**Representing Fire using Billboard Particles in OpenGL**

Junda Yin, Danielle Schugars, Andrew Astakhov, Andrew Schafer

*Overview*

Realistic looking fire as a computer simulated graphics goal is one of the most used and important applications of particle effects in animation. The benefits are high including cutting costs of burning sets and props as well as promoting higher safety standards on sets. In a pure animation setting, fire representations are simply used to serve a story but in this medium, there is much of a higher priority than that. This implementation will explore the use of particle effects and seek to simulate dancing flames in a 3D space.

*Technology Used*

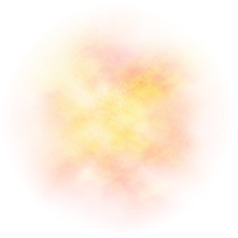
Keeping with standards set throughout this course, the platform that our project is based on is C++ and OpenGL. Additional dependencies not present in former projects include OpenGL Mathematics Library (GLM). From our research, it was found that particles can be one of four types. The types are Billboard, Trail, Beam, and Mesh Particles. Due to the nature of our final demo, the group elected to go forward with a Billboard Implementation. This specific particle type allows us to use 2d image files for texturing to represent each particle as they are rotated to always face the camera. This choice was made to give us more flexibility in altering the specific texture files and colors. A crucial class that was used is the particle emitter class. This allows the program to create the particle effect by spawning particles of similar characteristics. This allowed us to achieve the effect in a flame with groupings of brighter particles will be together and seem to flow naturally from the source.

*Starting Point*

**

*Alterations*

Texture file was changed from an arrow to a texture that has better opacity values in order to get a more realistic looking flame.

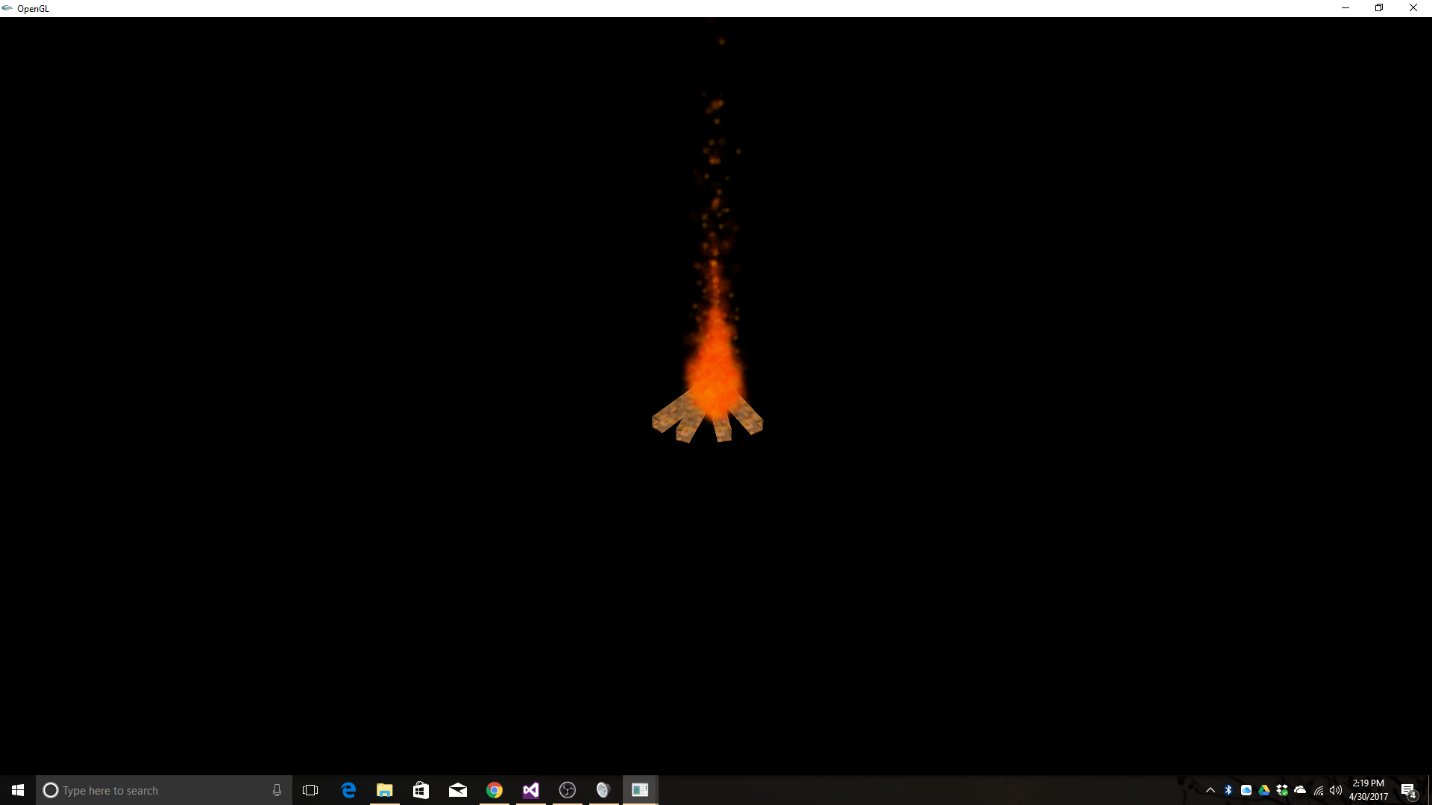


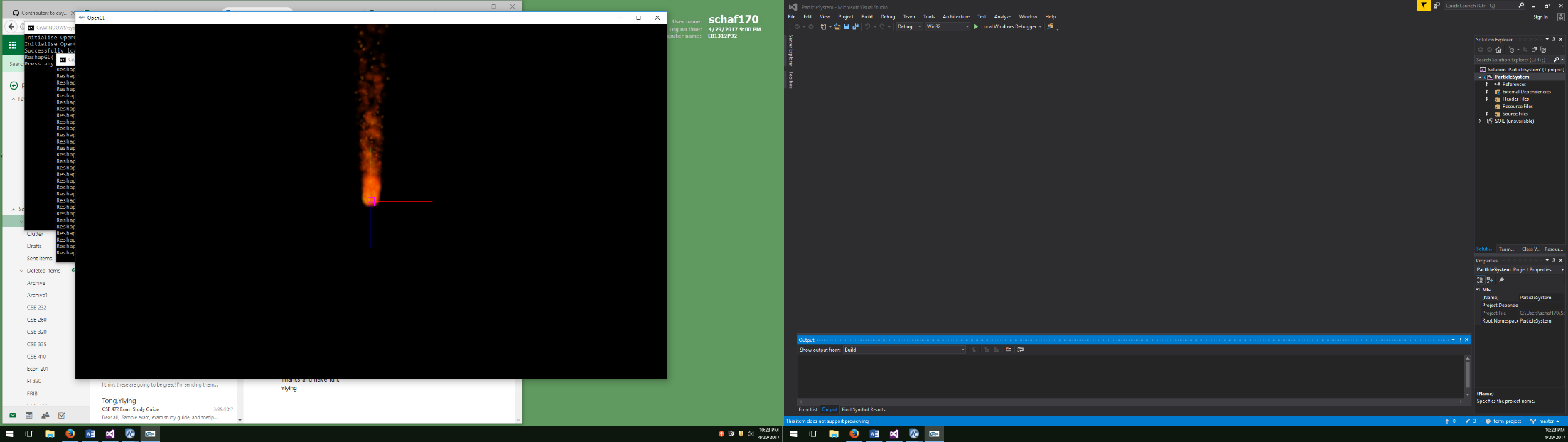
Lifespan behavior was also altered to allow for fading as the particles rise like flames would. This is also where colors would be adjusted so the images would change from white to a red as the flame left the source.

The particle emitter was also altered to make a narrower spawn width in order to more closer mimic a frame. This was accompanied with limiting movement to the positive y-axis further adding to the realism of the flame.

Finally, a wood cube was added to the solution to demonstrate how the program would be used in an animation setting.

*Finished version*





***References***

**Background**:

Fire and Flame Simulation using Particle Systems and Graphical Processing Units - <http://worldcomp-proceedings.com/proc/p2013/MSV2342.pdf>

*This article outline the basics of particle animation and helped us understand how to best implement the particle system. It’s main focus was following complex models that actual flames follow in creating a true simulation.*

Creating Fire in a Particle Editor - [www.astralax.com/articles/inflames](http://www.astralax.com/articles/inflames)

*Textures that are used for billboard particles impact the result, this article explained that in detail.*

**Technology**:

Simulating Particle Effects using OpenGL - <https://www.3dgep.com/simulating-particle-effects-using-opengl/>

*Basis for the particle system that we altered to fit the needs for flame.*

Project Two Demonstration (In-Class) - <https://www.cse.msu.edu/~cse472/>

*Accomplished tasks that were to be assigned to that project.*

**Visual Examples:**

Match Lighting in Slow Motion - <https://www.youtube.com/watch?v=EJFn6jf-hVI>

*Used to compare the effects we built with actual fire to make the simulation more true to life*