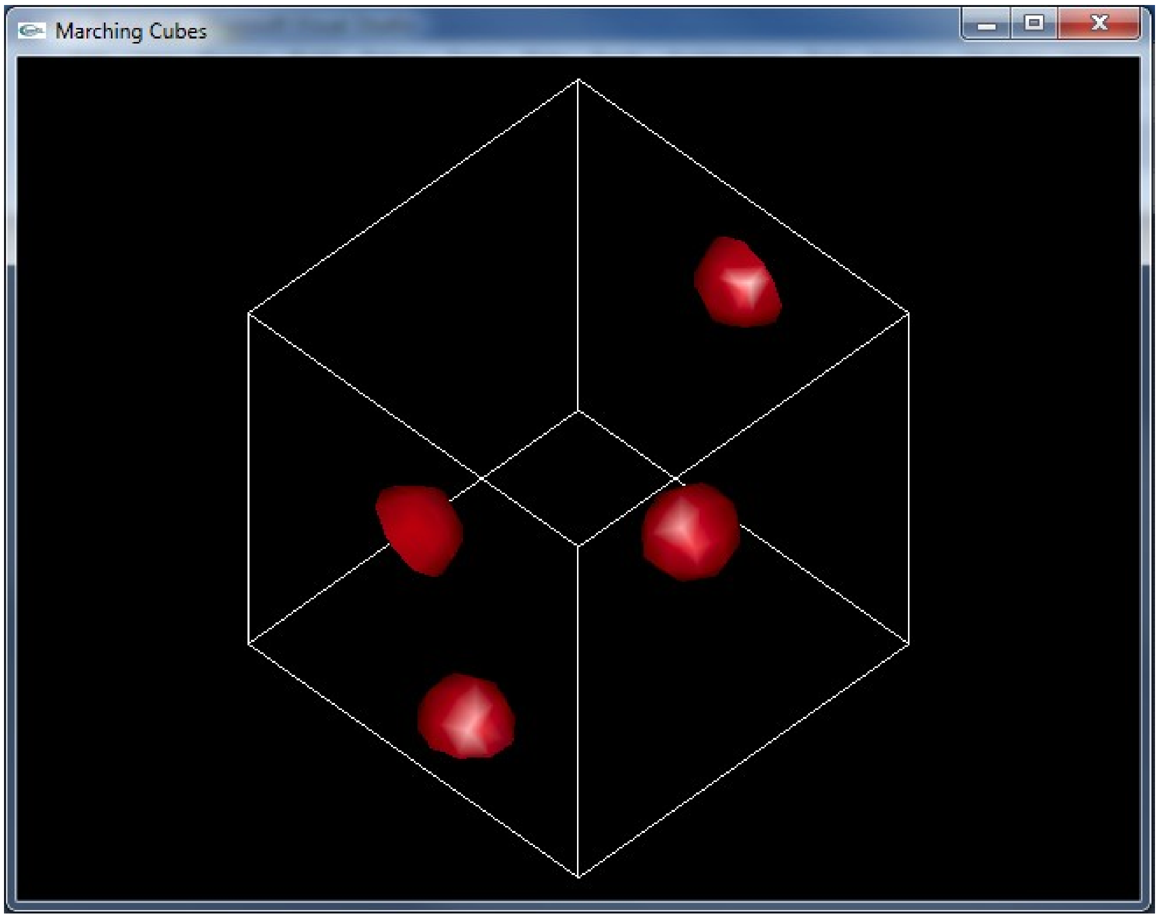
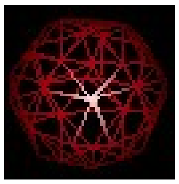
Blobby Dripping Blood



Justin Pinili & Dante Ratto
cmps161
March 16, 2011

Abstract:

The marching cubes algorithm is a computer graphics technique that takes a polygonal mesh of an isosurface from a three-dimensional scalar field, also known as voxels. This paper demonstrates the information we had to use in order to create a program that uses the marching cubes algorithm to simulate dripping blood inside a cube.



Introduction:

To start off with our final project, we first had to understand the marching cubes algorithm. After doing research on the internet, we found some very interesting articles that explained the algorithm. The marching cubes algorithm is broken down into two major components. First, the program has to be able to check which sections will be chopped up in an individual cube. In cubes, there are 256 possible configurations for each corner; where each point is either inside or outside of the cube. In addition to this, the program needs to be able to check where along each cube that the isosurface crosses, and use the intersections to create a new triangular part on the isosurface. So, the code is heavily based on look-up tables, where the table contains every possibility of where the isosurface could intersect. Once we were able to set up and figure this out, we needed the metaballs to collide with a cube surface so it looks like the "blood" was contained inside the cube.

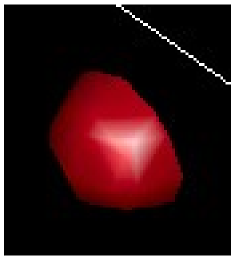
Related Works:

Some related works that we found were the exact same people that we reference for out project. Matthew Ward, part of the WPI CS department, gave a summary of "Metaballs/Blobby Objects". Paul Bourke called his work "Polygonising a Scalar Field" and even went into more detail creating marching tetrahedrons. Andreas Jonsson explained 2d metaballs, which explained a simpler version called marching squares. In addition to this, OpenGL also had example code on metaballs, where they also referenced Paul Bourke.

Technical Detail:

To create the project, we started by creating a project using Microsoft Visual Studios 2010. We set up a blood class with a header file, as well as a main class that runs the program. Our main method of the project was using the marching cubes algorithm. First, we would make a copy of the values for the cube, then check to see which vertices are on the inside or outside of the surface. Once we did this, we also had to calculate the new normals of the surface for those points. Like we mentioned before, the algorithm relies heavily on look up tables, which contain the information for where the marching cubes intersect, the new vertices for the intersected positions, the edge connections for those positions as well as the normals for those connections.

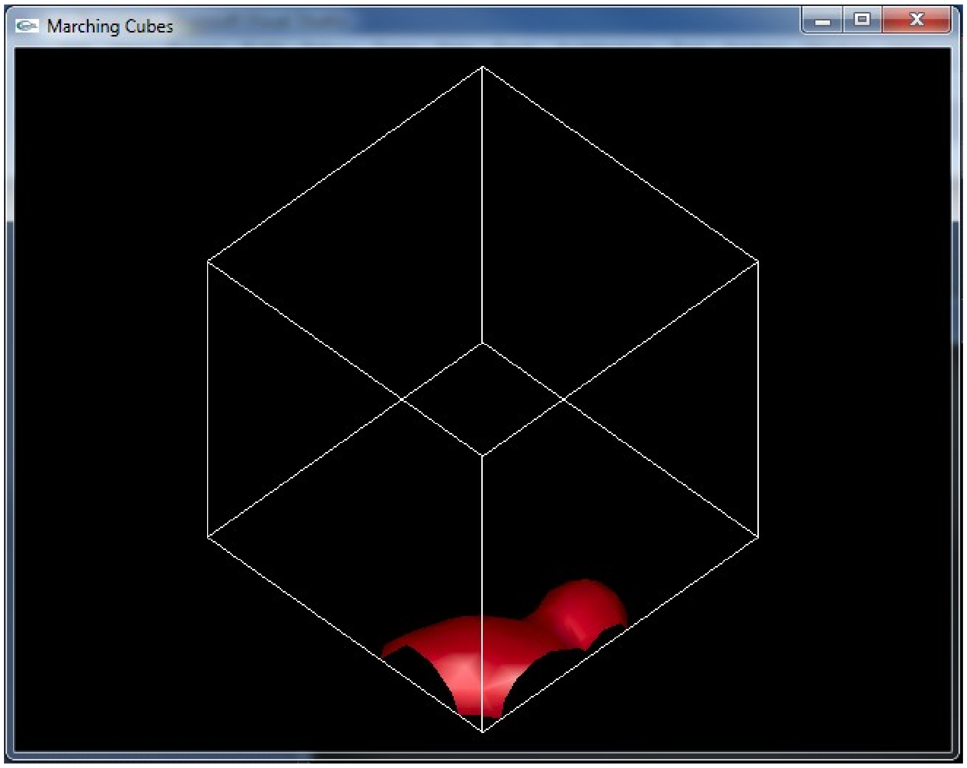
After we have calculated the normals, we then draw the triangles that were found on the cube. This gives each metaball that blobby looking formation. Once we were able to accomplish this, we needed to have the metaballs collide with a surface. Paul Bourke had a great example of how metaballs would collide with a cube. Again, this relies heavily on lookup tables because it contains the 8 vertices on the cube, and the two states that a surface collides with, either inside or outside the cube.



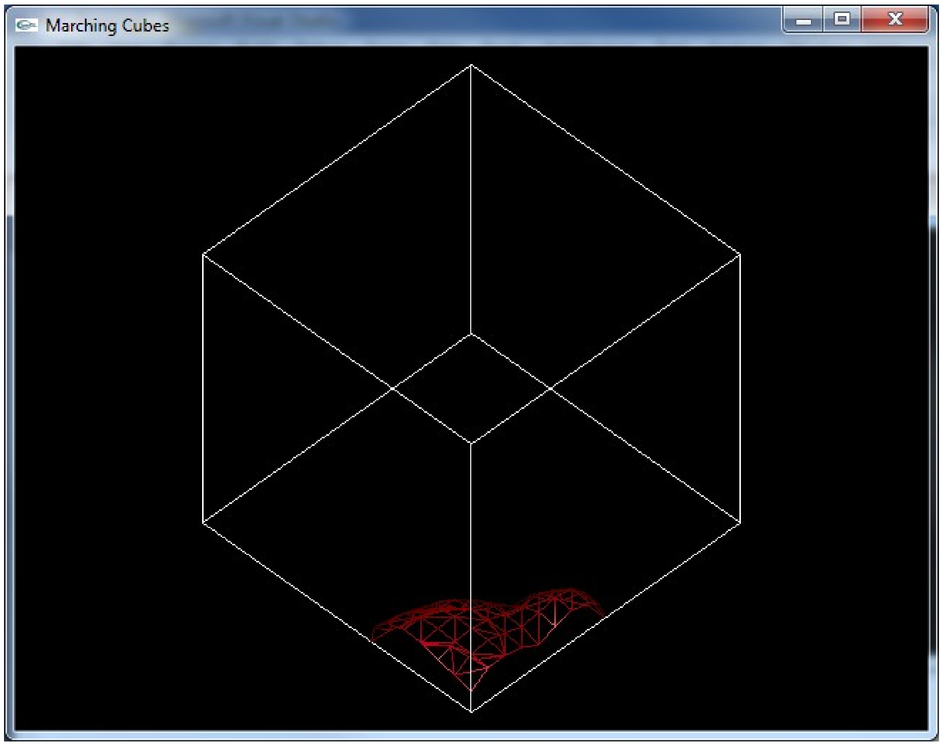
Results:

The results of our project were quite satisfying. Although our original objective was to have our metaballs affected by physics, the metaballs still move by a path of an upside down parabola. We ran out of time because we did not take into consideration how hard it was to get the marching cubes algorithm work, as well as getting the metaballs to collide correctly to the surrounding cube. We were able to create two variations of movement for the metaballs. In the future, if we decide to work on the project again, we would like to implement physics into the project. That would give the project a better feel of having the blood explode from a source and drip according to physics instead of being hard coded. The following is a screenshot of the working blobby dripping blood code:

This is the marching cubes algorithm in action.



Here is a wireframe image of the exact pool of "blood".



Conclusion:

This final project was very interesting to create over the course of this winter quarter. We originally learned about the marching squares algorithm during our cmps161 lecture. Pang also explained marching cubes briefly during class. It was then that we decided we wanted to work on this algorithm for our final project. The hardest part was trying to use the look-up tables to correct way so that we were able to reference the correct information at the right time. There was a lot of trial and error involved, but once we got the algorithm to work, it was very satisfying.

References:

Matthew Ward
Paul Bourke
Andreas Jönsson
Andreas Jönsson's code example
OpenGL metaballs