Controllable Fire using Textured Forces

Jake Lever[1] and Taku Komura[1]

[1]School of Informatics, The University of Edinburgh, UK

1 Introduction

Fire effects based on fluid dynamics have been used in motion pictures and video-games to captivate the audience and make the experience more immersive. Particle systems in particular are straightforward to visualise and understand but lack the correct physical motion that appears in real fire. Fluid simulation holds the key to the correct simulation of fire. Although the computational cost of simulation is fairly high, recent approaches for parallelising fluid simulation using the graphics processing unit (GPU) have enabled real-time animation on consumer level personal computers.

Getting the correct appearance for the fire normally requires careful control of the various parameters of the underlying physical equations. These parameters are neither intuitive nor straightforward to understand. It is especially difficult for the artists to create fire effects with a particular shape and turbulence. The main source of turbulence that occurs inside the flame is due to the pressure caused when fuel ignites and occurs on the interface between the fuel and the burning flames.

This paper presents a new method of controlling the shape of fire that is suited to both simple fire effects and more dynamic simulations with fully controllable shapes. A 2D or 3D texture mapped onto the simulation controls the strength of additional forces which control the growth of the flame and add artificial turbulence. This allows for easy control of the fire through textures that can be procedurally generated or created by an artist. Moreover, these textures can be adapted frame-by-frame to animate fire into different controllable shapes while maintaining the visually pleasing turbulence caused by burning fuel.

This method can be executed entirely on the GPU in real-time, including simulation, rendering and interaction stages. Using NVIDIA's CUDA frees the CPU to manage other parts of the animation process making it ideal for both videogames and fast animation tools for visual effects. Throughout the paper we show various types of fire that can be produced and simulated interactively by simply changing the texture pattern applied to the simulation.

2 Results

Various examples of fire were created using 2D and 3D textures. Some examples were created from keyframe textures, and some were designed by an animator.

By utilising a 2D texture that varies over time, the shape and turbulence of the fire can be adapted from frame-to-frame. One approach offsets the texture coordinate by a particular value every time step which creates the effect of a continuous moving texture (assuming that the texture was wrapped) and causes the shape and turbulence applied across the fire to be constantly changing. This is best used for general patterns such as in Figure 1 which applies a striped texture with an increasing V coordinate such that in appears to move upwards.

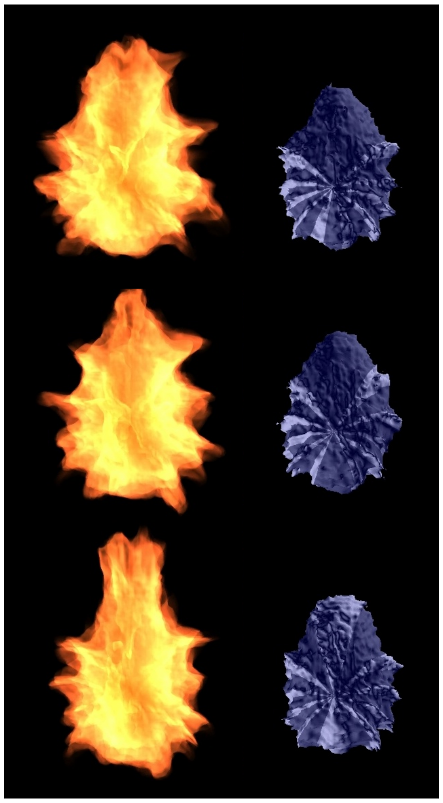
Fig. 1. Three frames from an animation of a striped texture which is moved upwards through the flame by incrementing the V texture coordinate

An alternative approach uses a 3D texture to store a set of 2D animation frames as slices. Then to animate the feature, the texture lookups incrementally move deeper through the texture going from frame to frame. By using a 3D texture, hardware linear interpolation can be used to blend between frames smoothly. An example of animating letters modeled by 2D textures blended from one to another are shown in Figure 2.

The figures shown throughout the paper are simulated in a 64x64x64 domain. The textures used were generated using the GPL licensed graphics tool Blender [1]. The images are rendered using a volume rendering approach based on [4] and [2].

The system was implemented using CUDA and executed on an NVIDIA Quadro 5800 graphics card with 4 GB memory. Figure 3 shows the achievable frame rates for simulations using an animated texture. The additional

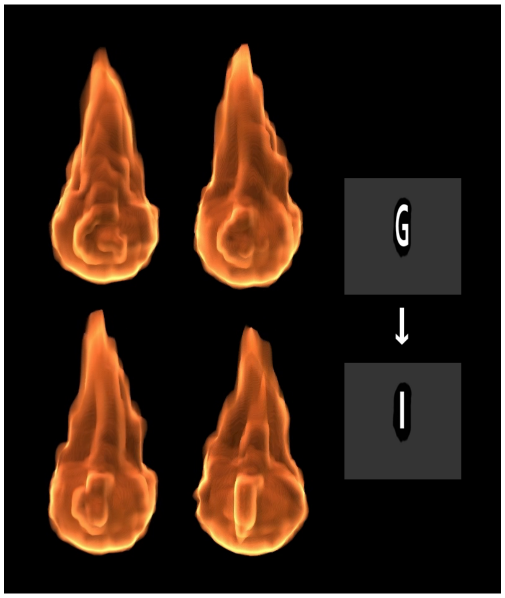

Fig. 2. A keyframe animation of letters blended from one texture to another.

cost of animating the texture is negligible. Even though 3D texturing requires more texture accesses, the actual simulation speed difference is minimal.

Domain Size	Steps/Sec
32	638
64	142
96	44.7
128	19.2
160	10.0

Fig. 3. Resulting execution speeds in simulations steps per second for cube domains on an NVIDIA Quadro 5800

3 Conclusion and Future Work

This paper introduces a real-time technique that allows easy control of the shape and turbulence of a fire using textured forces.

Our approach uses a 2D or 3D texture defined across the simulation to control the growth and turbulence of the fire. Trigonmetric based turbulence and buoyancy forces are added to a stable fluid solver in order to create a realistic fire effect.

These techniques integrate easily into a normal fluid solver and can be linked with an appropriate real-time volume rendering system. It requires minimal additional memory and the extra memory accesses are easily optimised so that the full simulation and rendering can execute in real-time.

This texturing approach offers many advantages over other techniques for animating fire. Most artists already have a deep knowledge of texturing and imaging software that will make this simulation much easier to understand. Furthermore with minor editing existing texture assets can be imported to be used as fire animations. Moreover animating the fire uses the same principals as animating drawn frames. An interactive interface allowing the user to directly influence the animation of the fire in real-time is of real importance for increasing animator work flow and gaining better results from physical simulations.

Creating arbitrary shapes of fire is a challenging problem compared to other fluids such as smoke and water, as done by other researchers [7, 5, 3, 6] . This technique allows for specific local control of the growth and turbulence of the fire and will provide an aesthetically pleasing global shape. However this requires careful control by the artist so that additional turbulence created by the noise forces does not overwhelm the shape created using the normal forces. Automatically adjusting the normal forces to interact better with noise forces is one of our future research topics.

References

[1] Blender Foundation. Blender 2.48, 2009. www.blender.org.

[2] K. Crane, I. Llamas, and S. Tariq. *Real-Time Simulation and Rendering of 3D Fluids*, volume GPU Gems 3, pages 633–675. NVIDIA, 2007.

[3] R. Fattal and D. LISCHINSKI. Target-driven smoke animation. *ACM Transactions on Graphics*, 23(3), 2004.

[4] M. Ikits, J. Kniss, A. Lefohn, and C. Hansen. *Volume Rendering Techniques*, volume GPU Gems, pages 667–692. NVIDIA, 2004.

[5] A. McNamara, A. Treuille, Z. Popovic', and J. Stam. Fluid control using the adjoint method. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23(3), 2004.

[6] L. Shi and Y. Yu. Controllable smoke animation with guiding objects. *ACM Transactions on Graphics*, 24(1):140–164, 2005.

[7] A. Treuille, A. McNamara, Z. Popovic', and J. Stam. Keyframe control of smoke simulations. *ACM Transactions on Graphics*, 22(3):716–723, 2003.