Interactive fractal flames with CUDA and OpenGL

Chris Foster

ROAMES

March 22, 2011

What is a fractal?

- Fractional (Hausdorff) dimension $>$ topological dimension. Power law for number of covering boxes: $N(\epsilon) \sim 1/\epsilon^D$
- (Quasi-) self-similar at all scales

Iterated function systems (IFS)

- Hutchinson 1981 [1]: general set of strictly self-similar fractals
- Popularised by Barnsley [2] in *Fractals Everywhere*, 1988
- Set of N contractive functions $F_i \colon \mathbb{R}^2 \to \mathbb{R}^2$
- "Attractor" obeys recursive set equation

$$A_{k+1} = \bigcup_{i=1}^{N} F_i(A_k) \qquad A_k \to A \text{ as } k \to \infty$$

- F_i traditionally *affine* (rotation/translation/scaling)

Flame Fractals

- Invented by Draves & Reckase 2003 [3] (flam3/electric sheep)
- Non-affine F_i: more artistic flexibility

$$F_i = P_i\left(\sum_m v_{im} V_m\big(Q_i(x,y)\big) \right)$$

- V_m are nonlinear "variations"; maps P_i and Q_i are affine:

$$Q_i(x,y) = \big(a_i x + b_i y + c_i,\ d_i x + e_i y + f_i\big)$$

Flame Fractals

Monte Carlo sampling

```
P = (0,0,0)     # Arbitrary position
C = (0,0,0)     # Arbitrary colour

for i in range(0,maxIter):
    func = choose_function_at_random(funcs)
    P = func(P)
    C = 0.5*(func.C + C)
    if i > discardCutoff:
        plot_point(P, C)
```

Implementation overview

- Generate point list using CUDA
- Render points with additive OpenGL blending
- HDR tone mapping and gamma correction

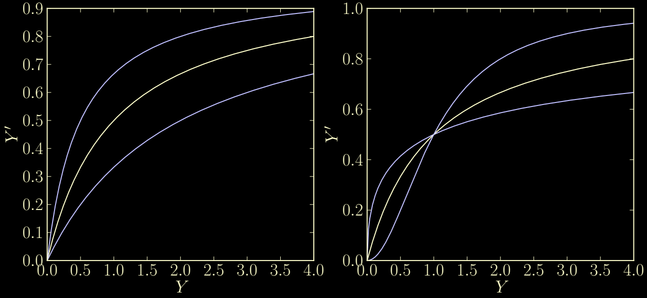
GPU flames

- History: S. Green [4], flam4 (keldor314), GPU gems [5]
- My implementation in CUDA is pretty naive:
- 50 points from each of 40,000 threads (total 2,000,000) generated into vertex buffer object
- curand for random numbers

OpenGL High Dynamic Range Pipeline

- Accumulate into offscreen float-precision FBO
- Tone mapping & gamma correction with GLSL:

$$\mathrm{RGB}_{\mathrm{linear}} \rightarrow \mathrm{xyY} \rightarrow \mathrm{xyY}' \rightarrow \mathrm{RGB}'_{\mathrm{linear}} \rightarrow \mathrm{RGB}'_{\mathrm{gamma}}$$



OpenGL High Dynamic Range Pipeline

- Accumulate into offscreen float-precision FBO
- Tone mapping & gamma correction with GLSL:

$$\mathrm{RGB}_{\text{linear}} \to \mathrm{xyY} \to \mathrm{xyY}' \to \mathrm{RGB}'_{\text{linear}} \to \mathrm{RGB}'_{\text{gamma}}$$

Future Performance Tuning

- Warp divergence with many variations and many functions
- GPU gems algorithm [5]

https://github.com/c42f/flamed

References

J. E. Hutchinson.
Fractals and self similarity.
Indiana Univ. Math. J., 30:713, 1981.

M. F. Barnsley.
Fractals Everywhere.
Academic Press, Boston, 1988.

S. Draves and E. Reckase.
The fractal flame algorithm.
http://flam3.com/flame_draves.pdf, 2003.
downloaded 20 March 2012.

S. G. Green.
Gpu-accelerated iterated function systems.
In Juan Buhler, editor, *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

C. Schied, J. Hanika, H. Dammertz, and H. P. A. Lensch.
High-performance iterated function systems.
In W.-M. W. Hwu, editor, *GPU computing gems, Emerald Edition*. Morgan Kaufmann, Burlington, 2011.