ahl552, Corbin Rogerson: cr34984

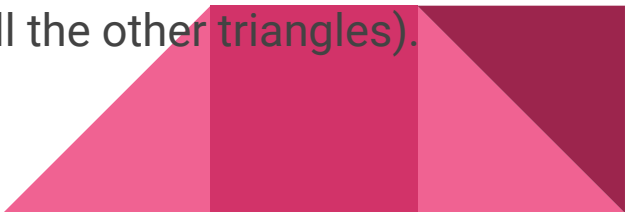
Github Repo Link

<https://github.com/adamhleslie/csg>

Background

Constructive solid geometry is the process of taking primitive shapes such as Prisms, Cones, spheres etc. and performing set operations (Union, Intersection and Difference) to create new, watertight shapes.

The algorithm we used was a BSP tree that split up each primitive into a tree, categorizing its triangles as either in front of, behind or on the plane of a specific triangle. This was handled recursively. We then, depending on the desired operation, reclassified each objects' triangles based on the other's triangles and decided which ones to keep. For example, for a Union operation we would keep all the “outside” triangles (those triangles that were classified as in front of all the other triangles).



What was accomplished

We have implemented a BSP tree and the methods necessary to classify any triangle within the BSP tree utilizing the plane each triangle generates.

The algorithm for merging trees is very similar to the algorithm for generating a BSP tree for a single primitive shape. Once merged we can easily decide which triangles from the merged tree we would like to keep or discard. This is the main algorithm for the project.

This allows us to make a myriad of shapes as we will show later in Artifacts.



What was accomplished (cont.)

We also implemented a simple way to view all the triangles and how they were split by the BSP tree by generating red lines around their edges. This was extremely useful for debugging.

We also implemented a very rudimentary language utilizing Polish Notation as it allowed us to ignore a lot of parsing difficulties like parenthesis. The language let us define position, rotation and dimensions of each primitive. We have included a few sample inputs for our language to generate different shapes.



What was accomplished (cont.)

A large part of the project was spent generating primitive shapes. The shapes we implemented were: Sphere, Cylinder, Cone, Rhombohedron, Rectangular Prism, and Triangular Prism. They are parameterized so the person using the language can define their dimensions and even their color for easier viewing.



Challenges

- Also we are not experts in C++ so we did some very poor construction of our BSP tree at first and poorly handled the copy constructor and copy assignment operators without cloning pointers. This caused a lot of crashing.
- The papers we read were ambiguous about a lot of very necessary details. For example, if using the Difference operation you need to flip the inside triangles otherwise the triangles will be facing the wrong direction and you will have a hole.
- Rotation also tripped us up as we didn't remember to update triangle normals after rotating, causing our algorithm to cut incorrectly.



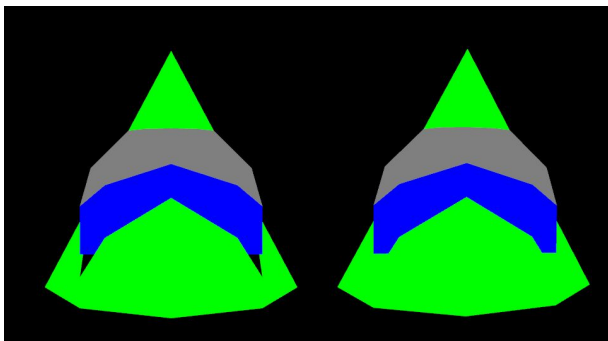
Challenges (cont.)

- Generating spheres was a bit of a pain as they are just fairly complicated shapes to generate relative to something like a cylinder.
- Parsing was also a bit of a pain as we didn't spend a lot of time making it error proof, it's fairly easy to break.
- We have a poor implementation of triangles. Each triangle separately stores its points. So even if two triangles share the same point, they will both have a separate copy. This was poor planning on our part.

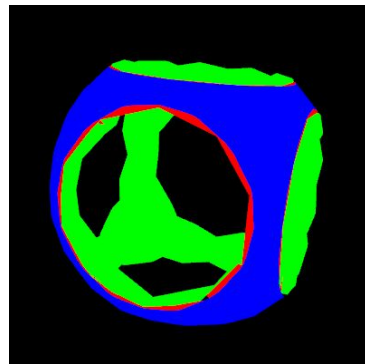


Challenges (cont.)

The biggest challenge we had was the case where triangles were coplanar with other triangles, we didn't know how to handle whether they were “in front” or “behind” other triangles. Depending on the binary operation they would need to be handled differently. This was not a problem for the Union and Intersection case but the Difference case is still a problem that we wish we had had more time to figure out.



Union before and after fix

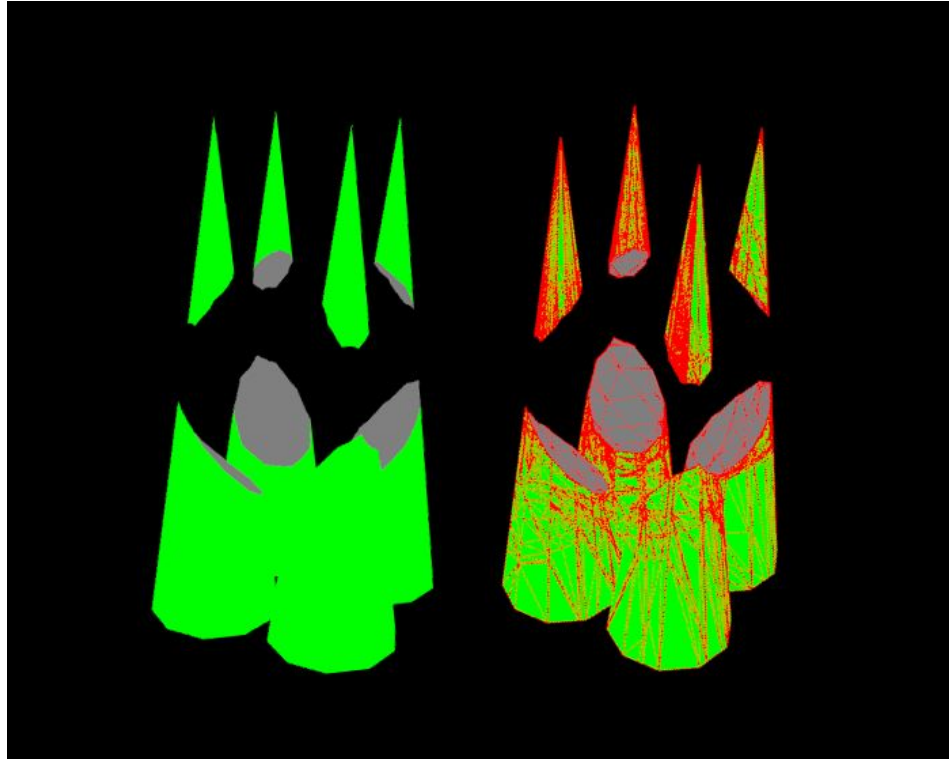


Current Difference errors

Artifacts

This is an example of 4 cones being differenced by a sphere.

The right image is also showing how all the triangles were split due to the sphere.

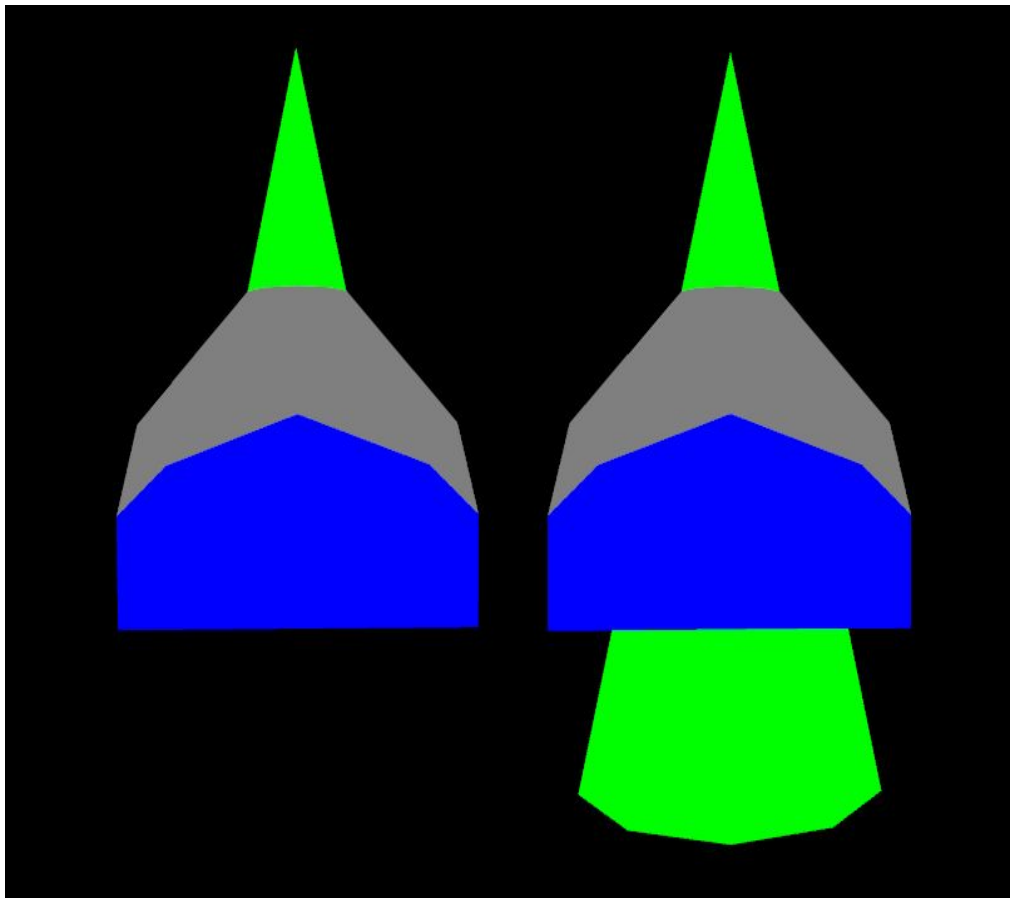


Artifact

The left image shows the prior issue we were having with the coplanar triangles.

The right image shows the fixed version where we considered coplanar triangles to be “front” triangles.

The shapes are a blue cube intersected with a grey cone all unioned with a green cone.



Resources

Constructive Solid Geometry Using BSP Tree Christian Segura, Taylor Stine, Jackie Yang

Computer Graphics: Principles and Practice Ch. 12.7 Constructive Solid Geometry, Van Dam, Feiner, Hughes

