Shaders for Gamedev VFX

Ryan King
Graphics

Abstract

This report explores the implementation of common visual effects (VFX) in game development through the use of GLSL (OpenGL Shading Language) shaders. By writing custom fragment and vertex shaders, I recreate a variety of real-time effects such as trails, screen space shaders, outlines, and procedural graphics. Each effect is designed to run efficiently on modern GPUs and demonstrates how low-level shader programming can be leveraged to achieve high-performance visual fidelity in interactive environments. The project emphasizes both the technical and artistic aspects of VFX, highlighting key optimization techniques and design considerations for integrating shaders into real-time rendering pipelines.

Introduction

This project is a step-by-step breakdown of recreating visual effects (VFX) commonly used in game development, with a focus on both the technical and mathematical foundations as well as the artistic decisions behind each effect. The goal is to provide a broad overview of techniques frequently employed in real-time VFX workflows. Six custom GLSL shaders are showcased, each demonstrating different vfx principles.
Ultimately, this work aims not just to inform, but to inspire others to delve into the creative and technical world of VFX.

1. Electricity Shader
2. Wiggly Trail Shader
3. Death Fog Screenspace Shader
4. Pixel Perfect Outline Shader
5. Procedural Flag Shader
6. Procedural Molten Lava Shader

The techniques shown within include but are not limited to:

Panning / Scrolling
- Modulo -> Fract optimization
- Domain Warping

Texture Sampling
- Color lookup tables
- Noise Texture sampling
- Curve Textures
- Viewport Sampling

Texture Painting

- Post processing
- Filtering
- Tiling

Shaping Functions

- Polar UVs
- Scaling and Shifting
- Curve Textures
- Oscillation
- Additive Signal Processing
- Logical Signal Processing
- Derivative Sampling

Stepify

Randomness

Branch Optimization

Vertex Shading

Noise

- Procedural Perlin Noise
- Procedural Fractal Brownian Motion
- Layered Domain Warping
- Voronoi Noise
- Worley Noise
- Modified Worley Noise

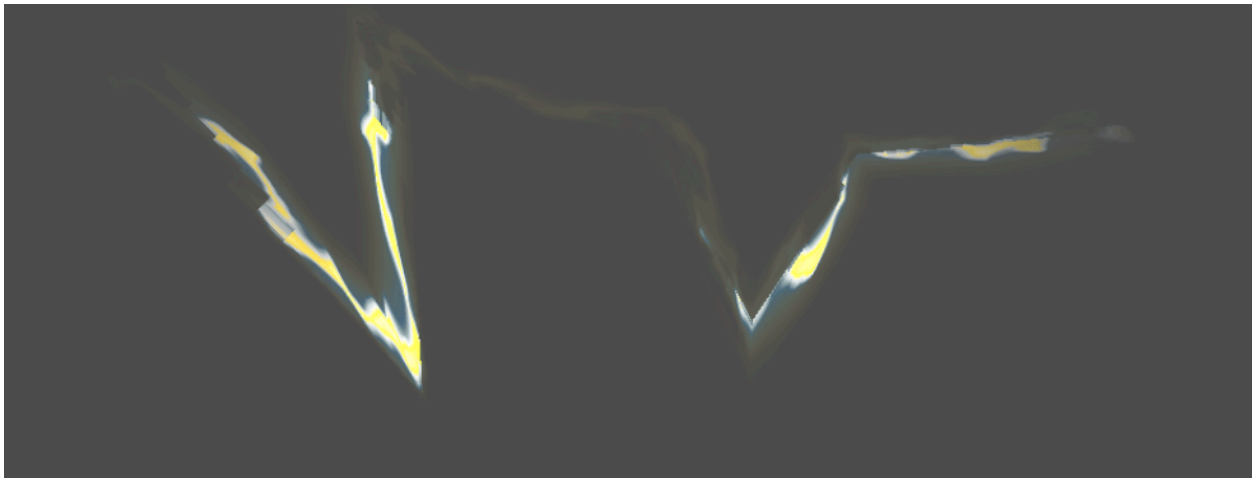
Prior work:

The techniques demonstrated in this project are not novel in themselves; they are foundational to the field of real-time visual effects and widely used in game development. Many of the underlying algorithms such as Perlin noise, fractal Brownian motion (FBM), Voronoi diagrams, and Worley noise are well-established and extensively documented in both academic literature and industry resources.

Results and Contribution:

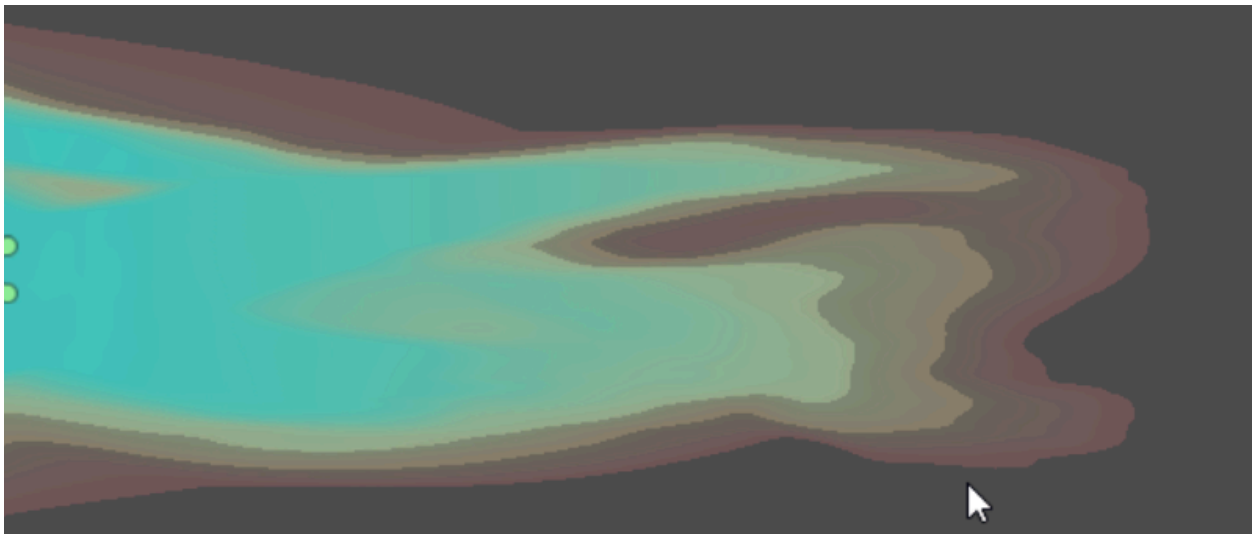
While the core techniques are well-established, all shaders presented in this project are original implementations. Several showcase uncommon or creative approaches, such as using derivative sampling for procedural flag animation or modifying Worley noise to compute borders between Voronoi cells. The primary contribution of this work lies not in introducing new algorithms, but in clearly showcasing a broad range of VFX techniques. By emphasizing both technical construction and artistic intent, this project aims to demystify shader-based effects in game development and inspire others to explore the creative potential of real-time VFX.

1. Electricity Shader:



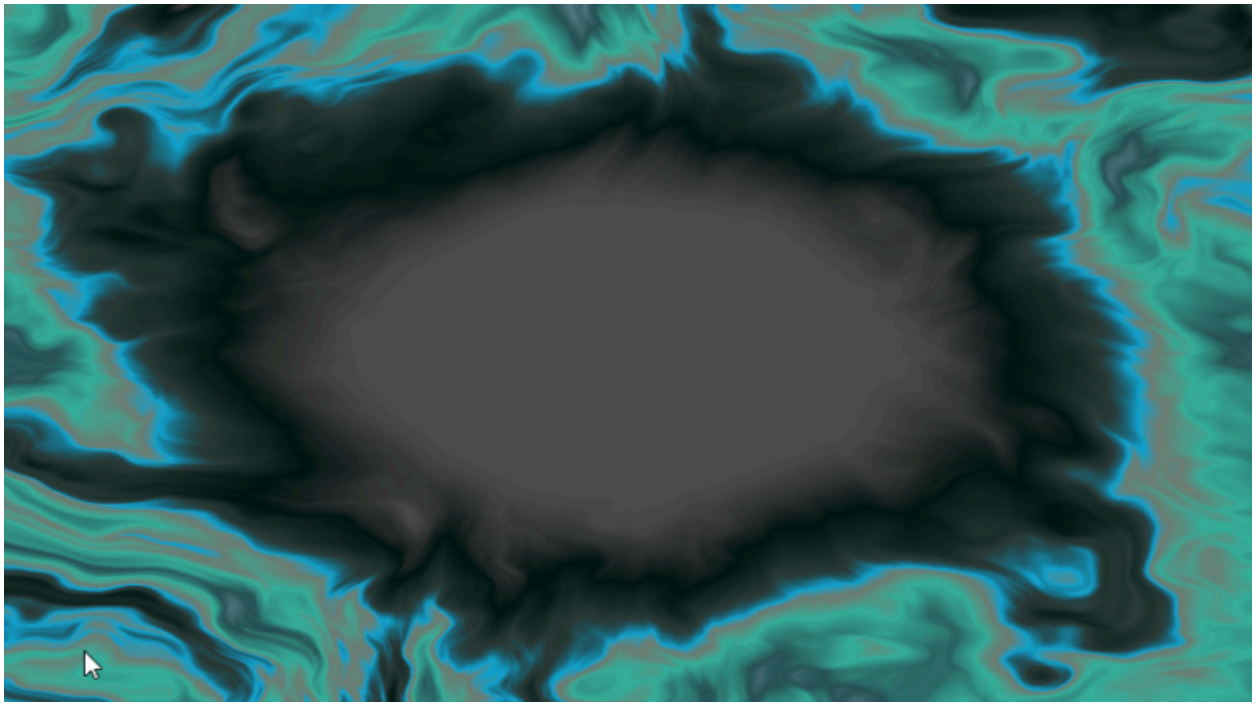
This shader demonstrates primarily the importance of texture painting and texture sampling. It demonstrates panning, and basic signal processing, and acts as a starting point for understanding VFX shaders.

2. Wiggly Trail Shader



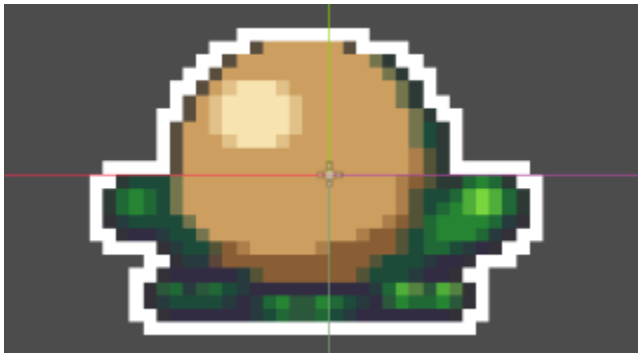
This shader builds off of the electricity shader, showing more advanced signal processing techniques such as stepification to reduce granularity, and value modulation.

3. Death Fog Screenspace Shader



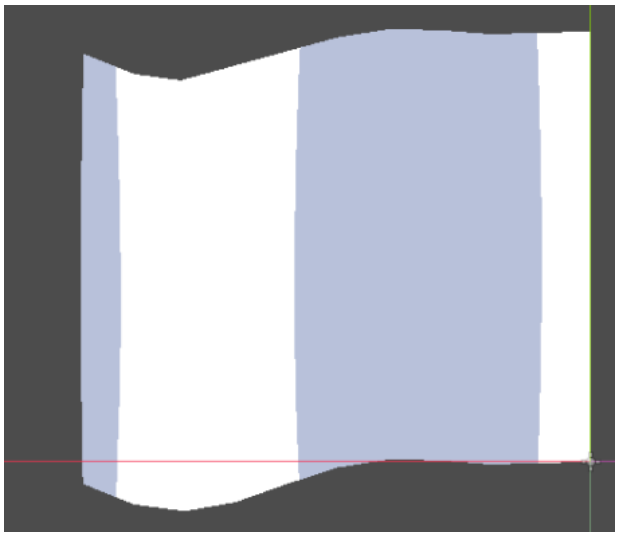
This shader demonstrates domain warping, polar UVs, fractal brownian motion, perlin noise, and randomness to create a screen space shader.

4. Hyperoptimized Pixel Perfect Outline Shader



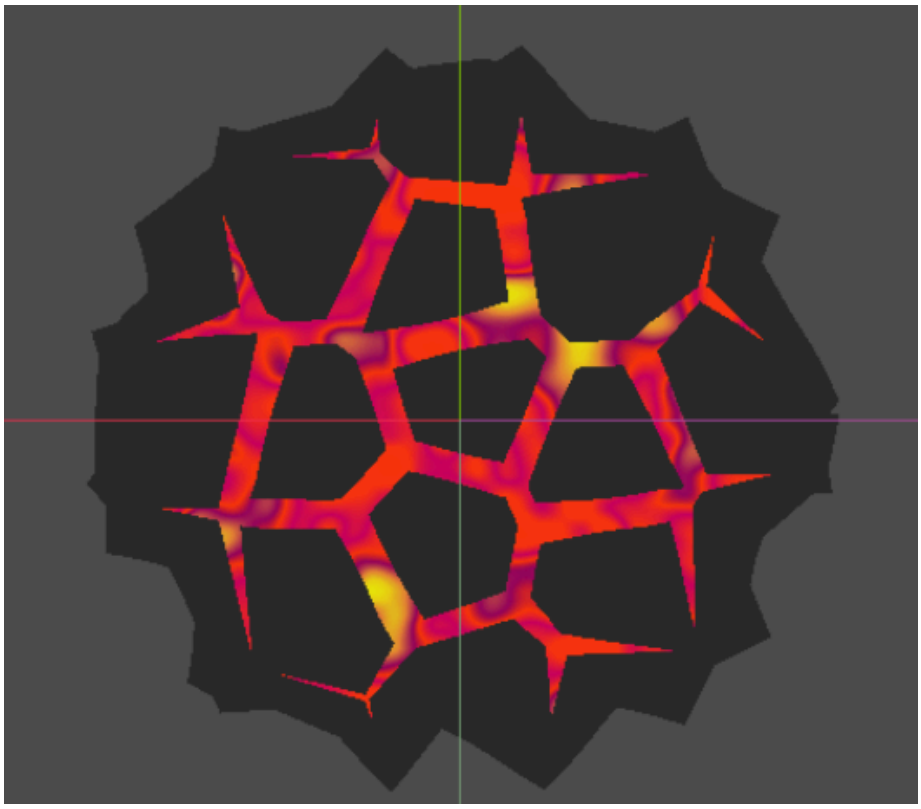
This shader leverages pixel-perfect art to create a highly optimized outline effect. It serves as a demonstration of several advanced concepts, including branchless programming, multiplicative logic-based signal processing, and viewport-relative sampling. Additionally, it highlights the importance of understanding the specific use case of a shader in order to optimize it effectively. The implementation also provides an opportunity to discuss key optimization principles and performance-conscious design practices in shader development.

5. Procedural Flag Vertex Shader



This shader is a simple vertex shader to demonstrate how vertex shaders can be leveraged to create procedural graphics. It also showcases a very unusual trick where we sample the euler's derivative of the vertex distortion to add shadows.

6. Procedural Molten Rock Shader



This shader showcases worley noise and voronoi noise, as well as showing off a modified worley noise algorithm that allows us to calculate borders between voronoi cells. It also leverages domain warping, and showcases a modified simpler and faster fractal brownian motion algorithm.

Conclusion:

This project explored the recreation of foundational VFX techniques in game development through original GLSL shader implementations. By breaking down each effect step by step, it bridged the gap between technical understanding and artistic intent, showing how low-level graphics programming can be both efficient and expressive. While the techniques themselves are well-established, their thoughtful combination and presentation offer a practical and approachable resource for those interested in real-time visual effects. Ultimately, this work aims not just to inform, but to inspire others to delve into the creative and technical world of VFX.