Real Time Mesh Combustion

Kevin Frans, Jeremy Ma

Motivation For this project, we aimed to simulate a burning animation for complex meshes. The main idea is to render fire that follows along the curvature and shape of a given mesh, producing a dynamic fire effect while the mesh slowly burns away. This effect would be useful for simulating fire in movies and games. For the games domain especially, it is important that the fire simulation be efficient enough to run in real-time. In addition, since fire affects the 3D shape of the mesh in real-time, the mesh combustion can provide the basis for real gameplay elements in 3D worlds or games.

Background A number of approaches exist to modeling fire, which generally fall into the categories of physically-based modeling and particle-based modeling. While we did not implement any of these methods fully, we took inspiration from these papers and chose parts that seemed fit for our use case. Specifically, we opted to use a particle-based approach to simulating fire, and we made use of neighboring particles to calculate the color of the flame.

Goals The goal of this project was to simulate a burning mesh in real-time. We aimed to optimize performance for real-time simulation, so that the framework can be used efficiently in games. We wish for the mesh to deform in a natural “burning-away” behavior, gradually breaking down from a starting point. The fire should be simulated in a realistic-looking manner, ideally with some approximation to the Navier Stokes equation, and flame color should emulate the color gradient due to temperature, e.g. with yellow inside zones and red outside zones.

Mesh Deformation In this section we present our algorithm for deforming a mesh to simulate the combustion of an object. Our method relies on procedurally removing triangles from the mesh in order, spawning fire particles when each triangle is removed. Once a triangle is removed, its neighbors should then be removed, then its neighbors' neighbors, and so on. A key difficulty is that in OBJ files, the triangles are not necessarily stored in any reasonable order, so we must figure out this order during simulation time. The resulting algorithm (see below) involves maintaining a queue of "burnable" triangles, starting from a specified start triangle. When a triangle is burned, any triangles which share a vertex with that triangle are then added to the burn queue. This approach is similar to a breadth-first search over all triangles.

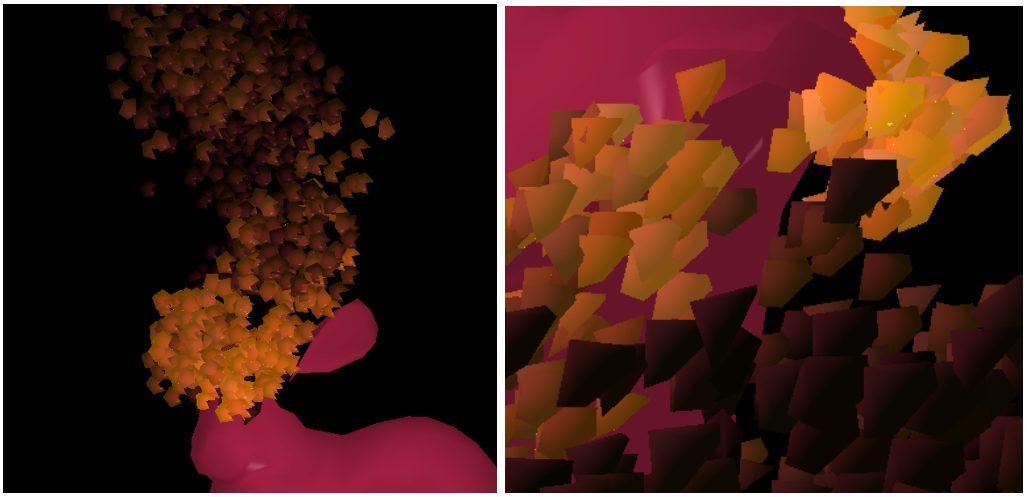
```
burn_queue = queue()

burn_queue.push(triangle_start)
while burn_queue is not empty:
      burn_triangle = burn_queue.pop()
      for other_triangle in all_triangles:
            if other_triangle shares a vertex with burn_triangle:
                  burn_queue.push(other_triangle)
      Burn(burn_triangle)
```

Particle System In this section we describe how we implemented realistic flame particles. A key contributing factor of the color of the flame is determined by the distance from the "innermost zone". Due to the smaller amount of oxygen available near the innermost zone, the flame burns with a more yellowish color, slowly becoming red as we move away. In order to do this, the

innermost zone is kept track through a running average of the x and z coordinate of the particles. Then using the distance(`dist`) from the innermost zone, the color of the next spawned particle and it's decay rate can be determined using the code below. Given the decay rate, alpha blending was used to emulate the effect of particles fading away. Finally, an RK-4 integrator is used to approximate a differential equation involving the wind force and a viscous drag force in order to simulate the motion of the particles.

```
mat.r = 1.0f
mat.g = 0.5f + 0.3f * rngG * dist/4.0f
mat.b = 0.5f * rngB * dist/4.0f
decay_rate = 1-(0.01/ (dist / 4.0f)
```



Results We can see that the flame spawns in bursts as each vertex of the mesh is combusted. Regarding motion we can see that the particles follow a general upward trend and the trajectory of the motion is quite realistic. Regarding color, we can see that the center of the flame is much brighter and more yellow and slowly fades to red as the particle moves away from the innermost zone. Although we used alpha blending to simulate the fading of particles, which would introduce artifacts such as 'faded' particles occluding other particles, overall the flame still looks reasonable.

A video of the results is linked below.

https://drive.google.com/file/d/1-1ABsSijYFJ-OaCuFDeh1DKF3NMtuZKU/view?usp=sharing

Conclusion In this project we created a method for real-time mesh combustion. Our method involved a mesh deformation algorithm which gradually removes triangles from a given OBJ file, simulating the spread of heat as an object burns. This mesh deformation follows the curvature of the mesh, giving a realistic-looking effect to fire spread. Our method also simulates fire as a particle system. Particles are released at the vertices of a triangle when it is burned, and particles then enter a system where they are affected by wind, viscous drag, and a repulsion force from other particles. Particle motion is calculated with the RK4 integrator.

Future work can involve an algorithm that can burn multiple meshes together, with fire spreading between them, or a fire spreading algorithm that can involve multiple heat sources. In addition, the particle system simulation for fire can be reconfigured to support an implicit ray-casting surface, or the particle system could involve more complicated fluid dynamics based on density and temperature of certain particles.

References

Nguyen, Duc Quang, Ronald Fedkiw, and Henrik Wann Jensen. "Physically based modeling and animation of fire." In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 721-728. 2002.

Lyes, T. S., and K. A. Hawick. "Fire and Flame Simulation using Particle Systems and Graphical Processing Units." In Proceedings of the International Conference on Modeling, Simulation and Visualization Methods (MSV), p. 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2013.

Vanzine, Yuri, Dana Vrajitoru, Zhong Guan, and Michael R. Scheessele. "Realistic Real-Time Rendering of Fire in a Production System: Feasibility Study." (2007).

“Lecture 17: Graphical Simulation of Fire”. Imperial College of London Department of Computing. 2017.