The Tricks Up Our Sleeves

A Walkthrough of the Special FX of Uncharted 3: Drake's Deception

Keith Guerrette
Lead Visual Effects Artist, Naughty Dog
www.keithguerrette.com

Mouse-over this icon to see my narration!

GAME DEVELOPERS CONFERENCE 2012
SAN FRANCISCO, CA
MARCH 5-9, 2012
EXPO DATES: MARCH 7-9

Overview

- Evolution of our FX pipeline across the Uncharted Franchise
- Specific FX Challenges of Uncharted 3 (and the solutions that we came up with)
- A few lessons we've learned, and plenty we haven't

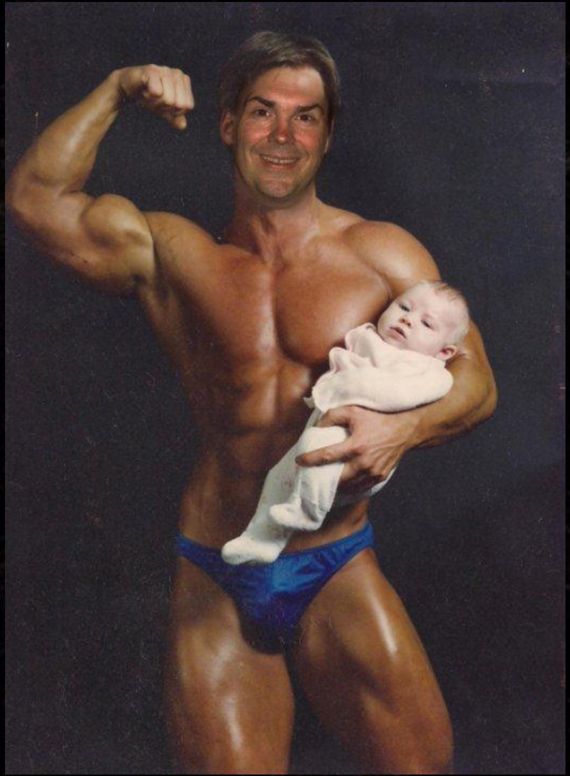
The Team

Marshall Robin

Genius, FX Programmer Extraordinaire, and All-Round Cool Guy

Marshall Robin

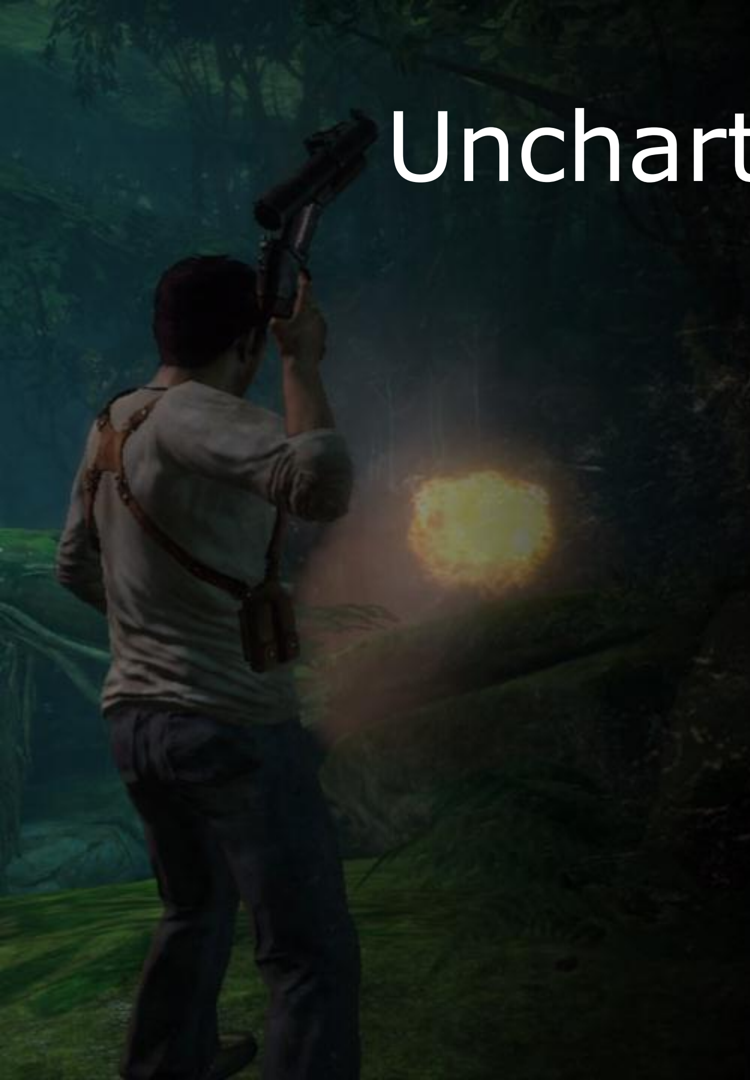
Genius, FX Programmer Extraordinaire, and All-Round Cool Guy



Evolution of the Tools

UNCHARTED
DRAKE'S FORTUNE™

Uncharted 1 FX Pipeline

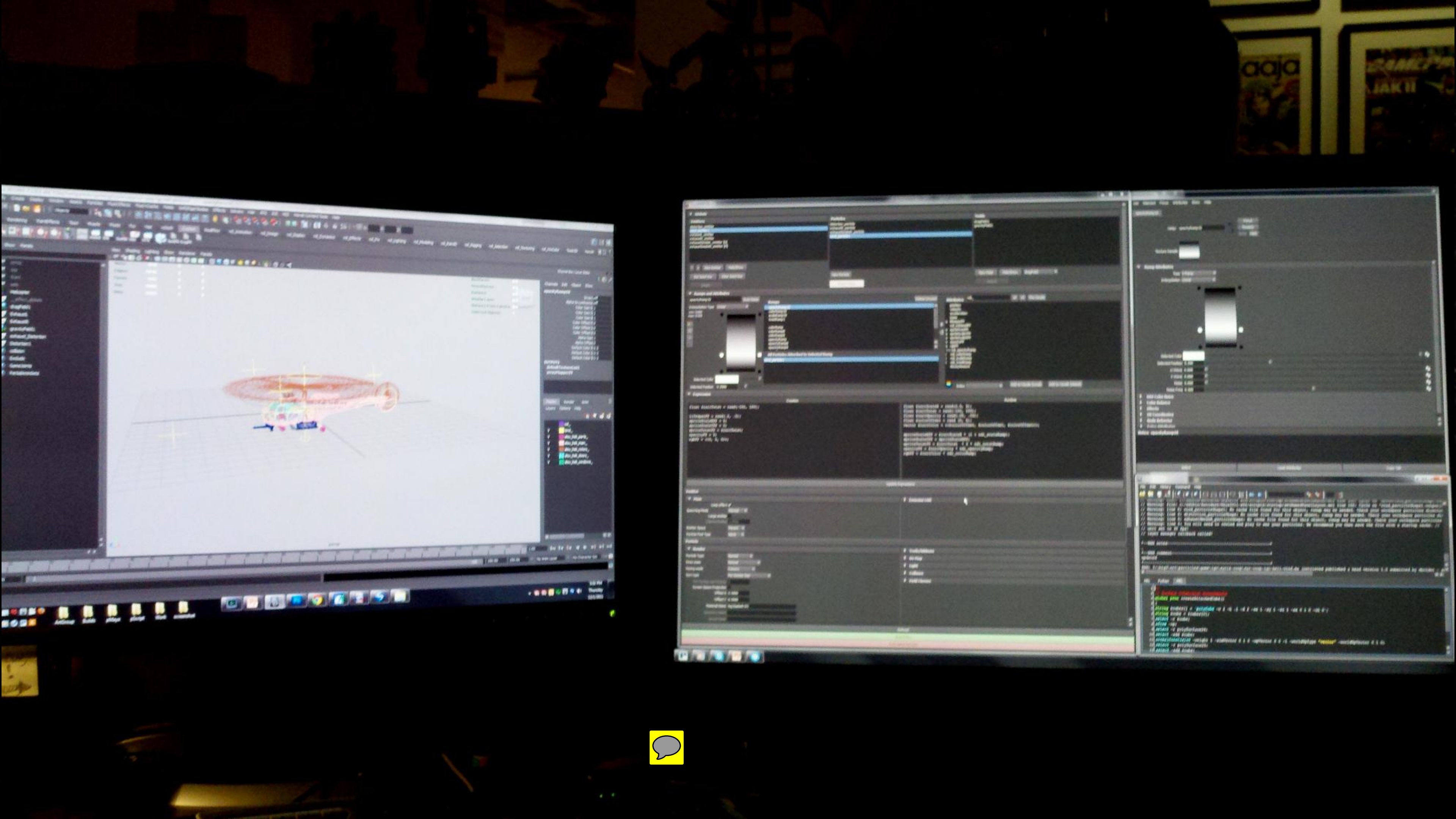


- FX were hand-written in a scripting language similar to LISP.
- Shaders were hand written in HLSL

UNCHARTED 2
AMONG THIEVES™

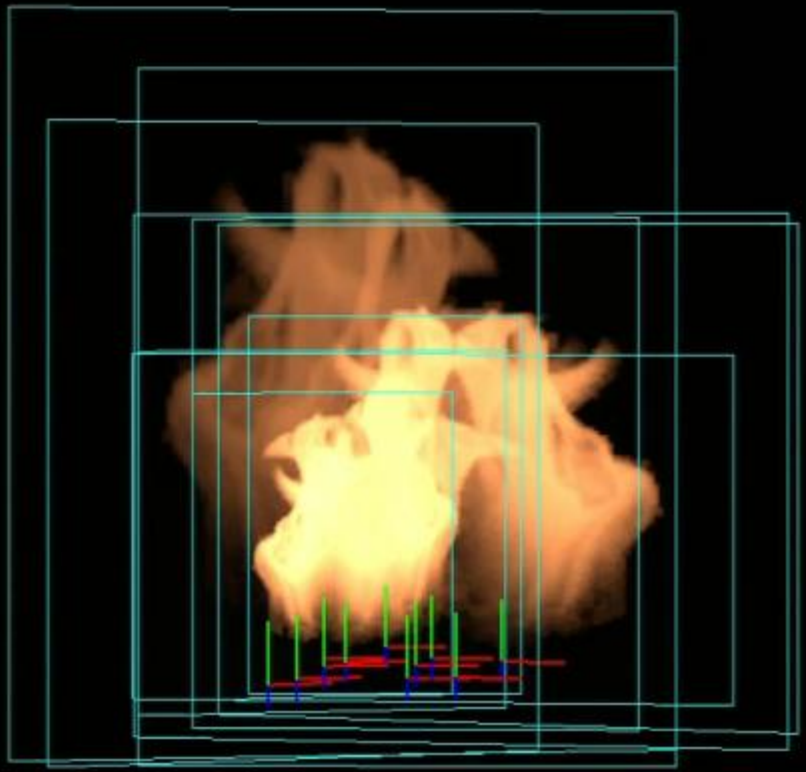
Uncharted 2 FX Pipeline

- Goals:
  - Artist friendly pipeline
  - Freedom and Power
  - Meet artistic standards set by the top of the industry



Static Materials

Static Materials



Static Materials

Static Materials

Uncharted 2 FX Pipeline

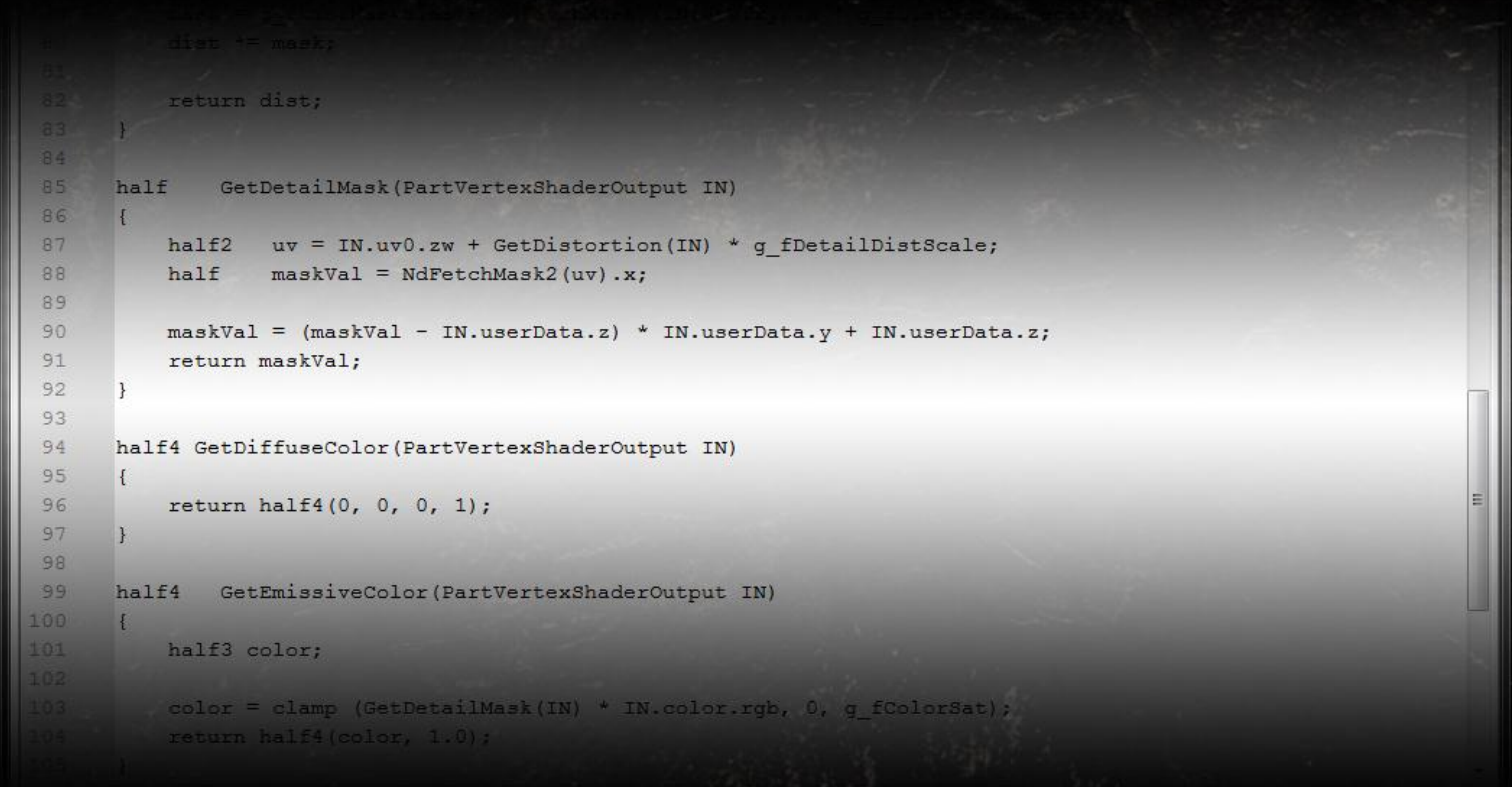
- Problems
  - Maya's sprite engine is terrible
  - We had to build our own controls, functions, and better workflow

Flipbook Materials

64 frames
512 x 512

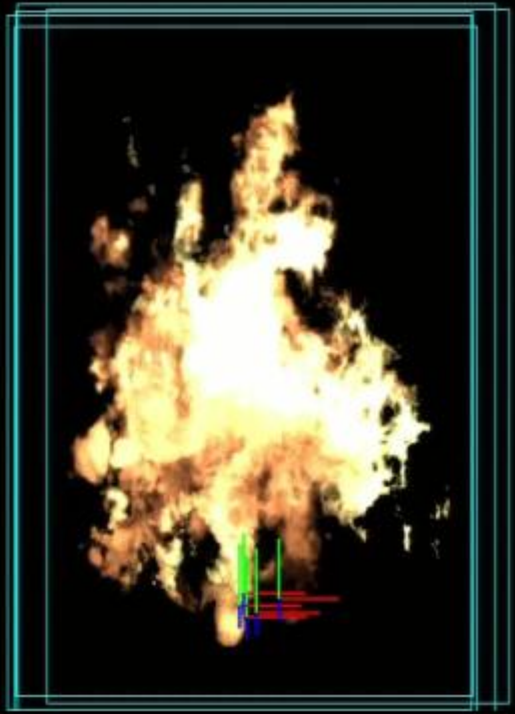
32 frames
512 x 512

Dynamic Materials



```
    dist += mask;

    return dist;
}

half    GetDetailMask(PartVertexShaderOutput IN)
{
    half2   uv = IN.uv0.zw + GetDistortion(IN) * g_fDetailDistScale;
    half    maskVal = NdFetchMask2(uv).x;

    maskVal = (maskVal - IN.userData.z) * IN.userData.y + IN.userData.z;
    return maskVal;
}

half4 GetDiffuseColor(PartVertexShaderOutput IN)
{
    return half4(0, 0, 0, 1);
}

half4   GetEmissiveColor(PartVertexShaderOutput IN)
{
    half3 color;

    color = clamp (GetDetailMask(IN) * IN.color.rgb, 0, g_fColorSat);
    return half4(color, 1.0);
}
```

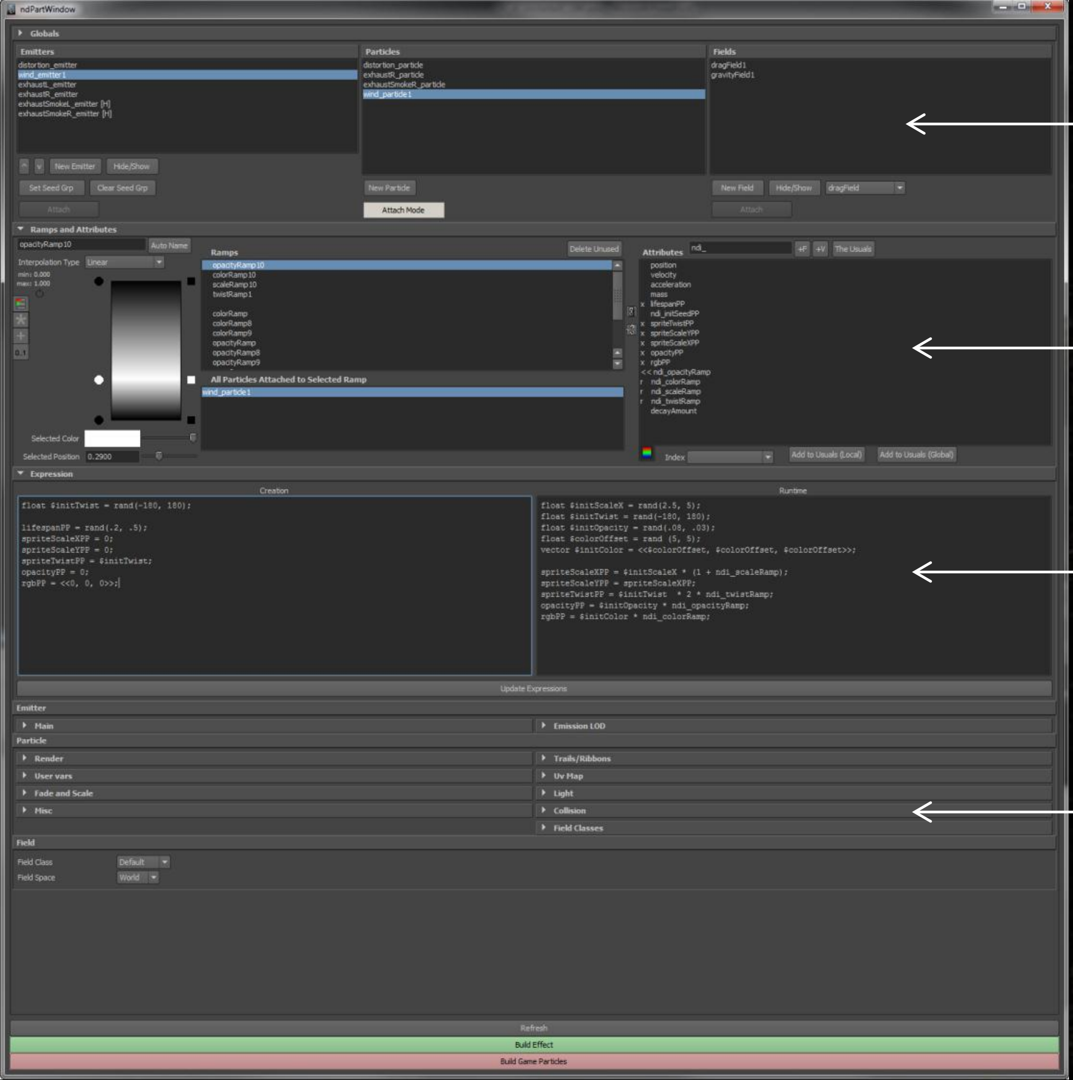
Uncharted 2 Fire Material



UNCHARTED 3
DRAKE'S DECEPTION™

Uncharted 3 FX Pipeline

- Goals:
  - Remove dependency on Programmers
  - Improve efficiency of workflow
  - Expose more control the tools & artists



ndPartWindow

Globals

Emitters: distortion_emitter, wind_emitter1, exhaustL_emitter, exhaustR_emitter, exhaustSmokeL_emitter [H], exhaustSmokeR_emitter [H]

Particles: distortion_particle, exhaustR_particle, exhaustSmokeR_particle, wind_particle1

Fields: dragField1, gravityField1

Ramps and Attributes

opacityRamp10 | Interpolation Type: Linear | min: 0.000 | max: 1.000

Ramps: opacityRamp10, colorRamp10, scaleRamp10, twistRamp1, colorRamp, colorRamp8, colorRamp9, opacityRamp, opacityRamp8, opacityRamp9

All Particles Attached to Selected Ramp: wind_particle1

Attributes: position, velocity, acceleration, mass, x lifespanPP, ndi_initSeedPP, x spriteTwistPP, x spriteScaleYPP, x spriteScaleXPP, x opacityPP, x rgbPP, << ndi_opacityRamp, r ndi_colorRamp, r ndi_scaleRamp, r ndi_twistRamp, decayAmount

Selected Color | Selected Position: 0.2900

Expression

Creation

```
float $initTwist = rand(-180, 180);

lifespanPP = rand(.2, .5);
spriteScaleXPP = 0;
spriteScaleYPP = 0;
spriteTwistPP = $initTwist;
opacityPP = 0;
rgbPP = <<0, 0, 0>>;
```

Runtime

```
float $initScaleX = rand(2.5, 5);
float $initTwist = rand(-180, 180);
float $initOpacity = rand(.08, .03);
float $colorOffset = rand (5, 5);
vector $initColor = <<$colorOffset, $colorOffset, $colorOffset>>;

spriteScaleXPP = $initScaleX * (1 + ndi_scaleRamp);
spriteScaleYPP = spriteScaleXPP;
spriteTwistPP = $initTwist  * 2 * ndi_twistRamp;
opacityPP = $initOpacity * ndi_opacityRamp;
rgbPP = $initColor * ndi_colorRamp;
```

Emitter: Main, Emission LOD

Particle: Render, User vars, Fade and Scale, Misc, Trails/Ribbons, Uv Map, Light, Collision, Field Classes

Field: Field Class: Default | Field Space: World

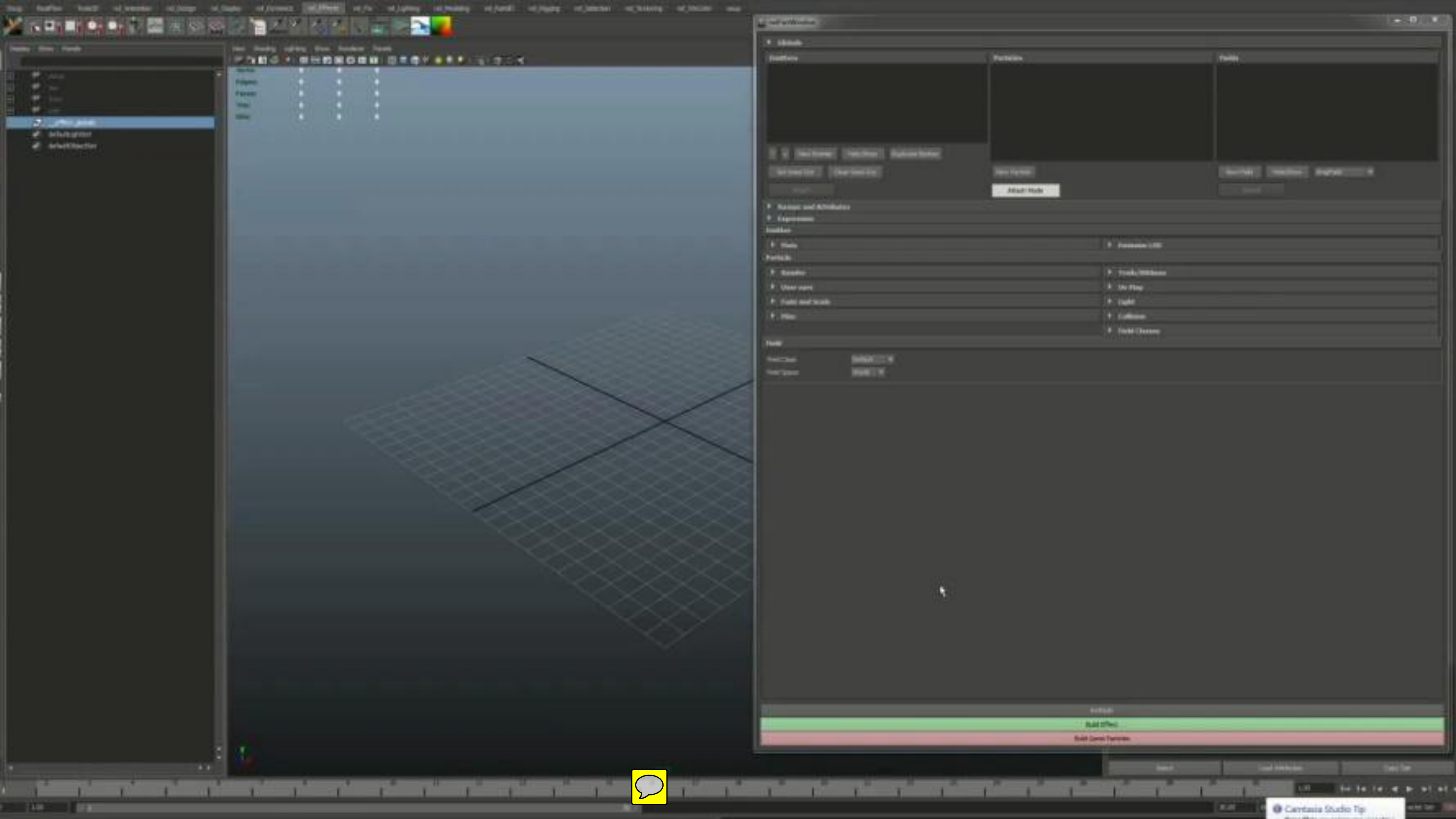
Particle System & Emitter List

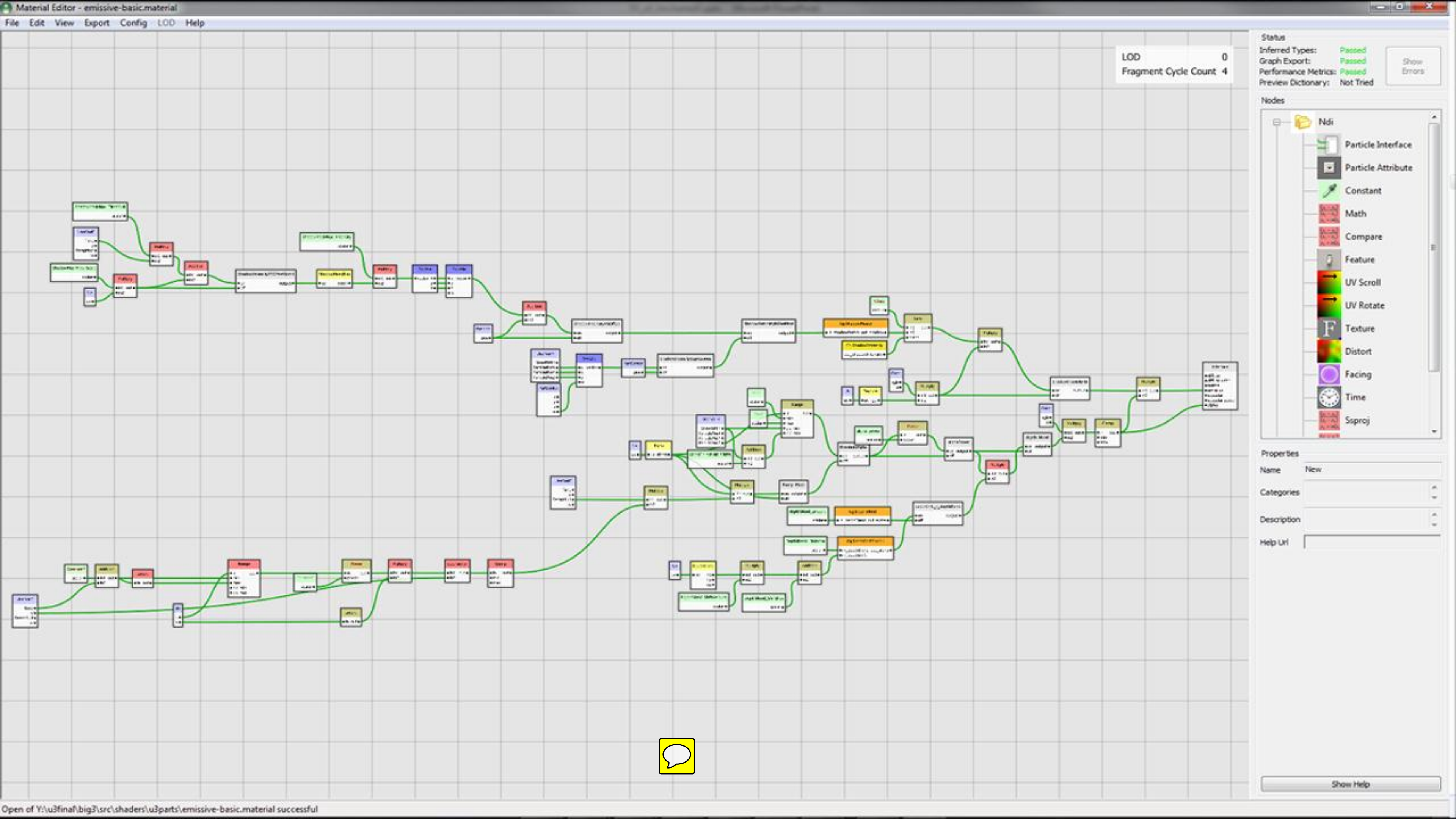
Ramp & Attribute Controls

Creation & Runtime Expression Controls

Tons of other options, including:

Material Assignments, Spawn Methods, Custom Material Variables, Collisions, Sounds, Lights, Global Fields, UV Controls, Trails, Ribbons, etc


Globals
Emitters
Particles
Fields
Attach Node
Ramps and Attributes
Expressions
Emitter
Rate
Particle
Color and Scale
Misc
UV Map
Light
Collision
Field
Build Effect
Camtasia Studio Tip



Material Editor - emissive-basic.material

File Edit View Export Config LOD Help

LOD 0
Fragment Cycle Count 4

Status
Inferred Types: Passed
Graph Export: Passed
Performance Metrics: Passed
Preview Dictionary: Not Tried

Show Errors

Nodes
- Ndi
  - Particle Interface
  - Particle Attribute
  - Constant
  - Math
  - Compare
  - Feature
  - UV Scroll
  - UV Rotate
  - Texture
  - Distort
  - Facing
  - Time
  - Ssproj

Properties
Name: New
Categories:
Description:
Help Url:

Show Help

Open of Y:\u3final\big3\src\shaders\u3parts\emissive-basic.material successful

Uncharted 3 FX Pipeline

- Current Readable Particles Attributes:
  - Ramp output (with custom V inputs)
  - Position (world, local)
  - Velocity (world, local)
  - Age (particle)
  - Time (emitter)
  - Bouncecount
  - Timedelta
  - Bounce Count

Uncharted 3 FX Pipeline

- Current Particle Expression Functions:
  - +-x/
  - Modulus
  - Random
  - Linstep & Smoothstep
  - Clamp
  - Magnitude
  - Sign
  - Sin & Cos

Uncharted 3 FX Pipeline

- Very powerful, math oriented pipeline
- Many different types of controls & customizations
- Fully supportive team of programmers
    - Open communication
    - No Politics
    - Easy, understanding discussions of priorities

Uncharted 3 FX Pipeline

- Any attribute in a shader can be controlled at run-time
  - Up to 8 real-time inputs into the shader, not including the vertex color and vertex opacity (12 Total)

The FX Challenges of Uncharted 3

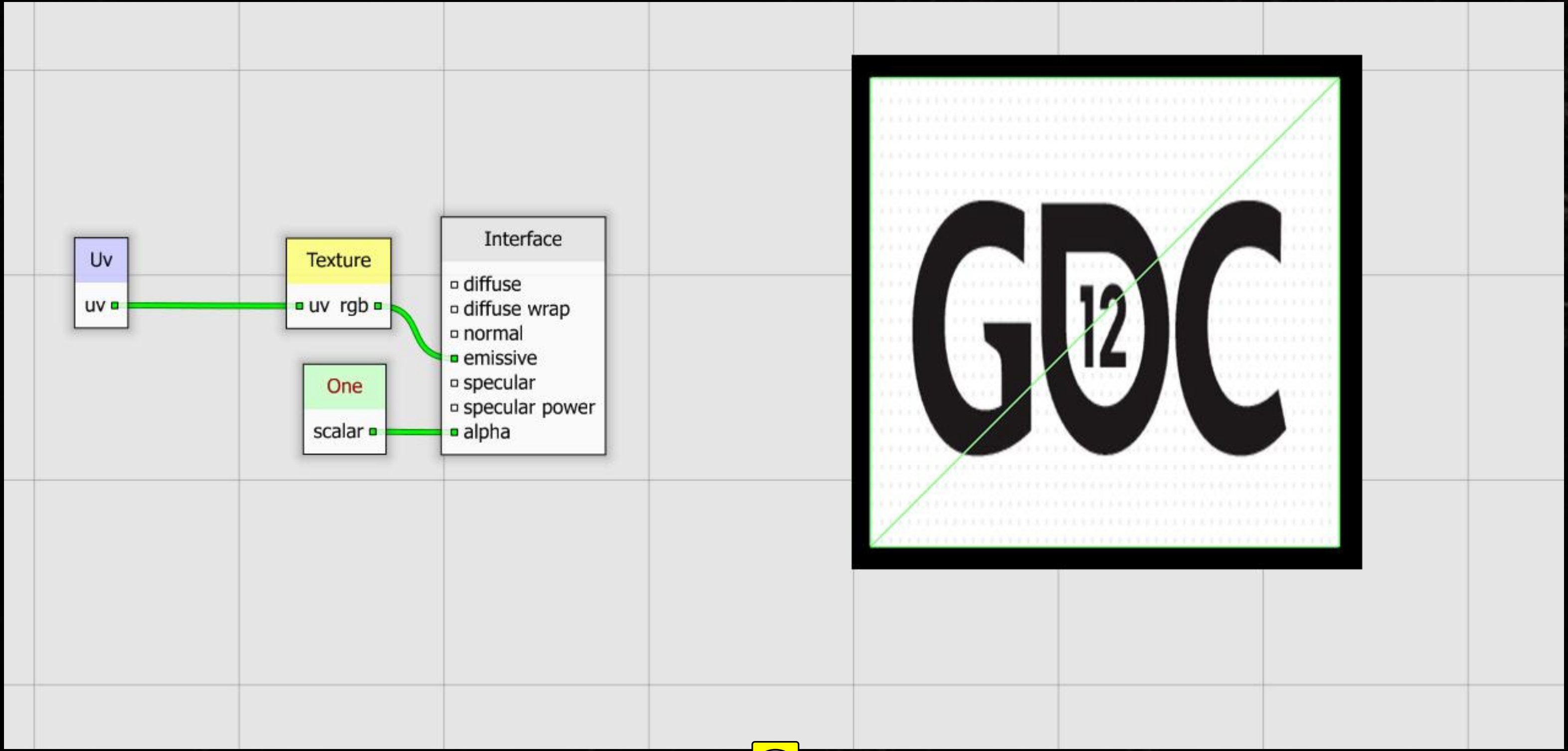
Challenge:

How do we utilize dynamic materials to create *complex detail* and *motion* within the particle system?

Creating Motion in Particles

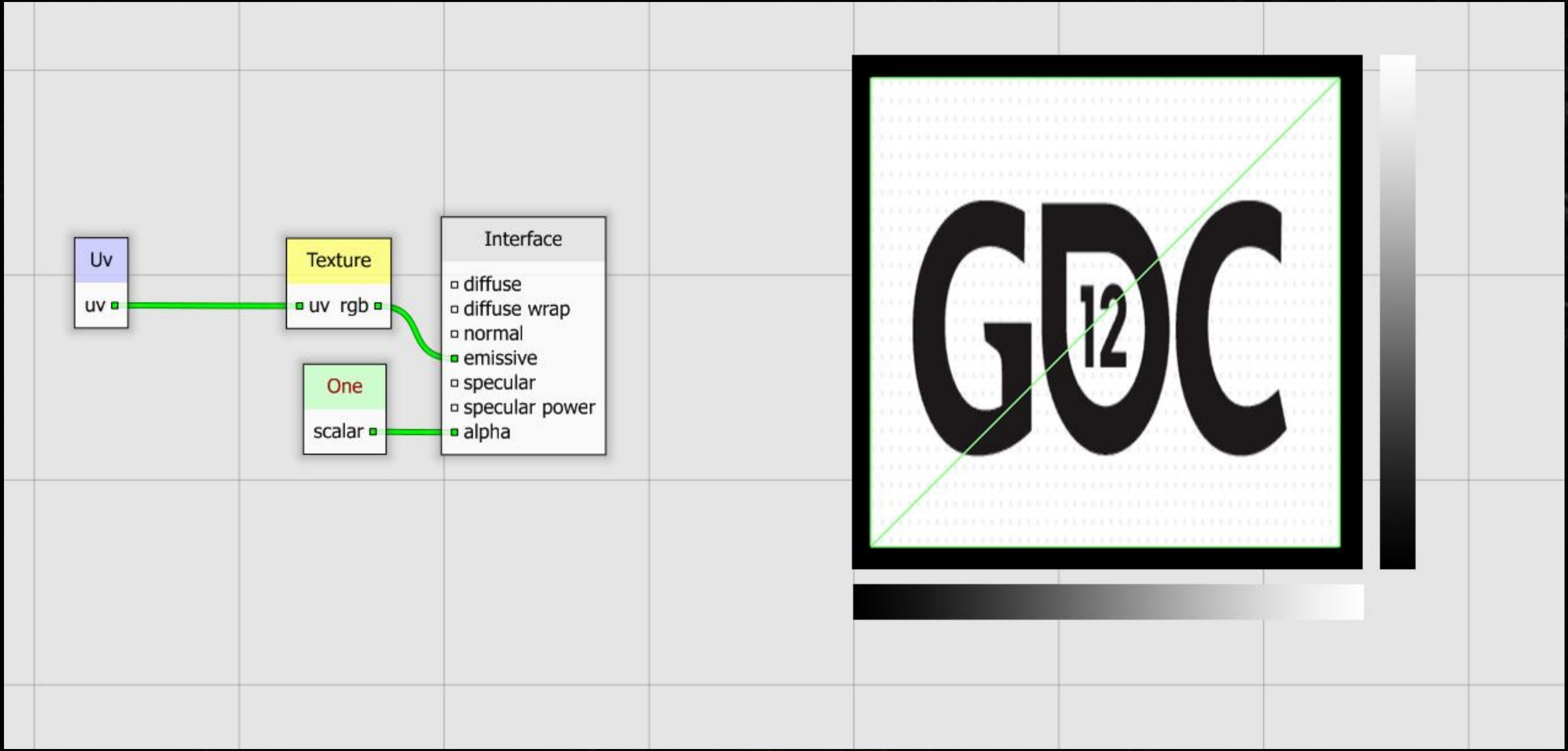
Creating Motion in Particles

Prerequisite Knowledge: UV Math



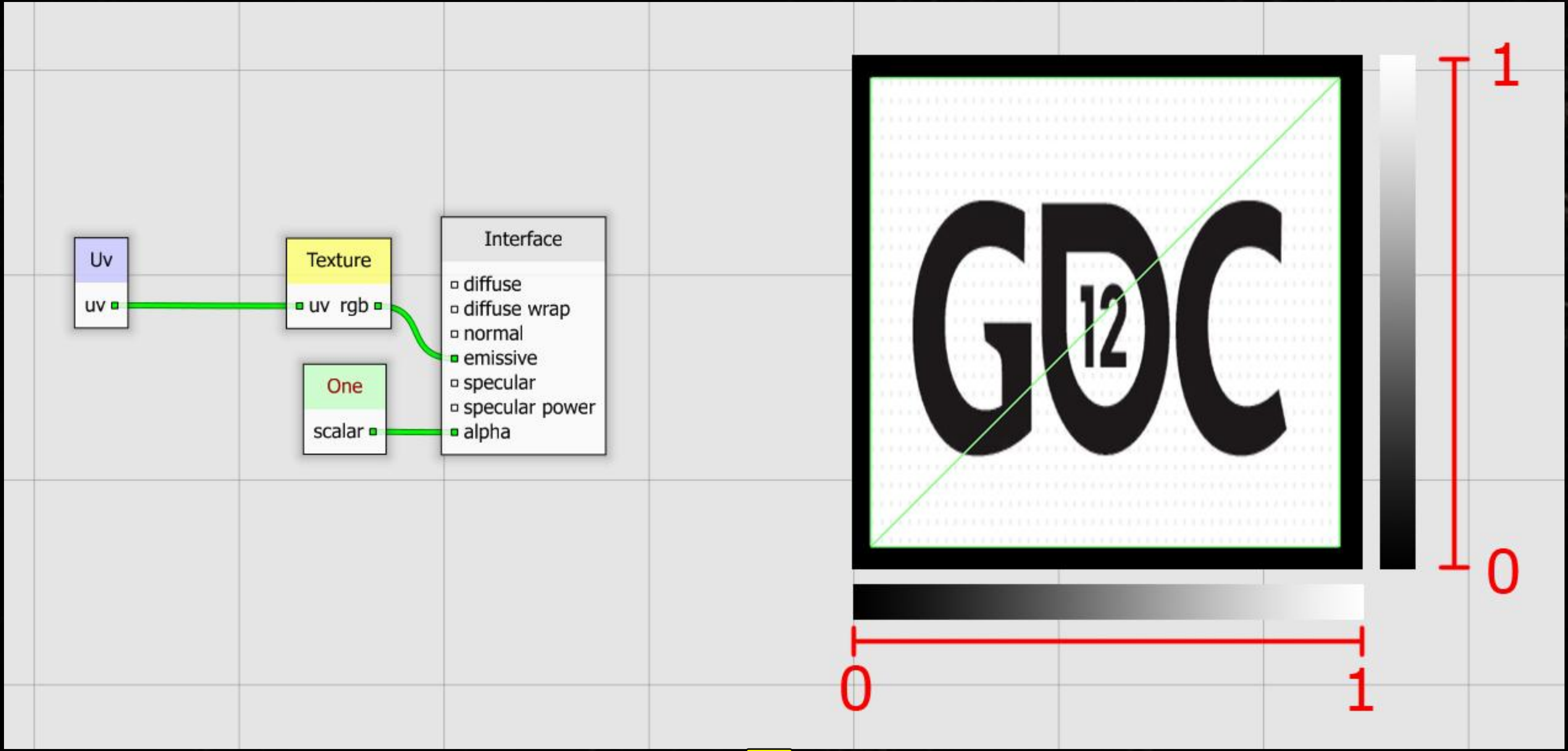
Creating Motion in Particles

Prerequisite Knowledge: UV Math



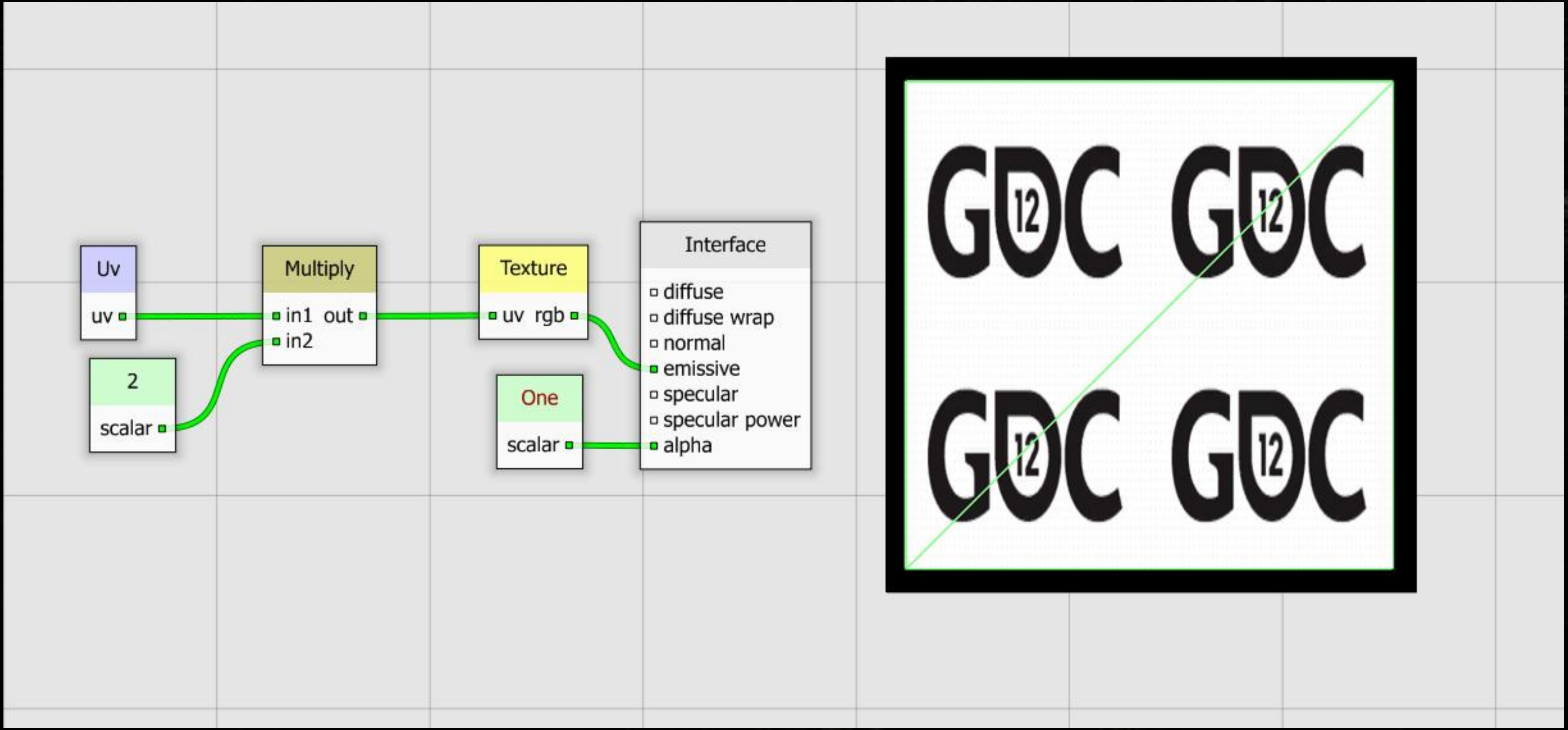
Creating Motion in Particles

Prerequisite Knowledge: UV Math



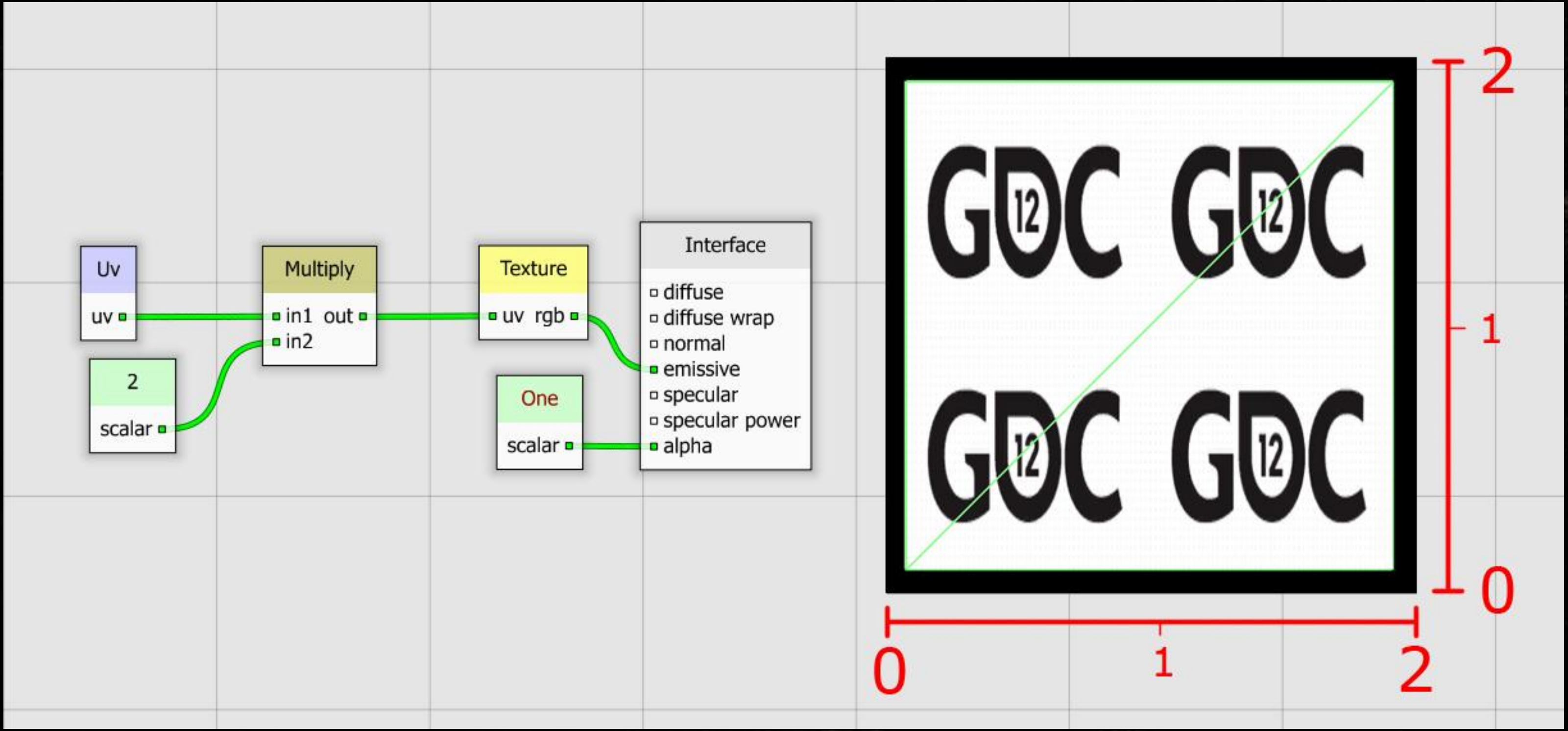
Creating Motion in Particles

Prerequisite Knowledge: UV Math



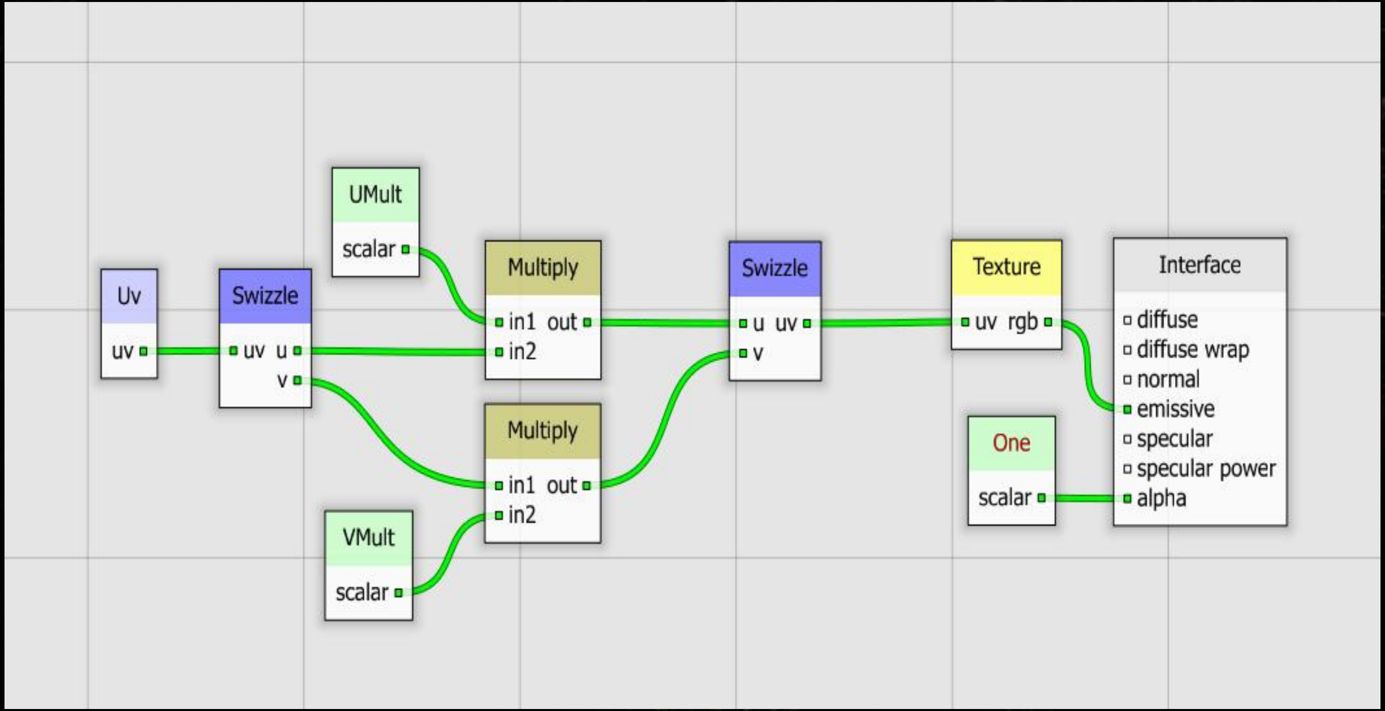
Creating Motion in Particles

Prerequisite Knowledge: UV Math



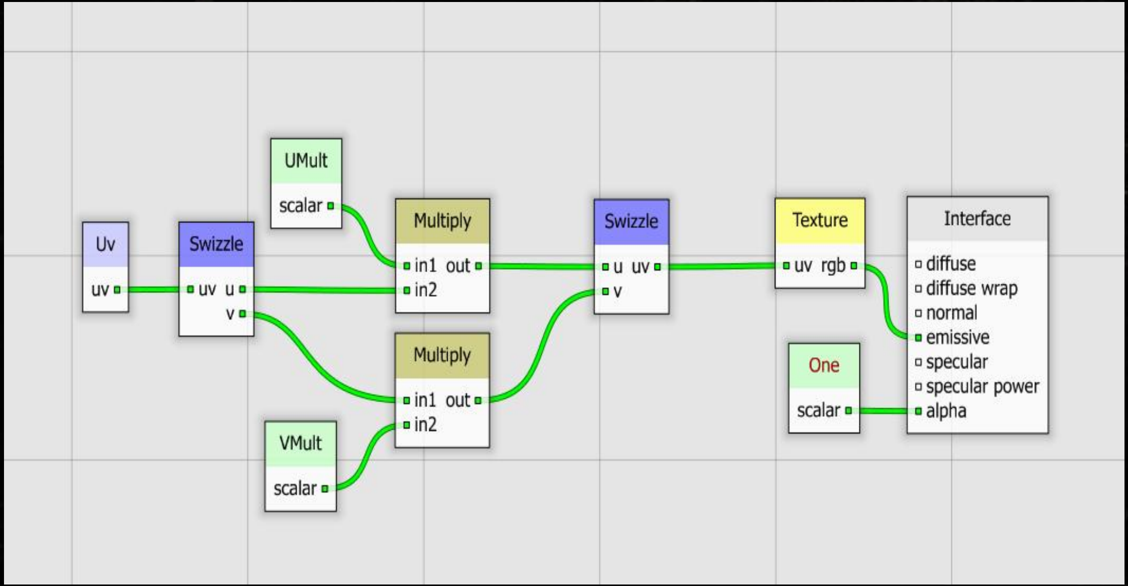
Creating Motion in Particles

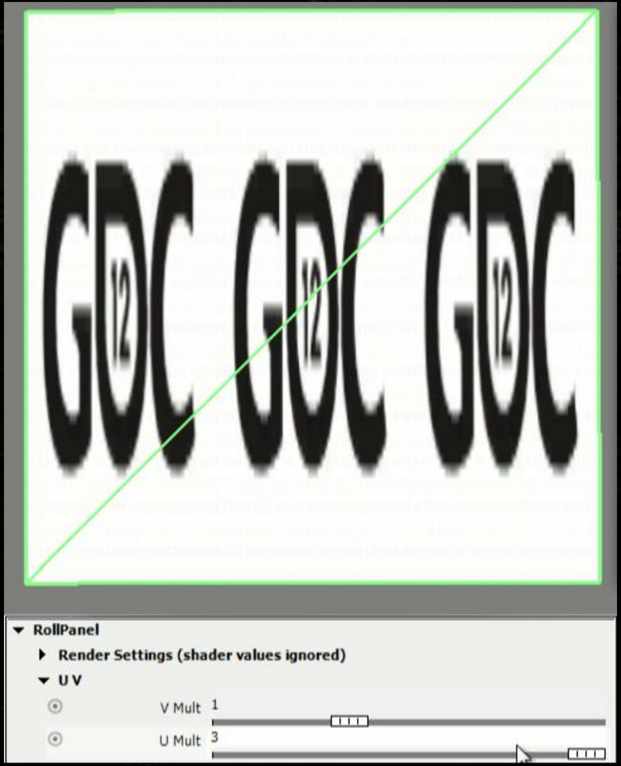
Prerequisite Knowledge: UV Math



Creating Motion in Particles

Prerequisite Knowledge: UV Math

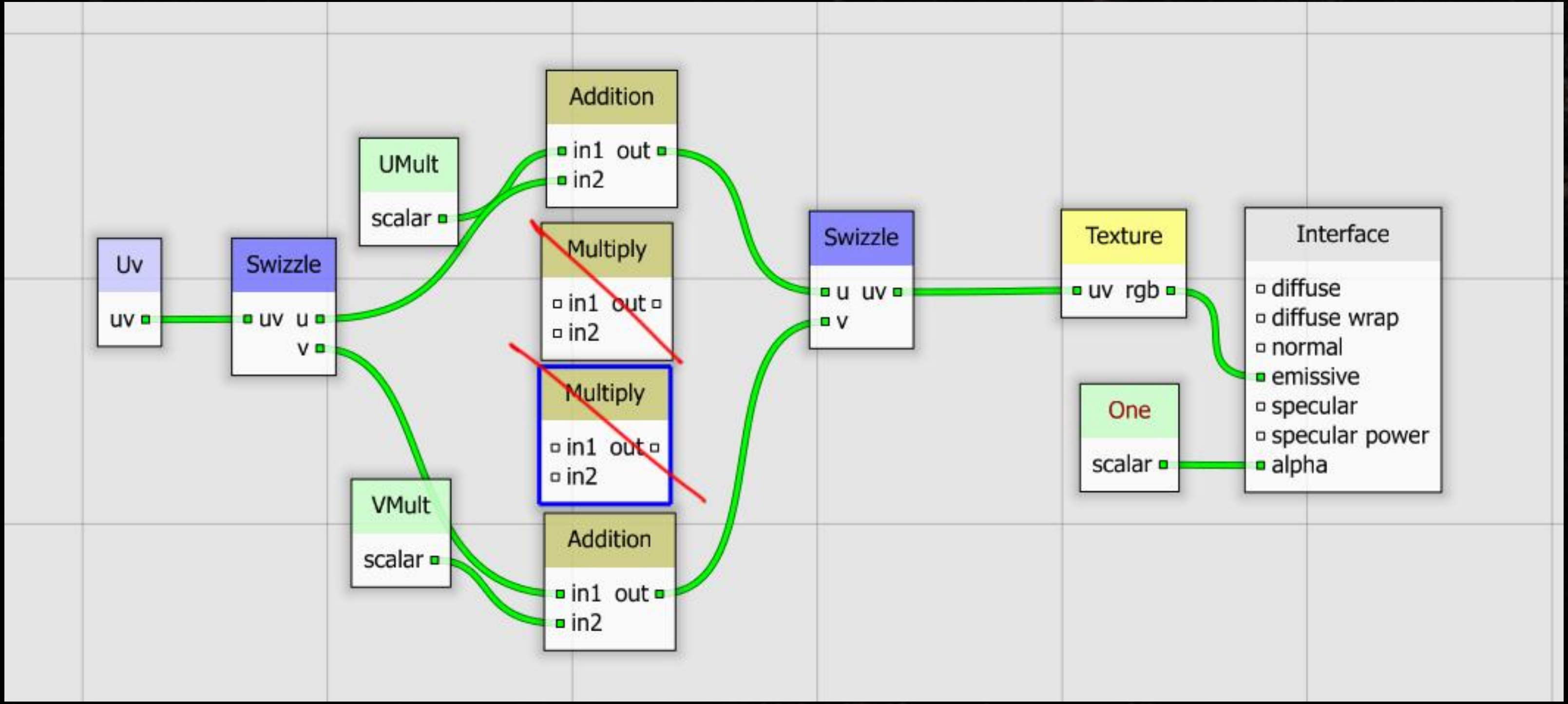




RollPanel
Render Settings (shader values ignored)
UV
V Mult 1
U Mult 3

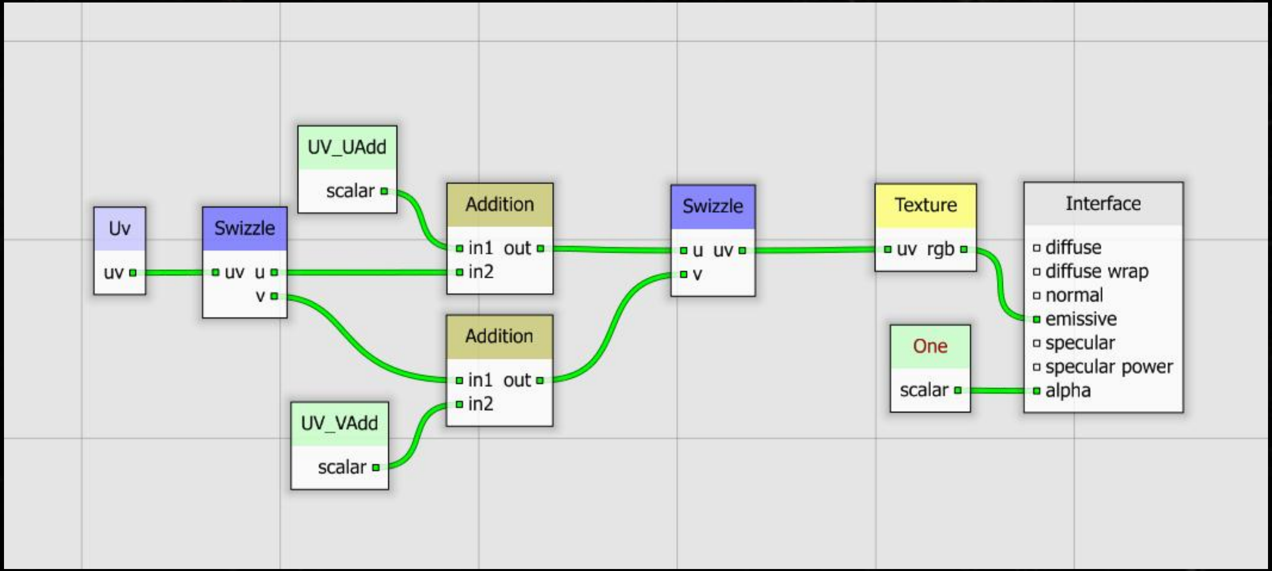
Creating Motion in Particles

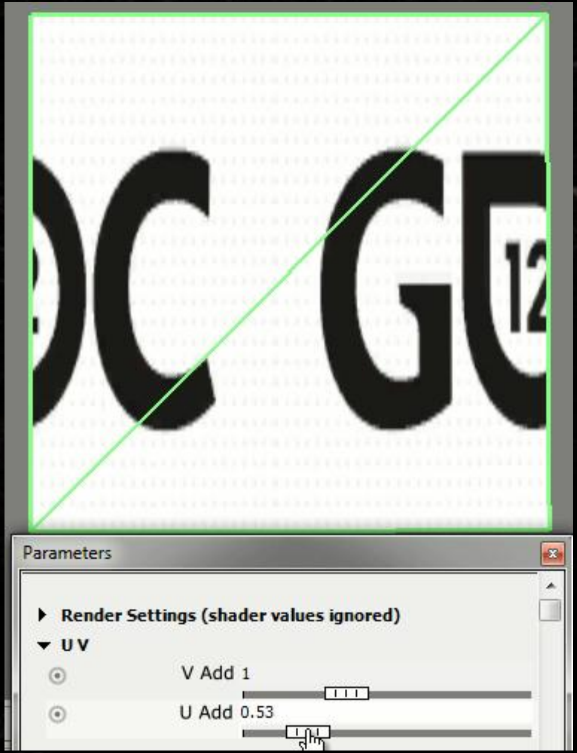
Prerequisite Knowledge: UV Math



Creating Motion in Particles

Prerequisite Knowledge: UV Math





Render Settings (shader values ignored)

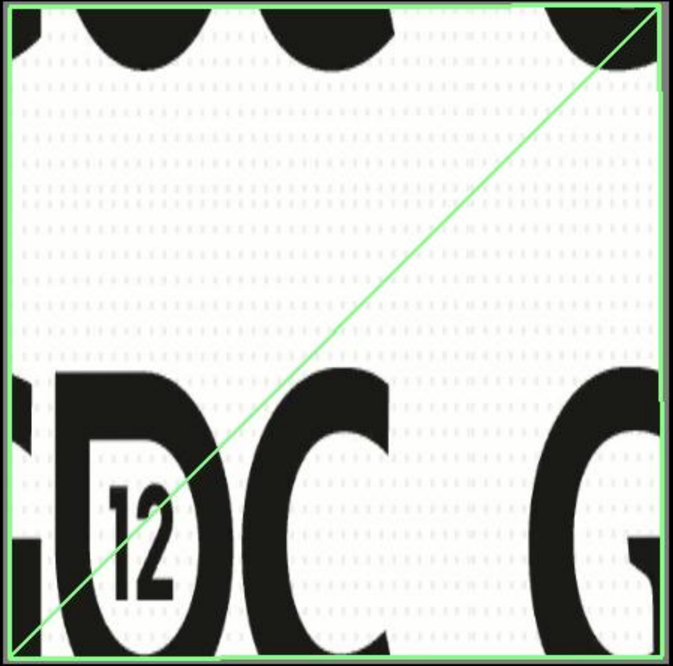
UV

V Add 1

U Add 0.53

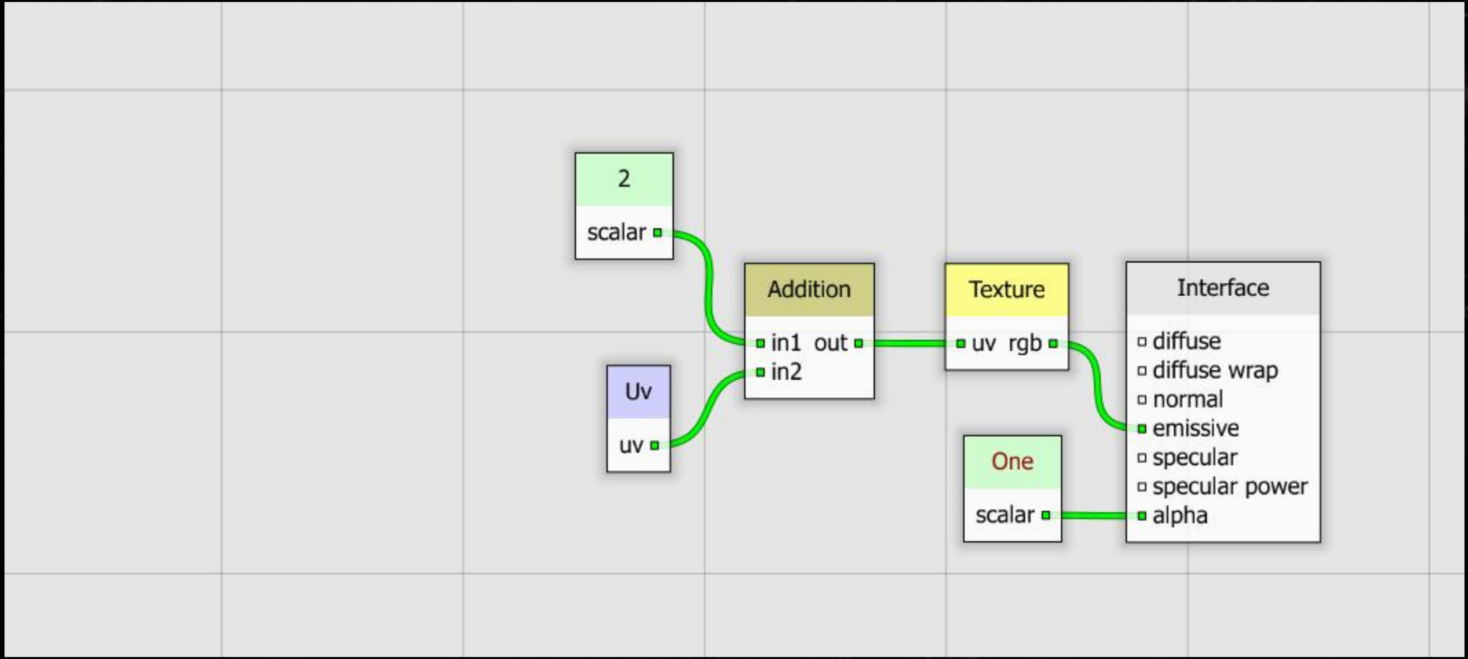
Creating Motion in Particles

Prerequisite Knowledge: UV Math



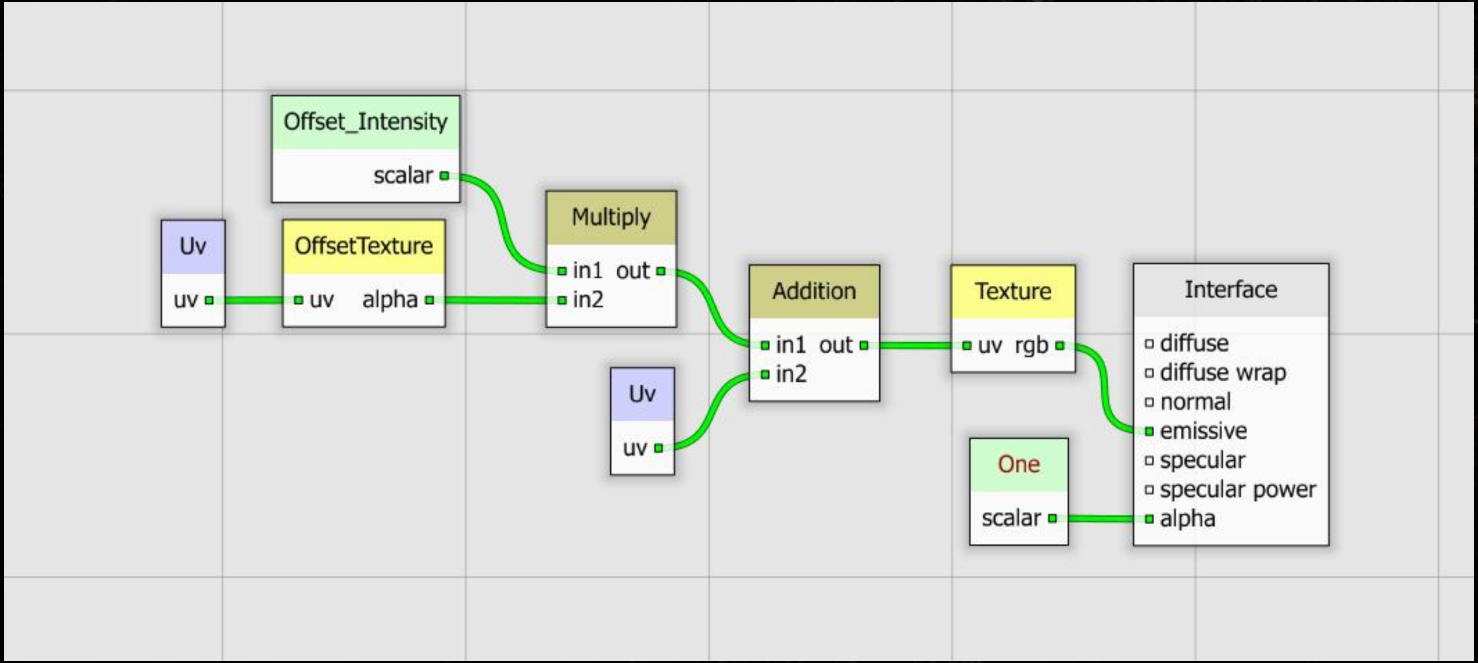
Creating Motion in Particles

Prerequisite Knowledge: UV Math



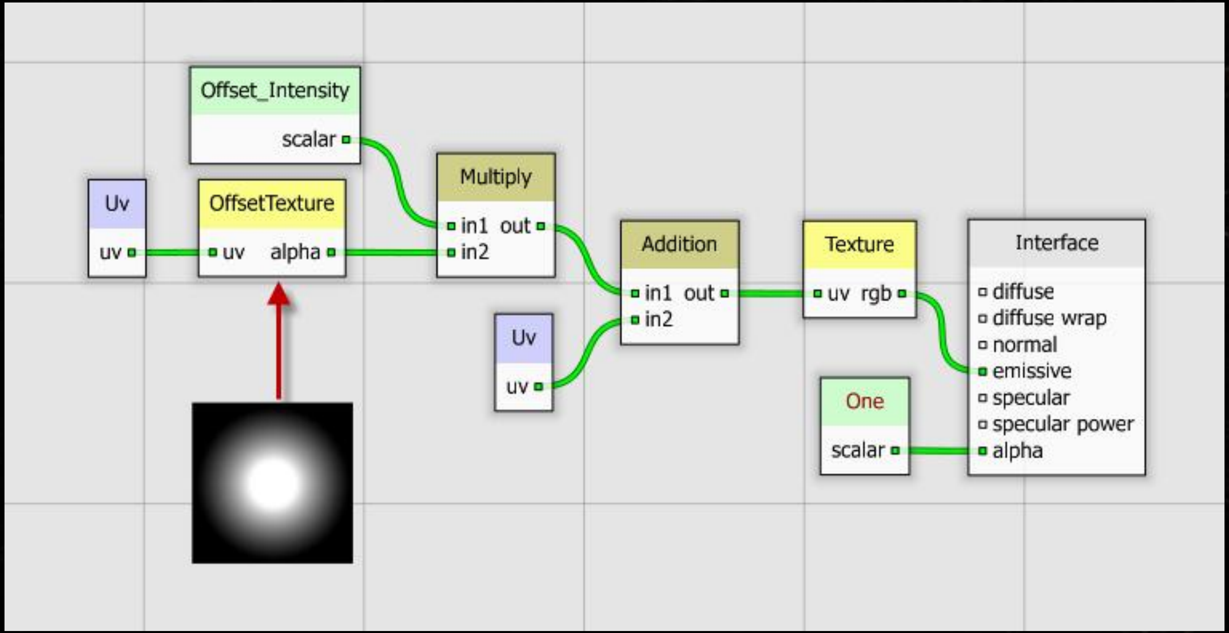
Creating Motion in Particles

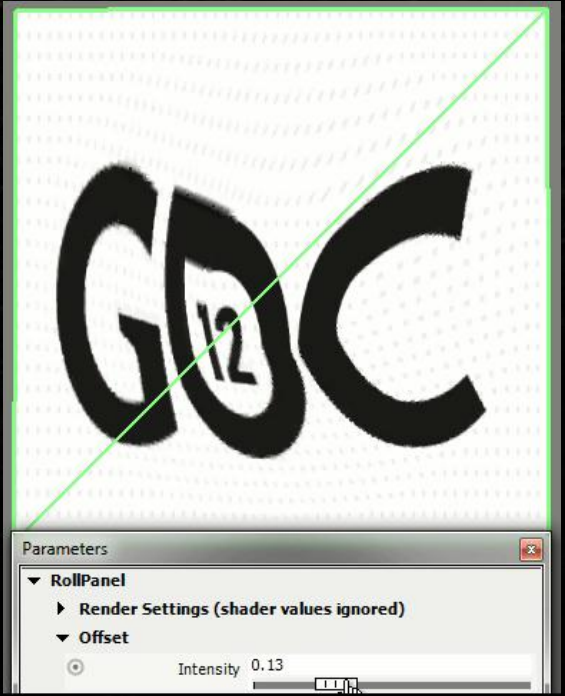
Prerequisite Knowledge: UV Math



Creating Motion in Particles

Prerequisite Knowledge: UV Math

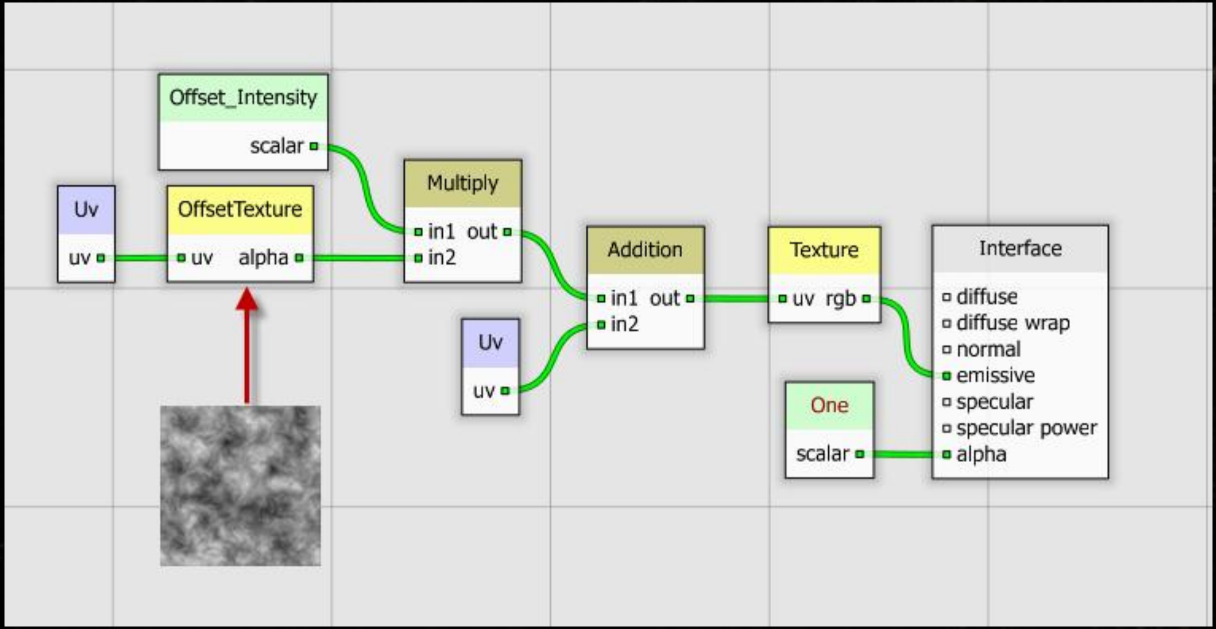


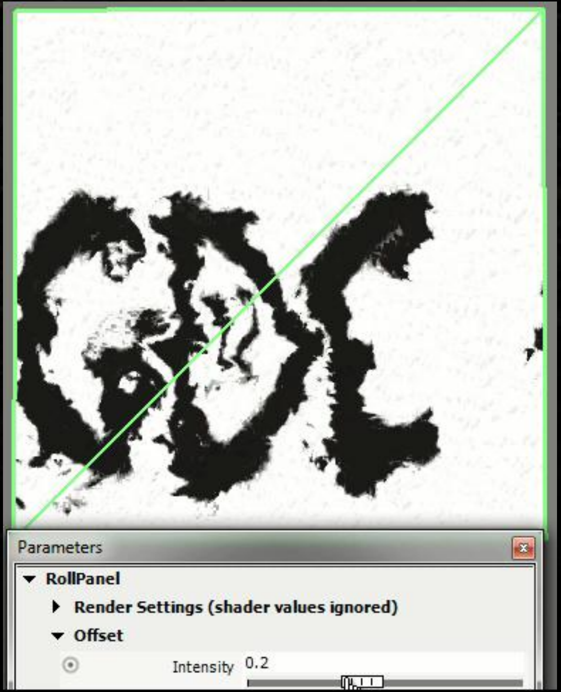


Intensity 0.13

Creating Motion in Particles

Prerequisite Knowledge: UV Math





Parameters

RollPanel

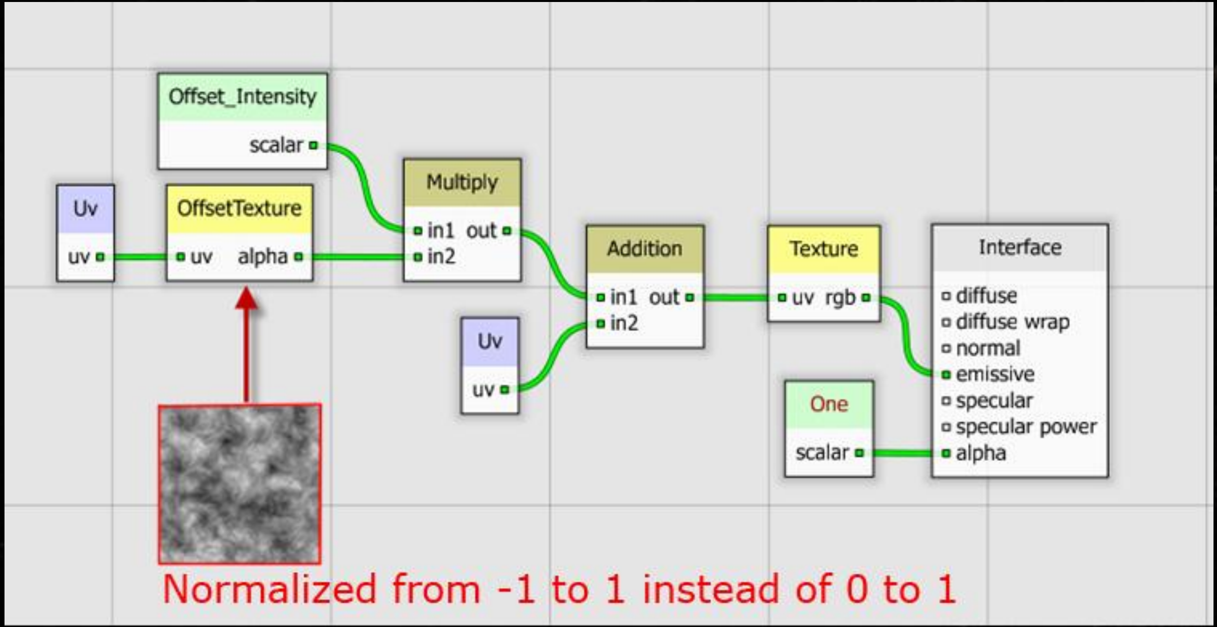
Render Settings (shader values ignored)

Offset

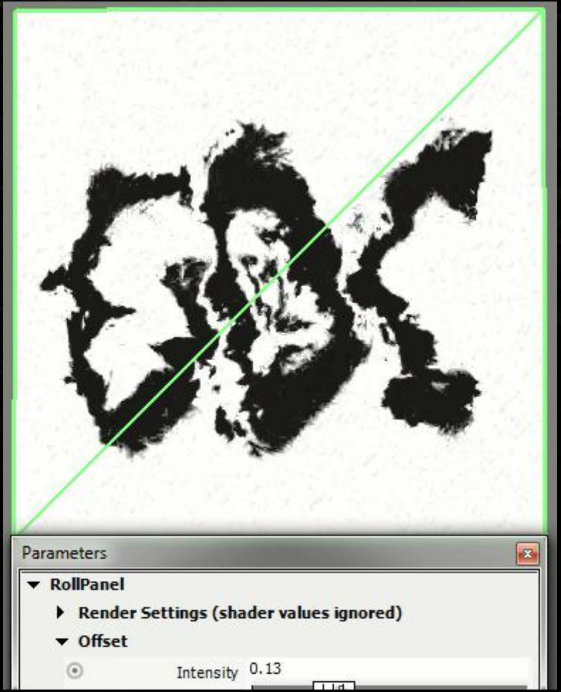
Intensity 0.2

Creating Motion in Particles

Prerequisite Knowledge: UV Math



Normalized from -1 to 1 instead of 0 to 1

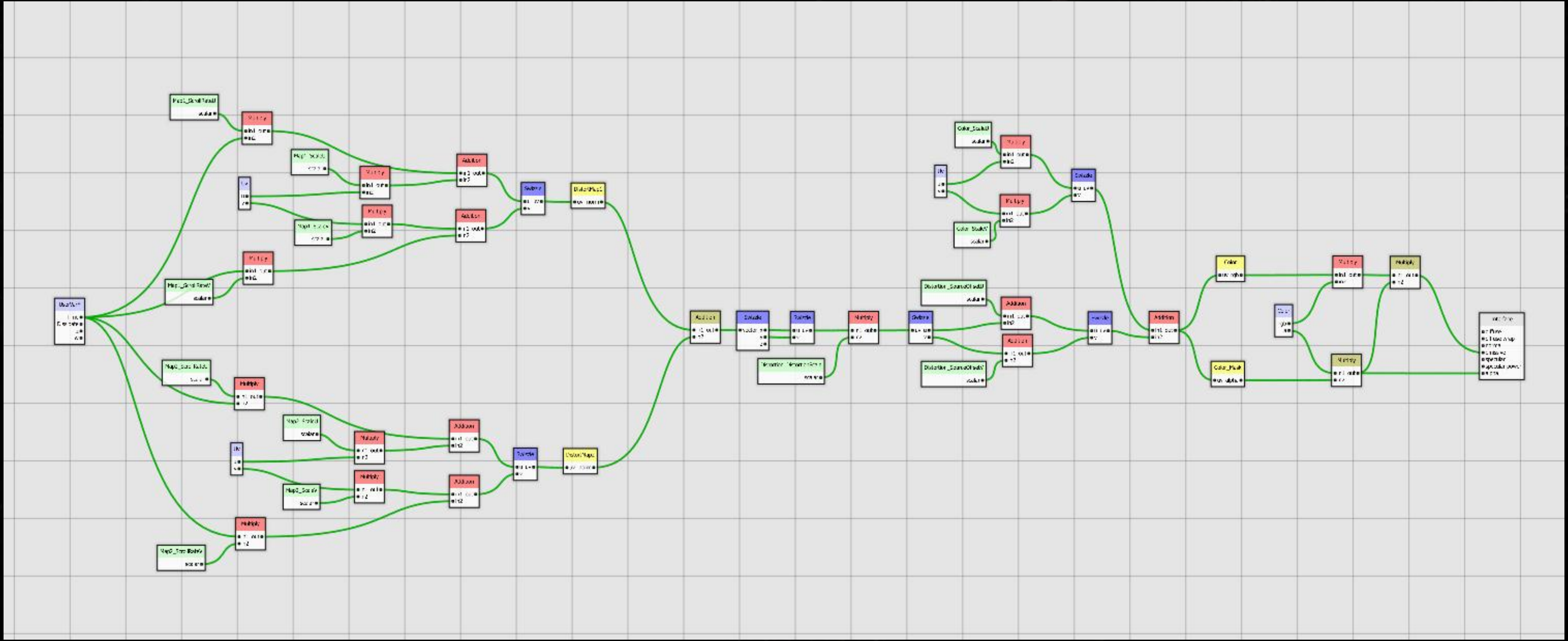

Parameters
RollPanel
Render Settings (shader values ignored)
Offset
Intensity 0.13

Creating Motion in Particles

Prerequisite Knowledge: UV Math

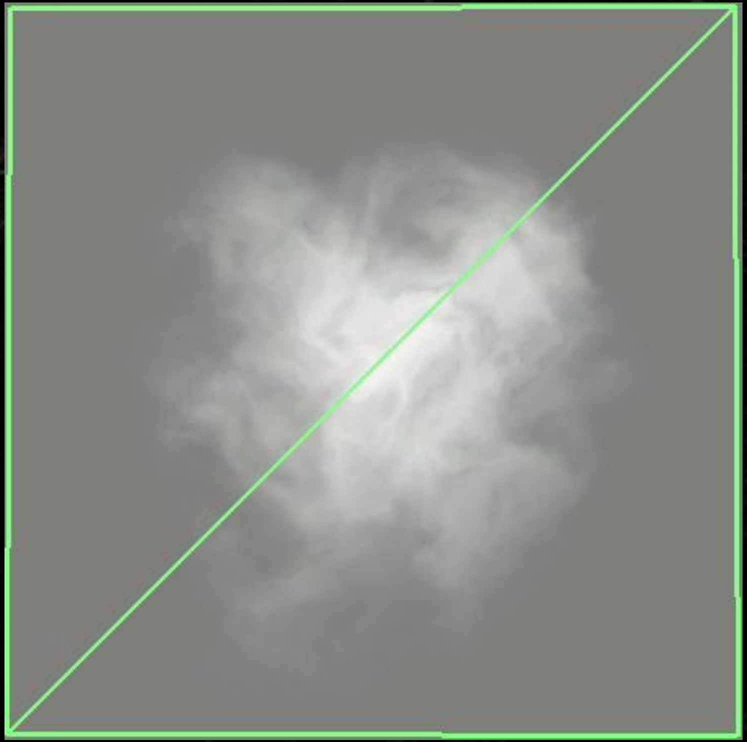
Creating Motion in Particles

Technique 1: Scrolling UV Distortion



Creating Motion in Particles

Technique 1: Scrolling UV Distortion



Creating Motion in Particles

Technique 1: Scrolling UV Distortion

Creating Motion in Particles

Technique 1: Scrolling UV Distortion

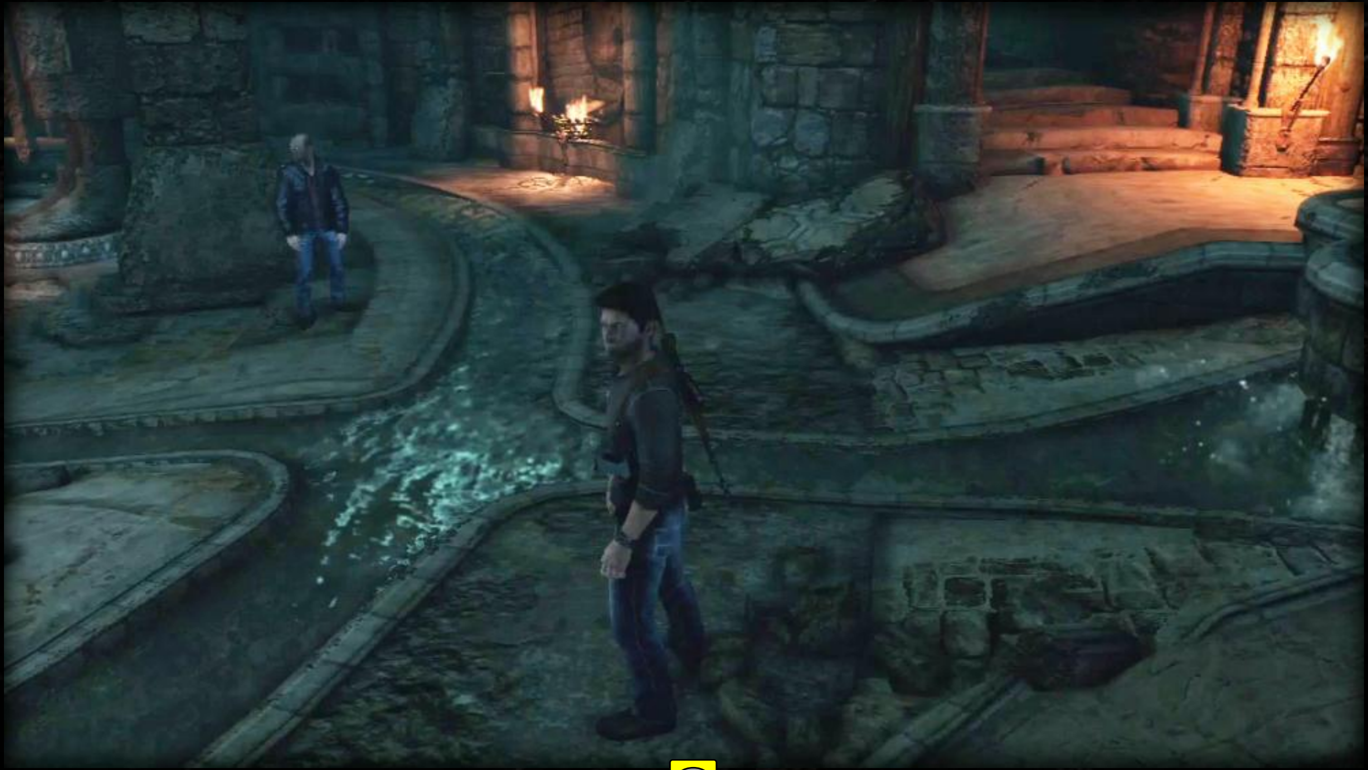
Creating Motion in Particles

Technique 1: Scrolling UV Distortion

- Pros
  - Breaks the silhouette
  - Adds internal motion
- Cons
  - It’s mostly non-directional motion and ambiguous detail

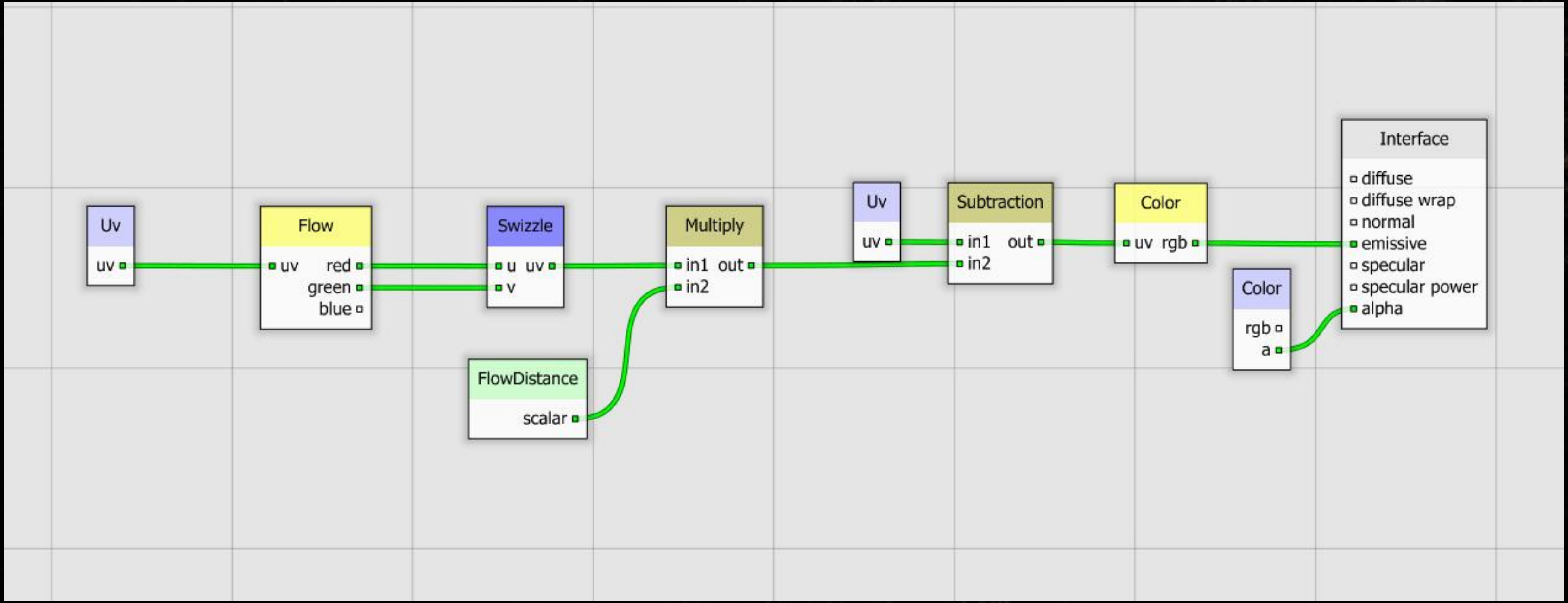
Creating Motion in Particles

Technique 2: Flow Technique



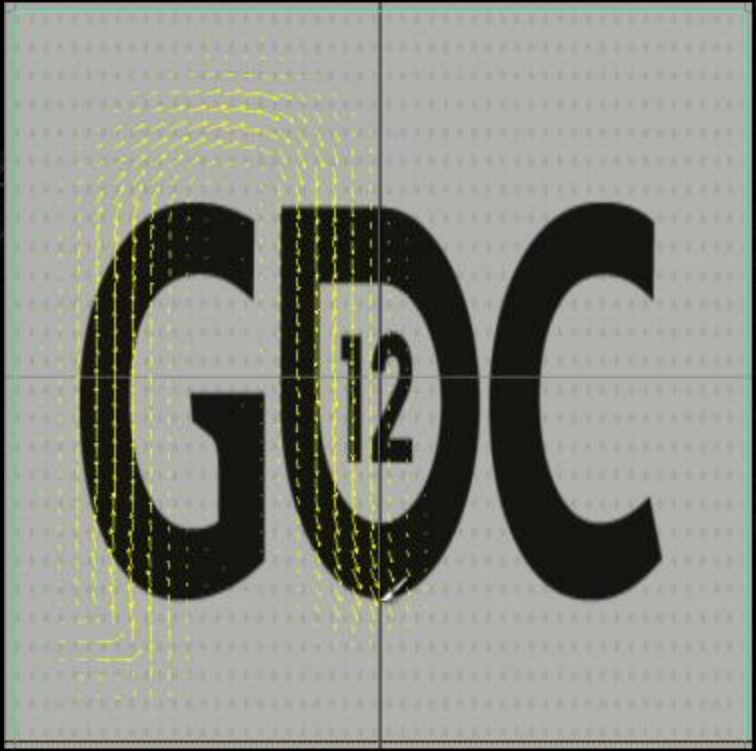
Creating Motion in Particles

Technique 2: Flow Technique



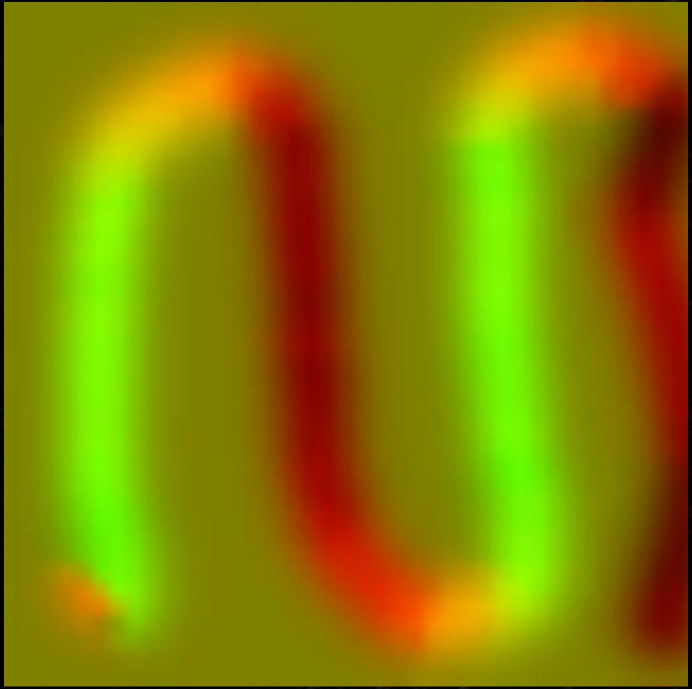
Creating Motion in Particles

Technique 2: Flow Technique



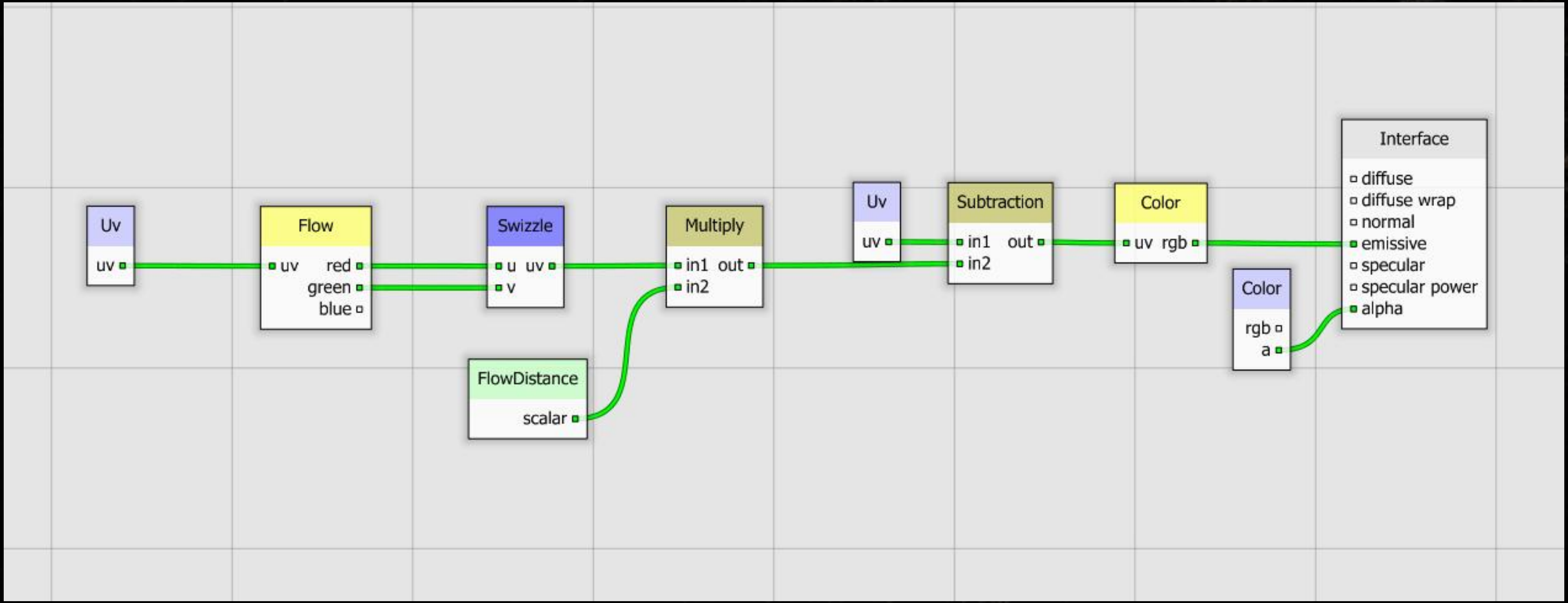
Creating Motion in Particles

Technique 2: Flow Technique



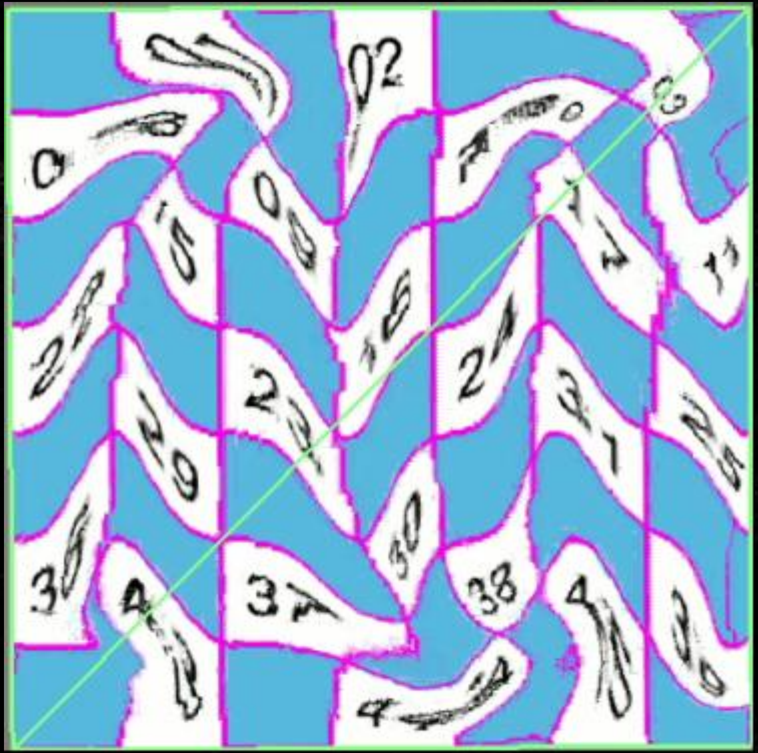
Creating Motion in Particles

Technique 2: Flow Technique



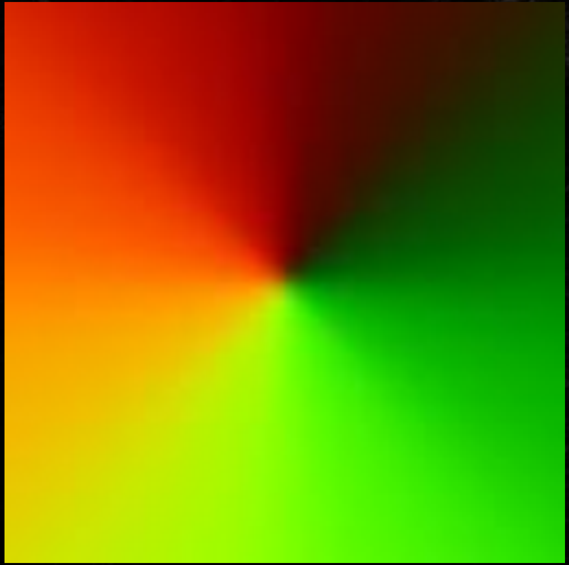
Creating Motion in Particles

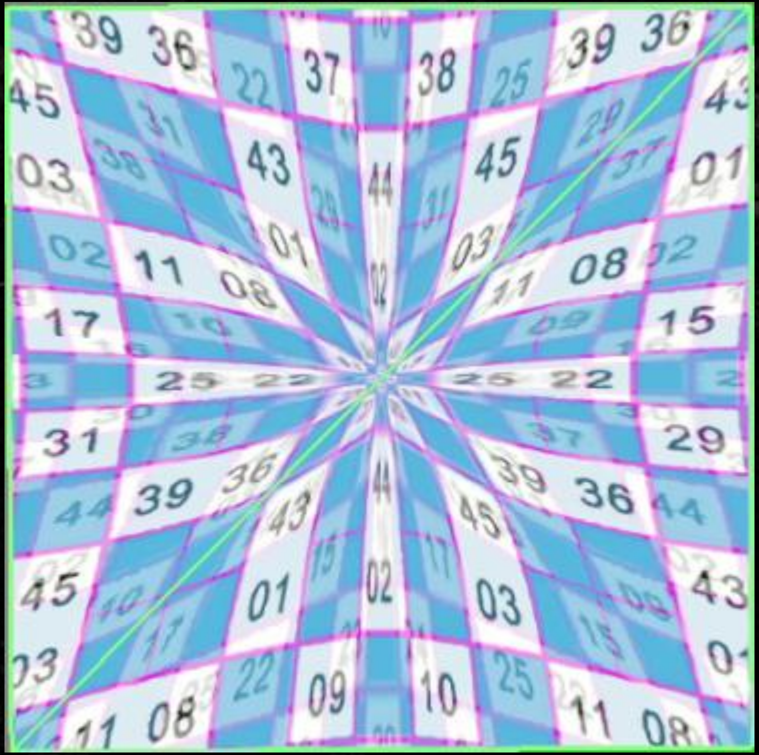
Technique 2: Flow Technique



Creating Motion in Particles

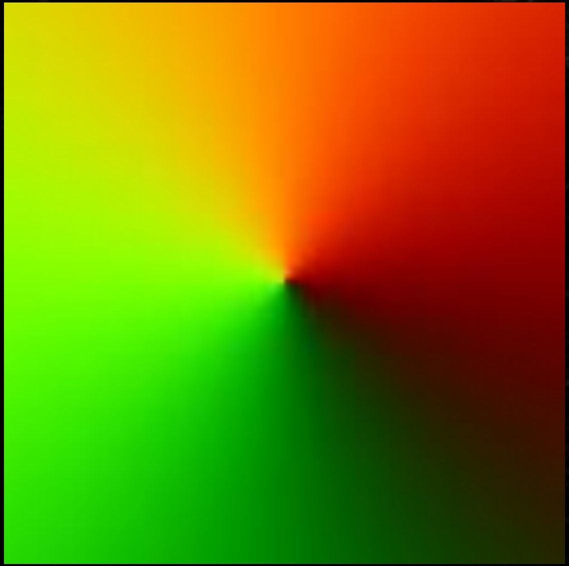
Technique 2: Flow Technique

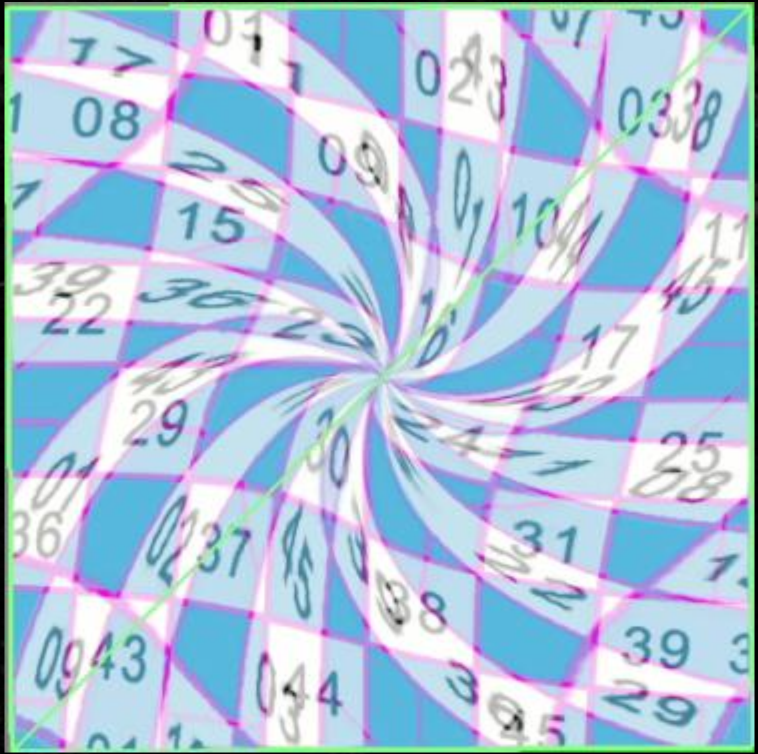




Creating Motion in Particles

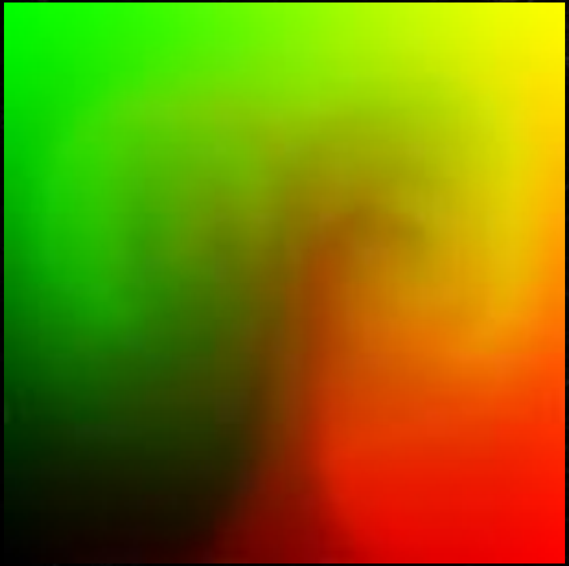
Technique 2: Flow Technique

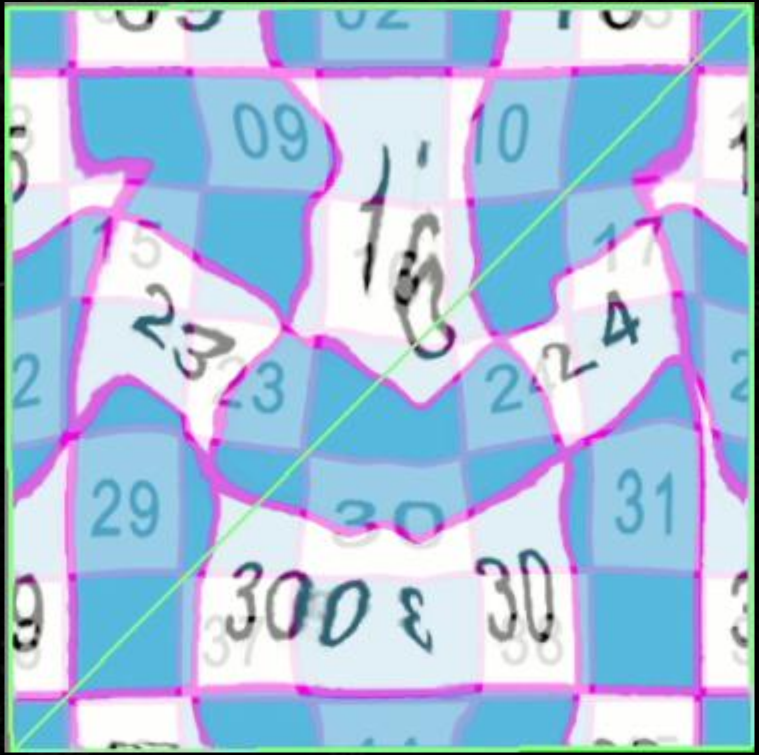




Creating Motion in Particles

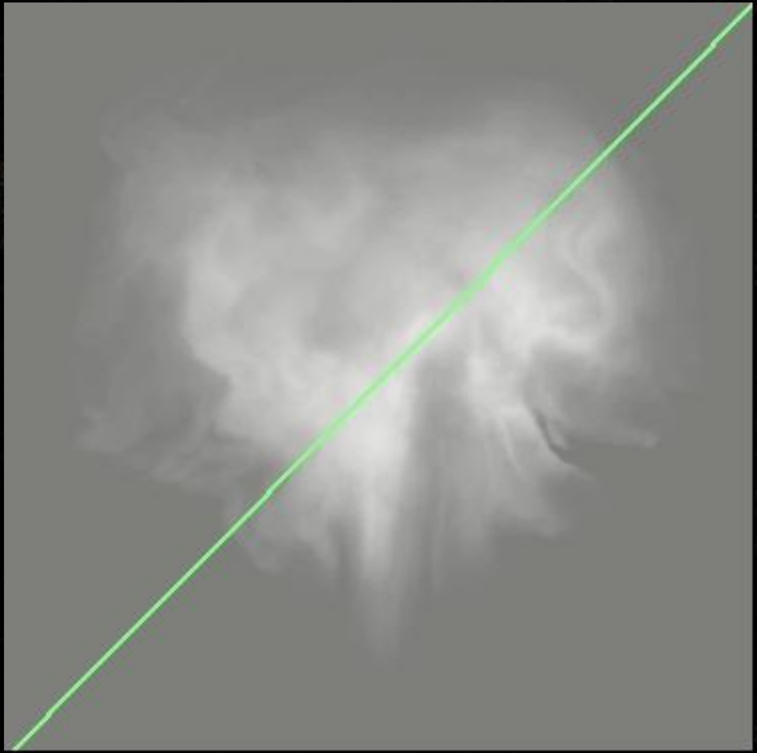
Technique 2: Flow Technique





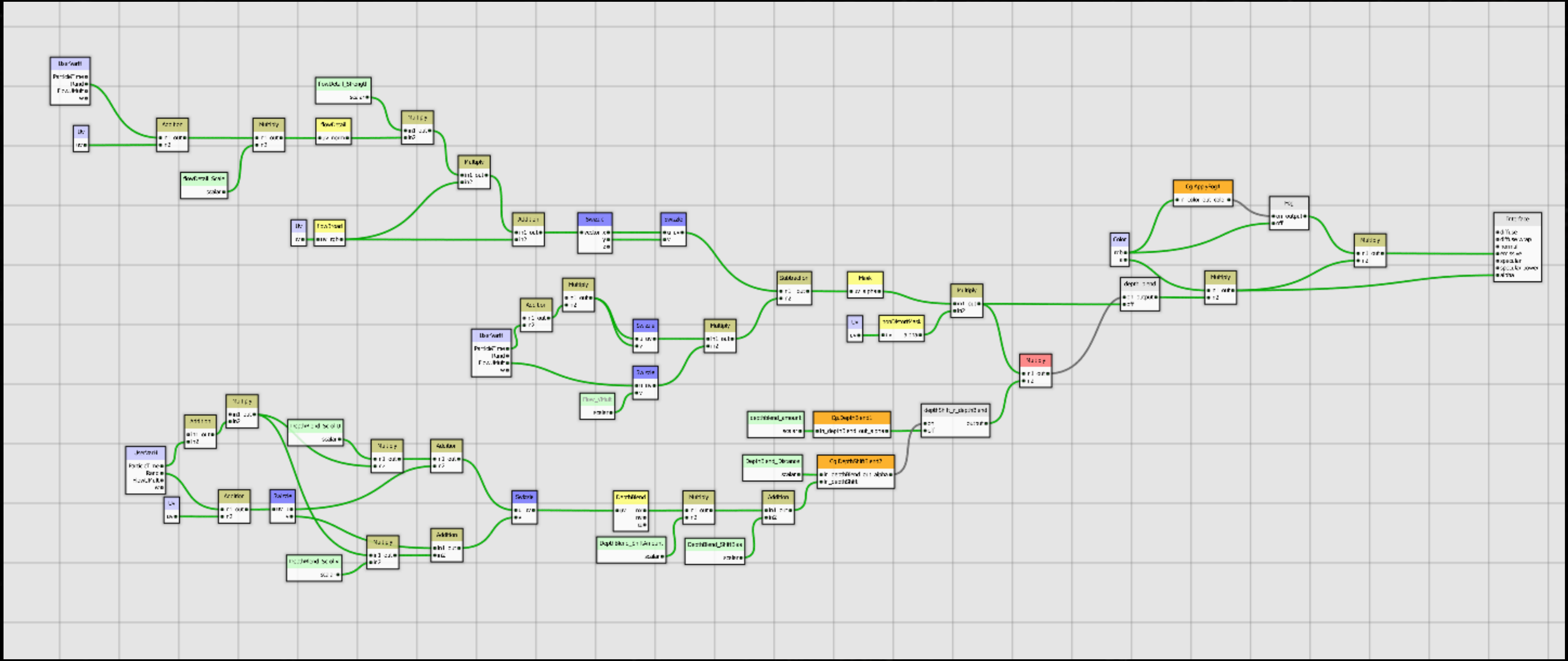
Creating Motion in Particles

Technique 2: Flow Technique



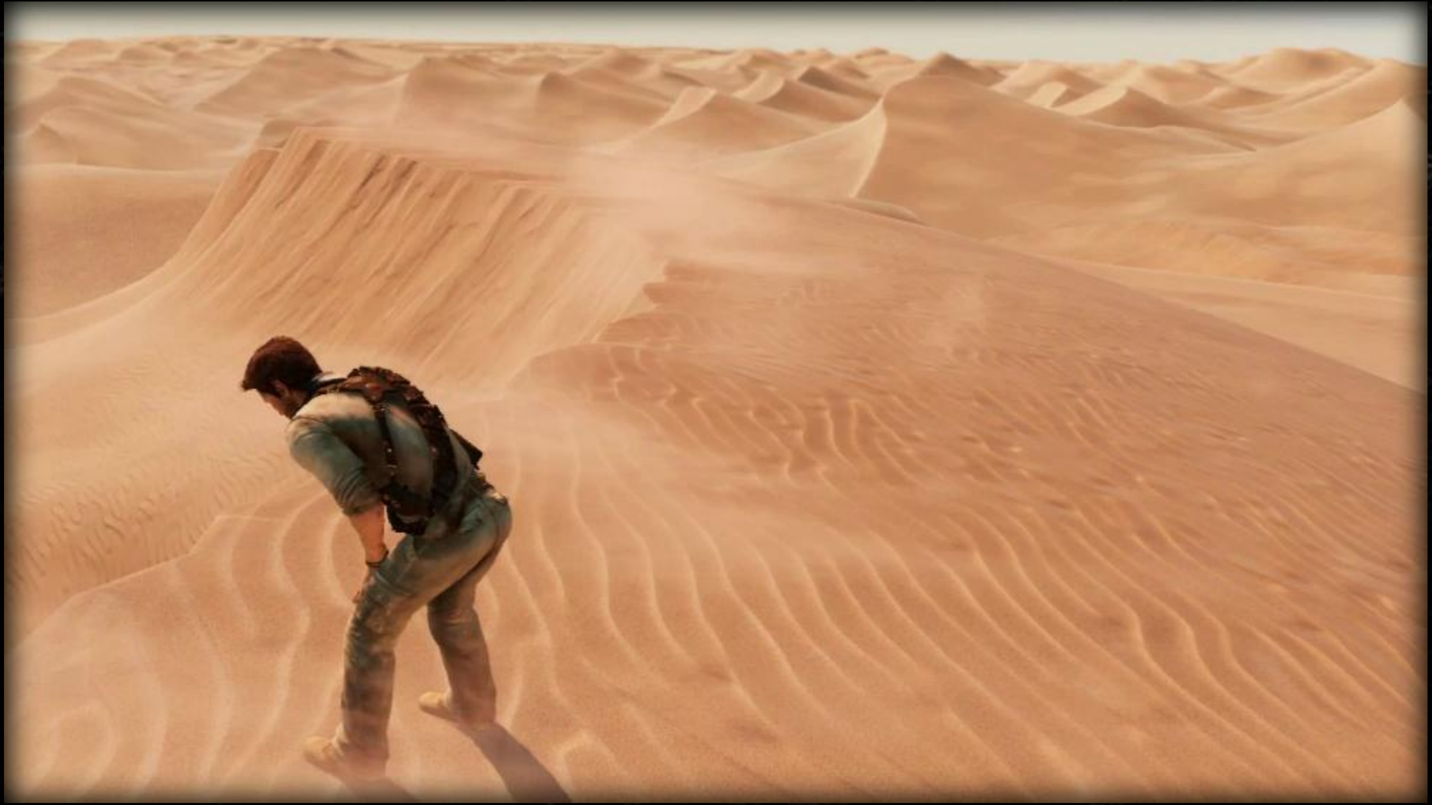
Creating Motion in Particles

Technique 2: Flow Technique



Creating Motion in Particles

Technique 2: Flow Technique



Creating Motion in Particles

Technique 2: Flow Technique

- Pros
  - Extremely controllable awesome motion
- Cons
  - Patterns of the motion are very visible (i.e. not very random...)
  - Lots of negative space in the particle (overdraw)

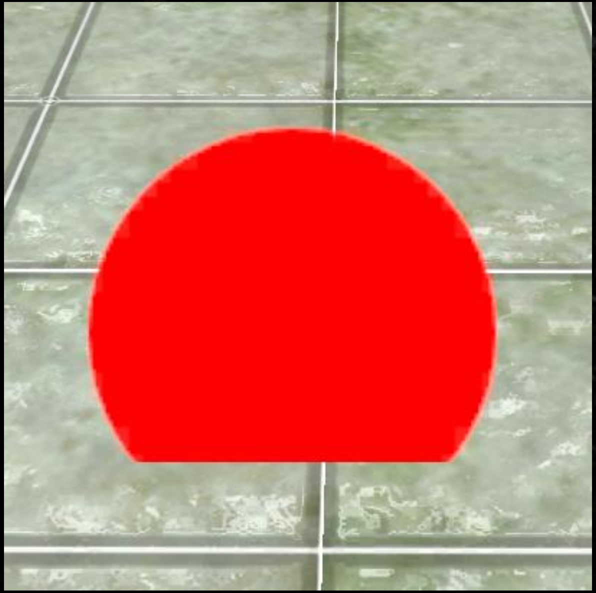
Burning Down a Chateau

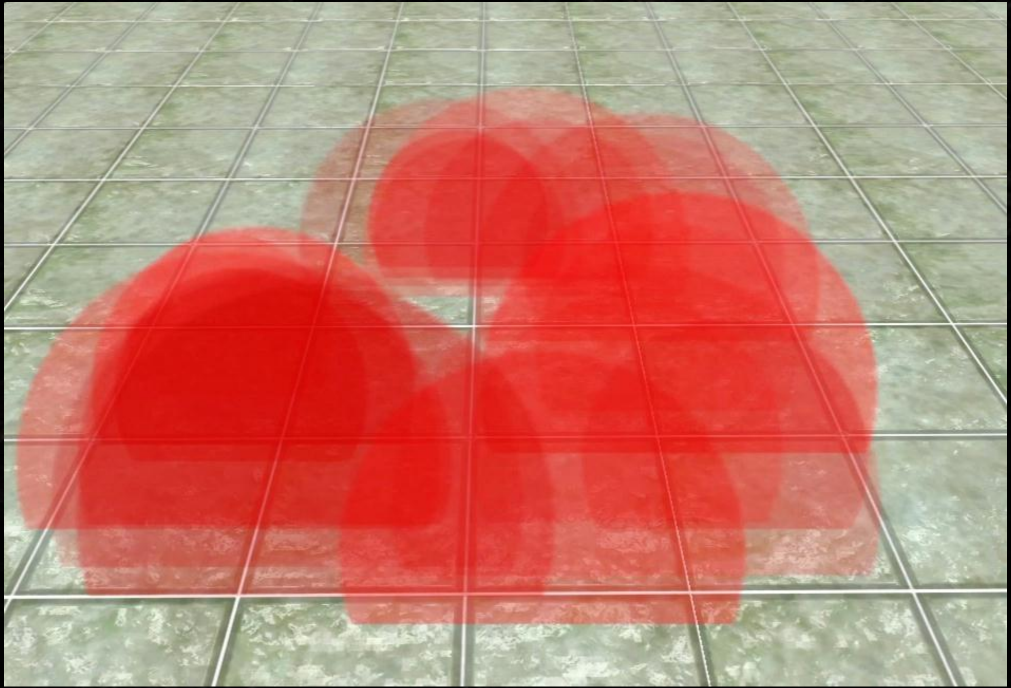
Challenge:

How do we make an awesome fiery inferno covering the walls, floor, & ceiling, while *running at 30 frames per second?*

Burning Down a Chateau

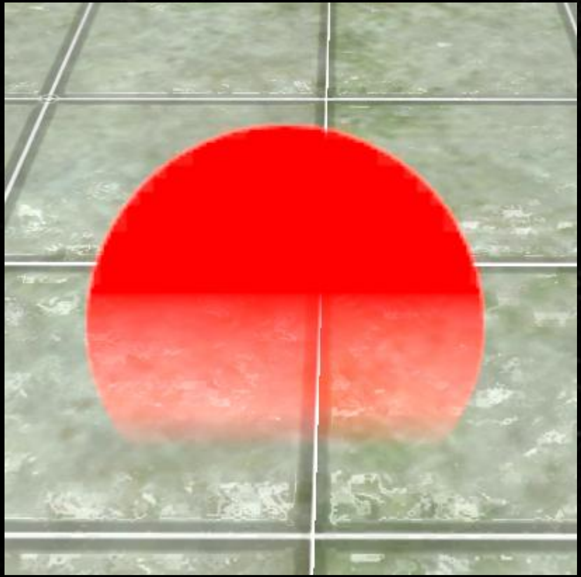
Prerequisite Knowledge: Z Depth Bias/Blending

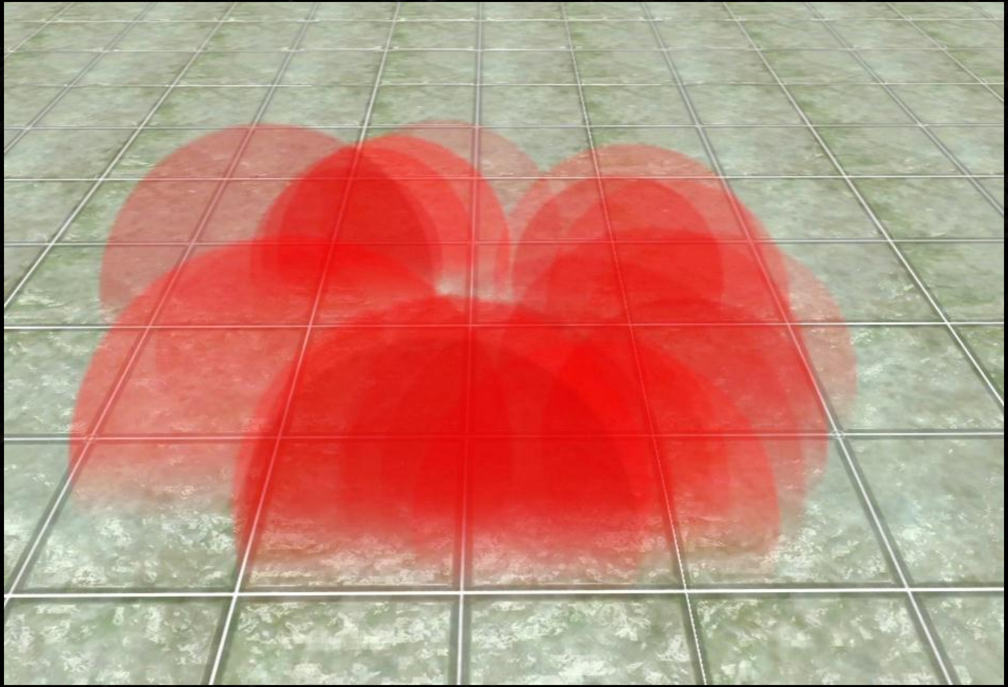




Burning Down a Chateau

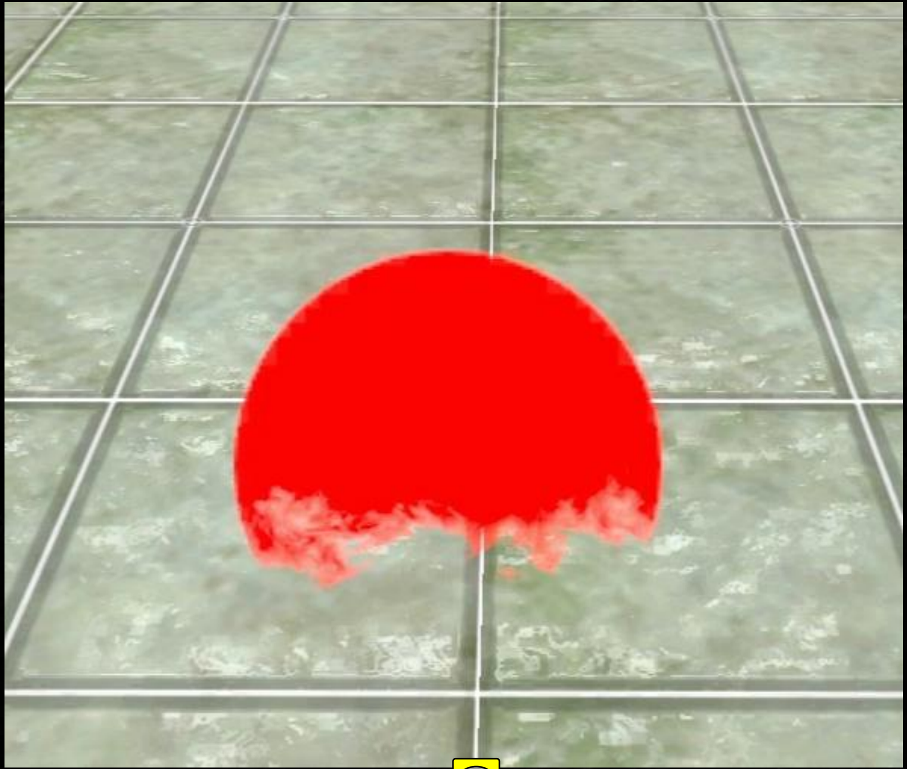
Prerequisite Knowledge: Z Depth Bias/Blending





Burning Down a Chateau

Prerequisite Knowledge: Z Depth Bias/Blending



Burning Down a Chateau

Burning Down a Chateau

Burning Down a Chateau

- Pros:
  - Less Particles = Less Overdraw = Better Frame Rate
- Cons:
  - All of the motion has to come from the material
  - Texture resolution is important/visible

Crashing a Cargo Plane

Challenge:

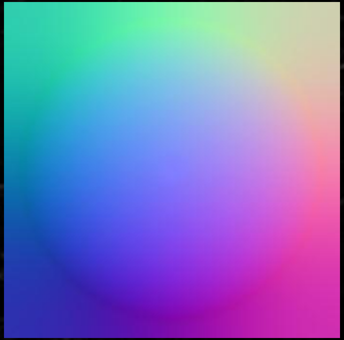
How do we make a realistic looking, *thick*, *volumetric* smokestack with enough *broad and subtle motion* to feel like its huge, but in the distance?

Crashing a Cargo Plane

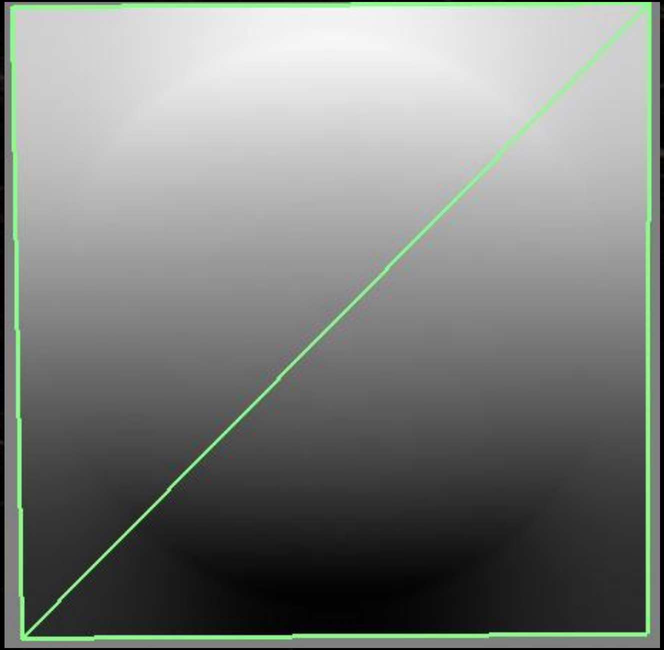
Prerequisite Knowledge: Dot Product Against a Normal Map

(0, 1, 0) ·

Light Vector Dot Product

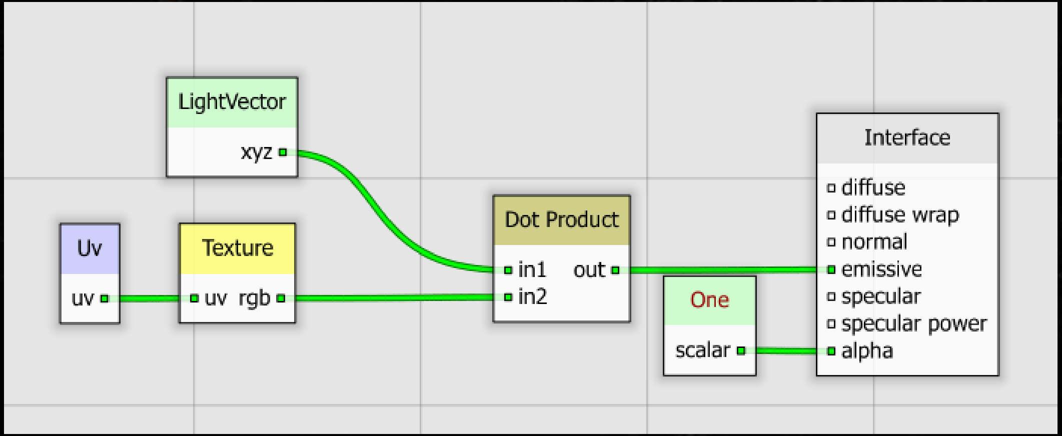


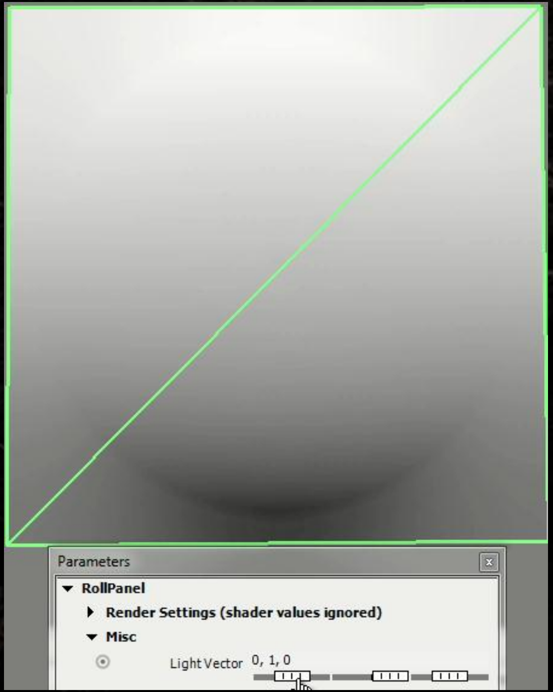
=



Crashing a Cargo Plane

Prerequisite Knowledge: Dot Product Against a Normal Map

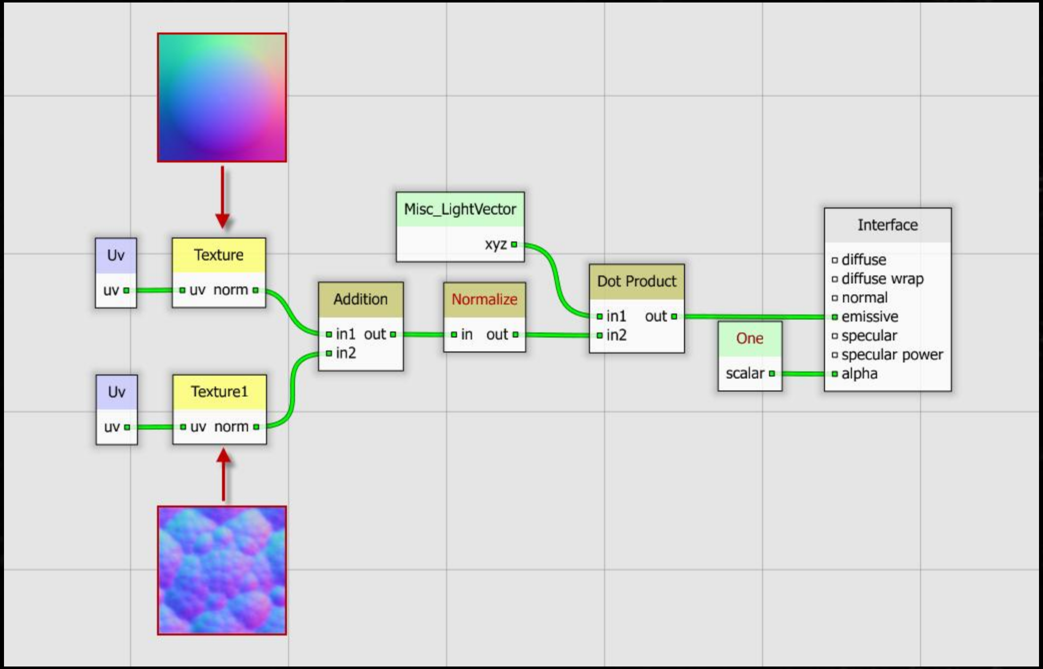


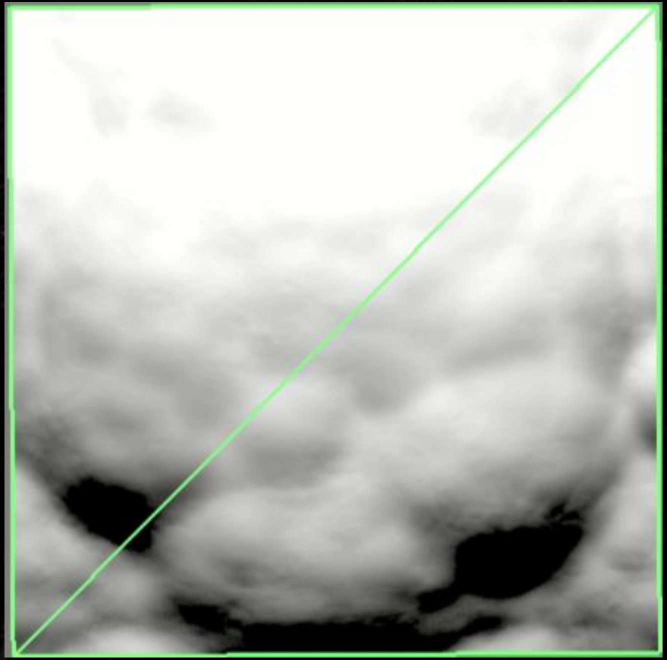


Light Vector 0, 1, 0

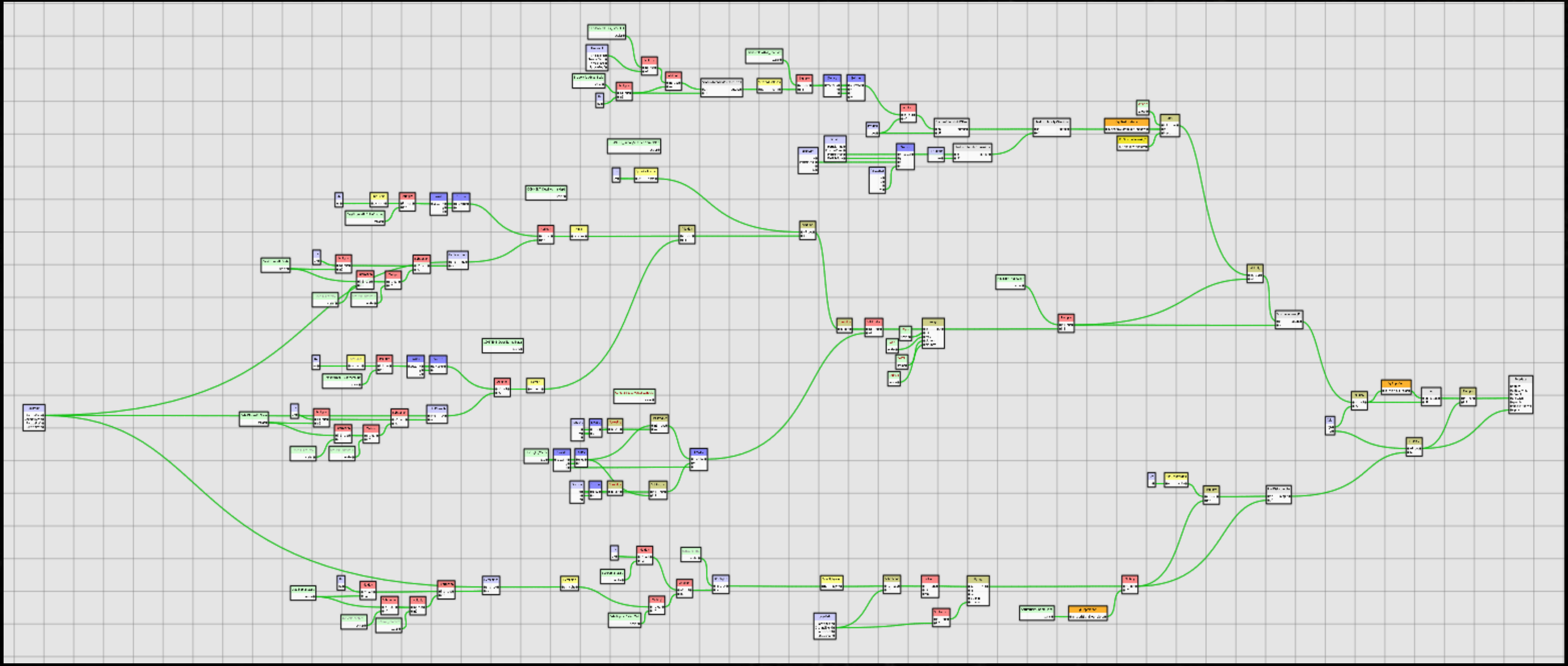
Crashing a Cargo Plane

Prerequisite Knowledge: Dot Product Against a Normal Map

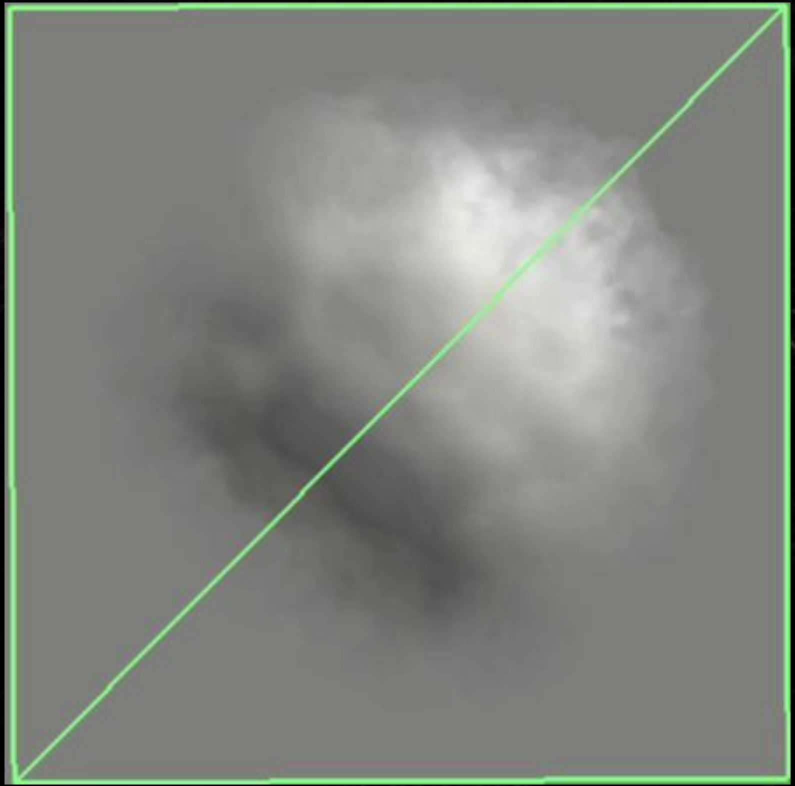




Crashing a Cargo Plane



Crashing a Cargo Plane



Crashing a Cargo Plane

Crashing a Cargo Plane

Crashing a Cargo Plane

- Pros:
  - Decent volumetric feel and motion
  - Tons of control over color, shape, motion, etc
- Cons:
  - Requires A LOT of particles
  - Very expensive shader

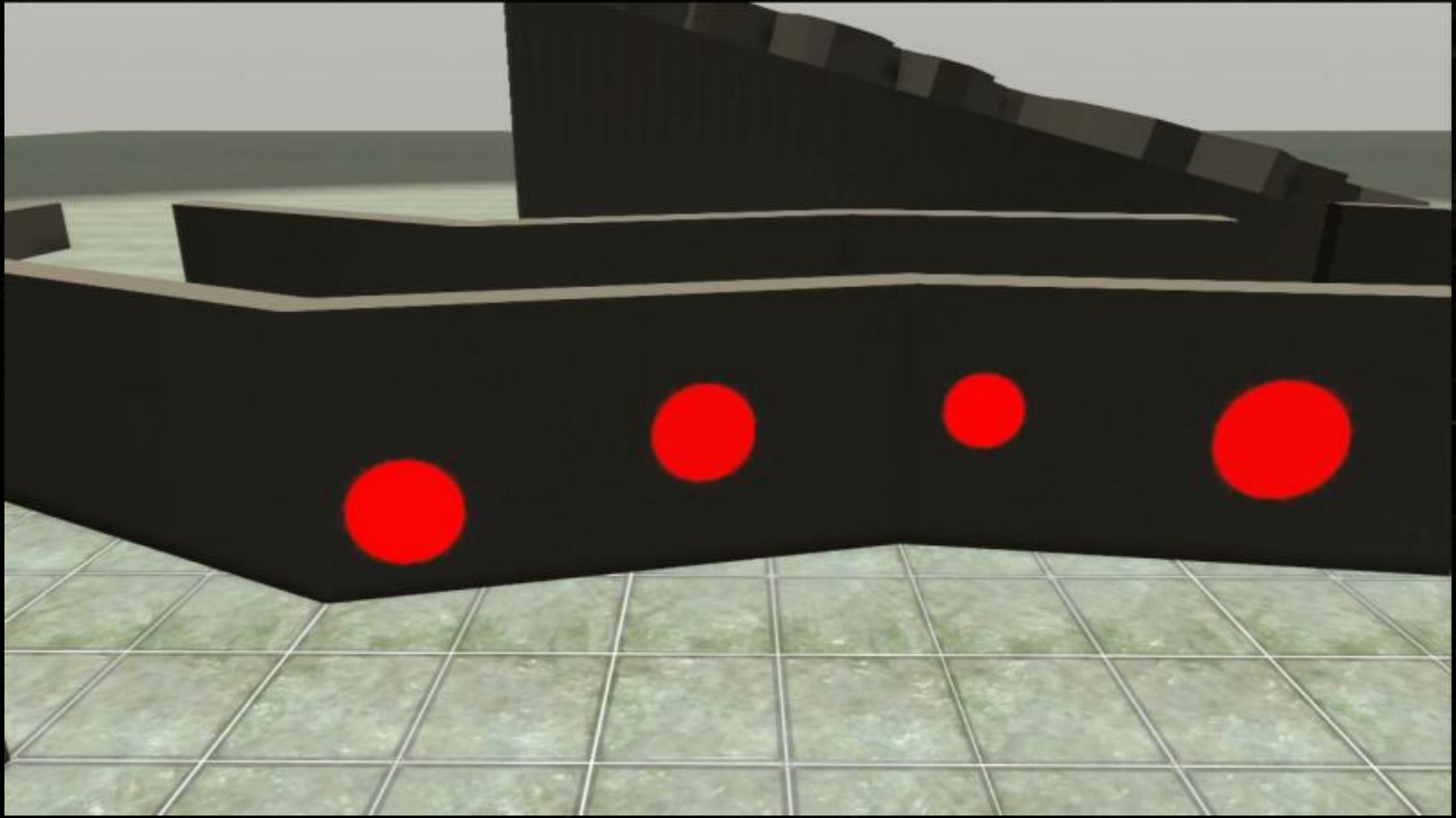
Interacting with the Desert

Interacting with the Desert

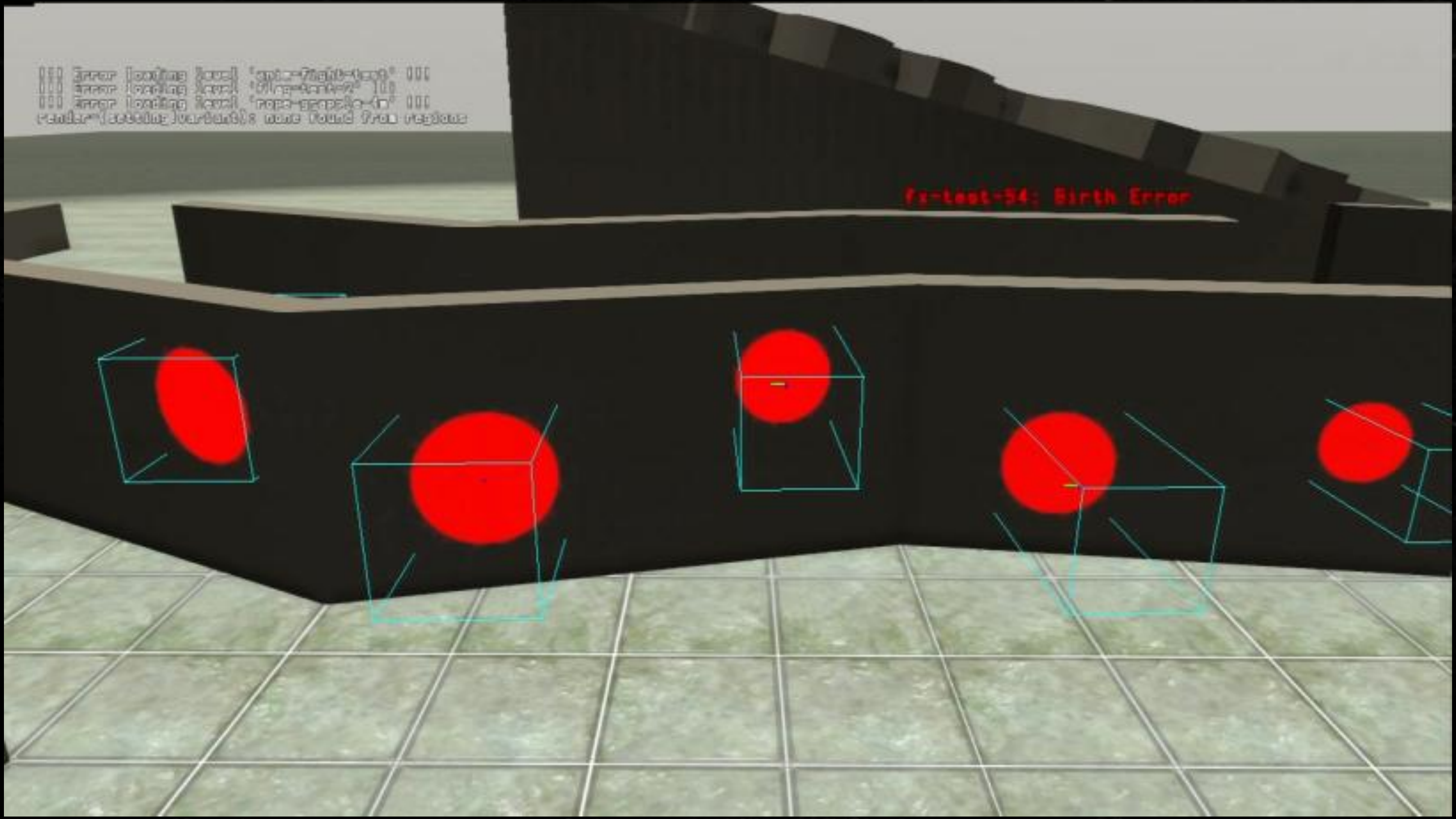
Challenge:

How do we make *fluid, realistic* sand interactions that consider the angle and direction of the sand dune?

Interacting with the Desert



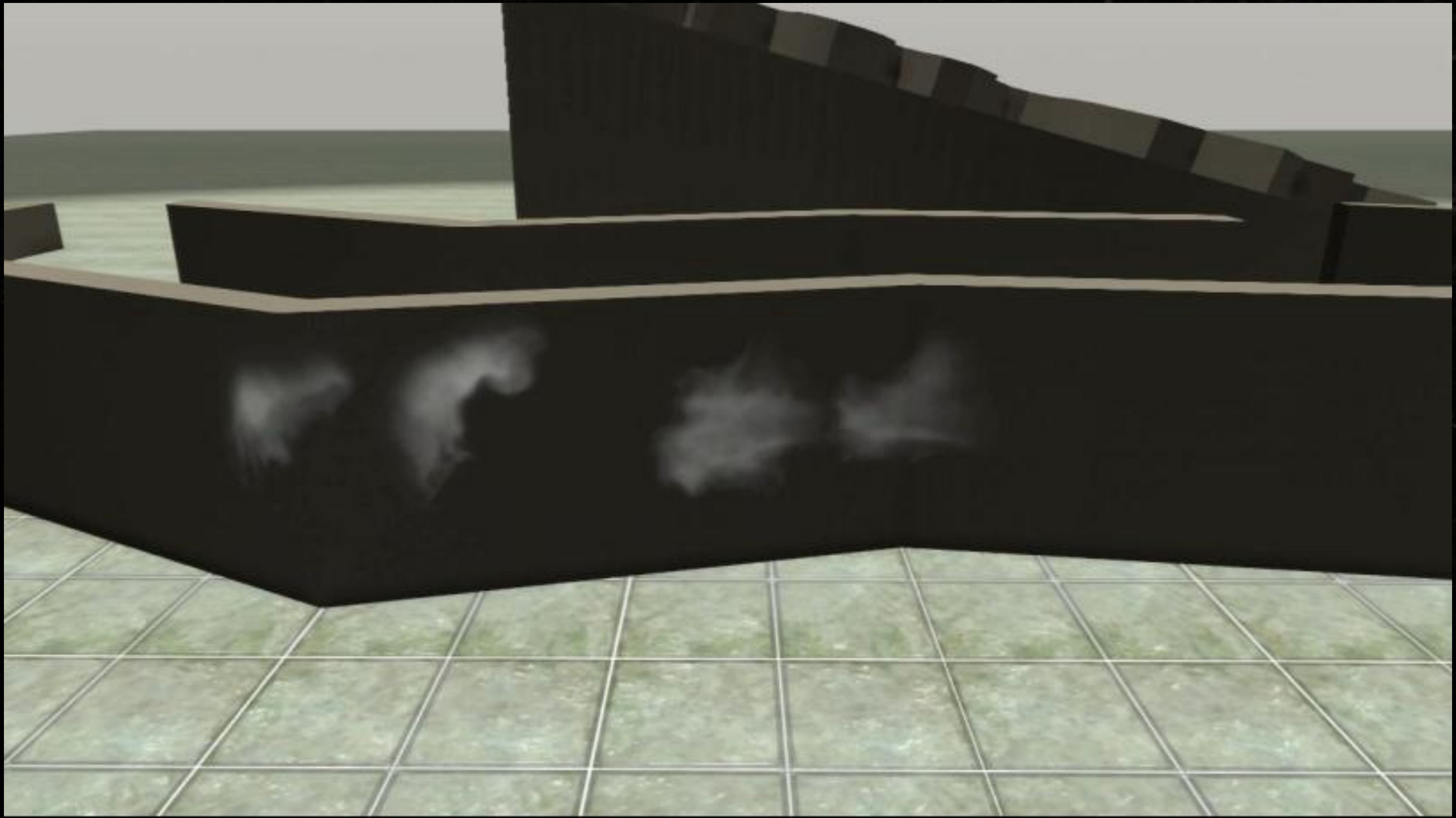
Interacting with the Desert



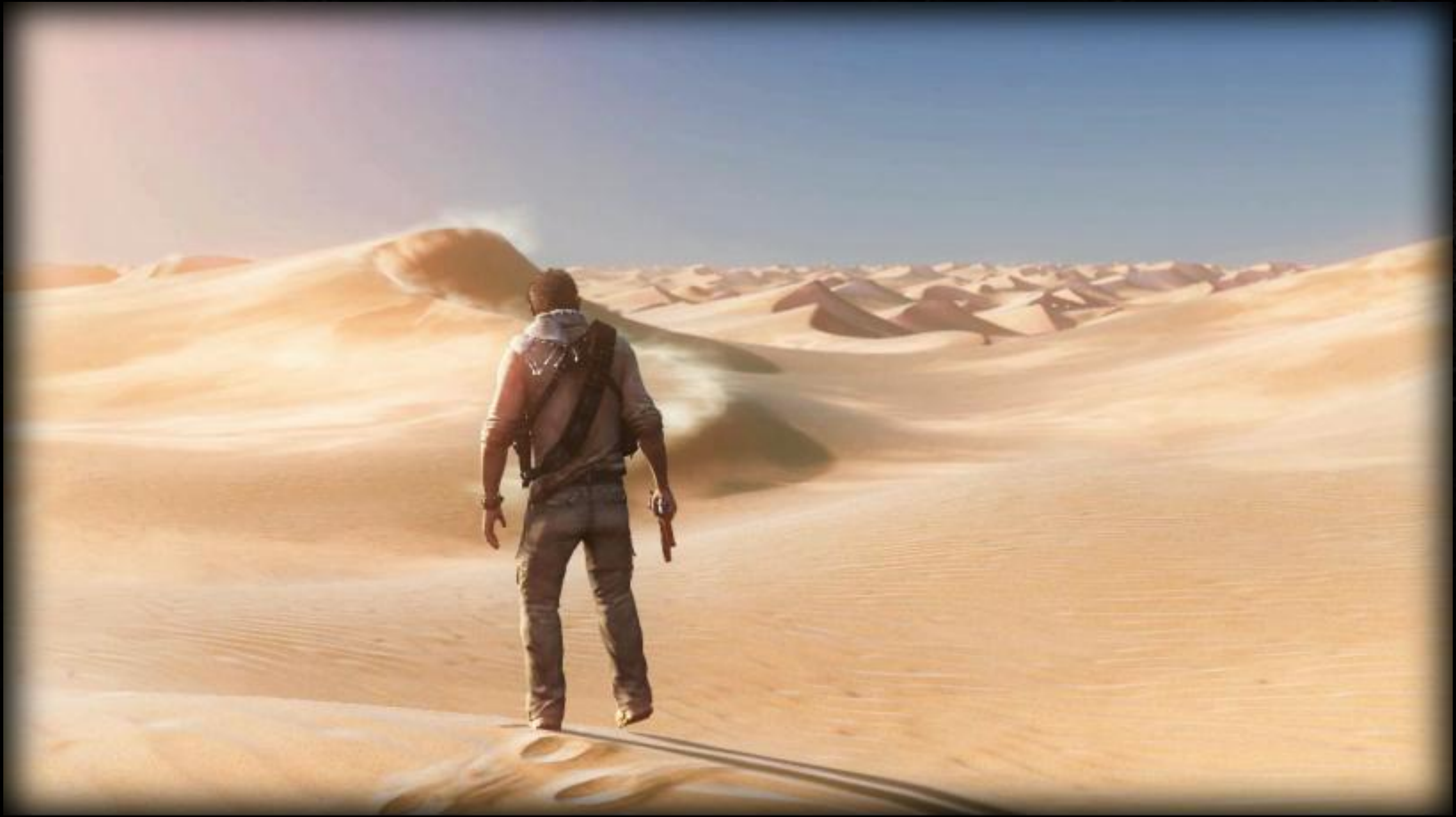
```
!!! Error loading level 'ambush-fight-test' !!!
!!! Error loading level 'flag-test-2' !!!
!!! Error loading level 'rope-grapple-fm' !!!
render-(setting)variant(): none found from regions
```

```
fx-test-34: Birth Error
```

Interacting with the Desert

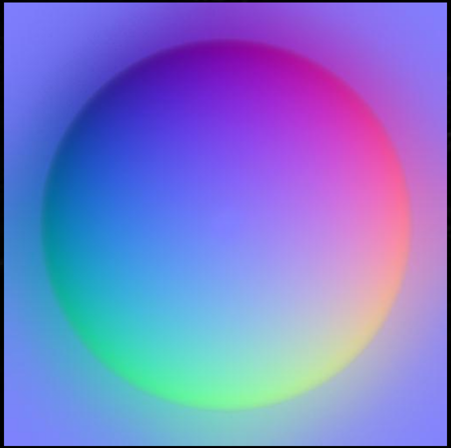


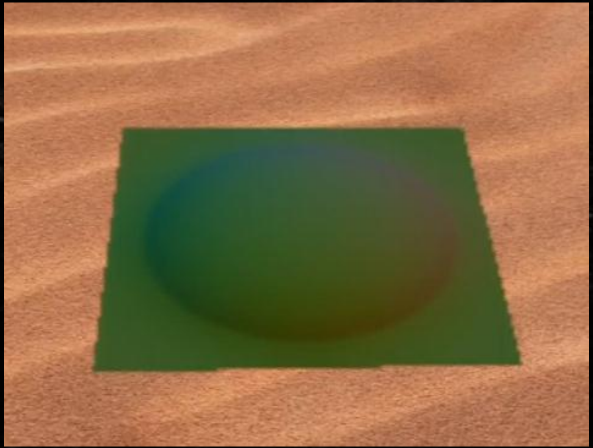
Interacting with the Desert

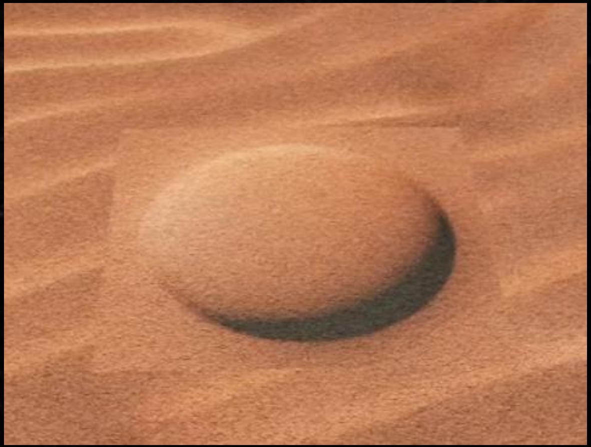


Interacting with the Desert

Projecting Particles into the World Normal Buffer


Particle Texture


Projected into
Emissive Render Pass

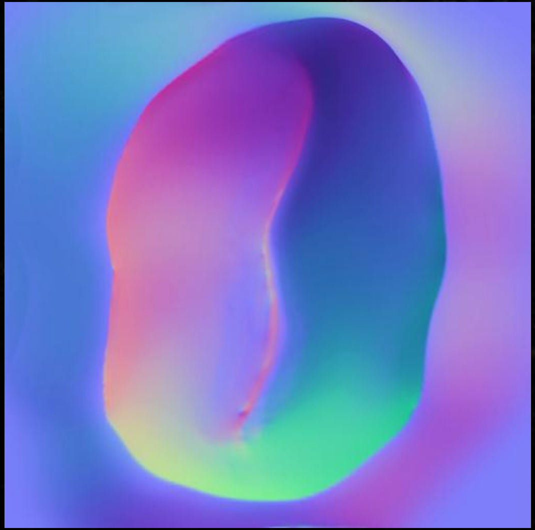

Projected into
World Normal Buffer

Interacting with the Desert

Sand Foot Prints

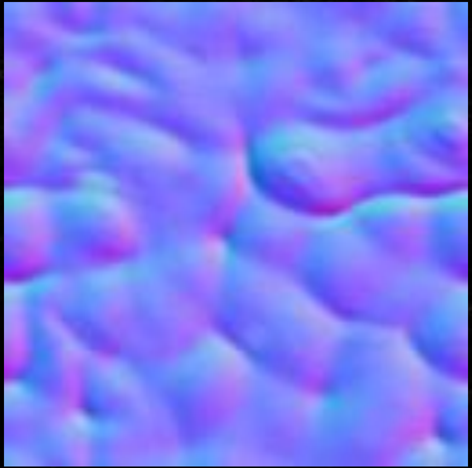
Interacting with the Desert

Sand Foot Prints



Interacting with the Desert

Sand Foot Prints



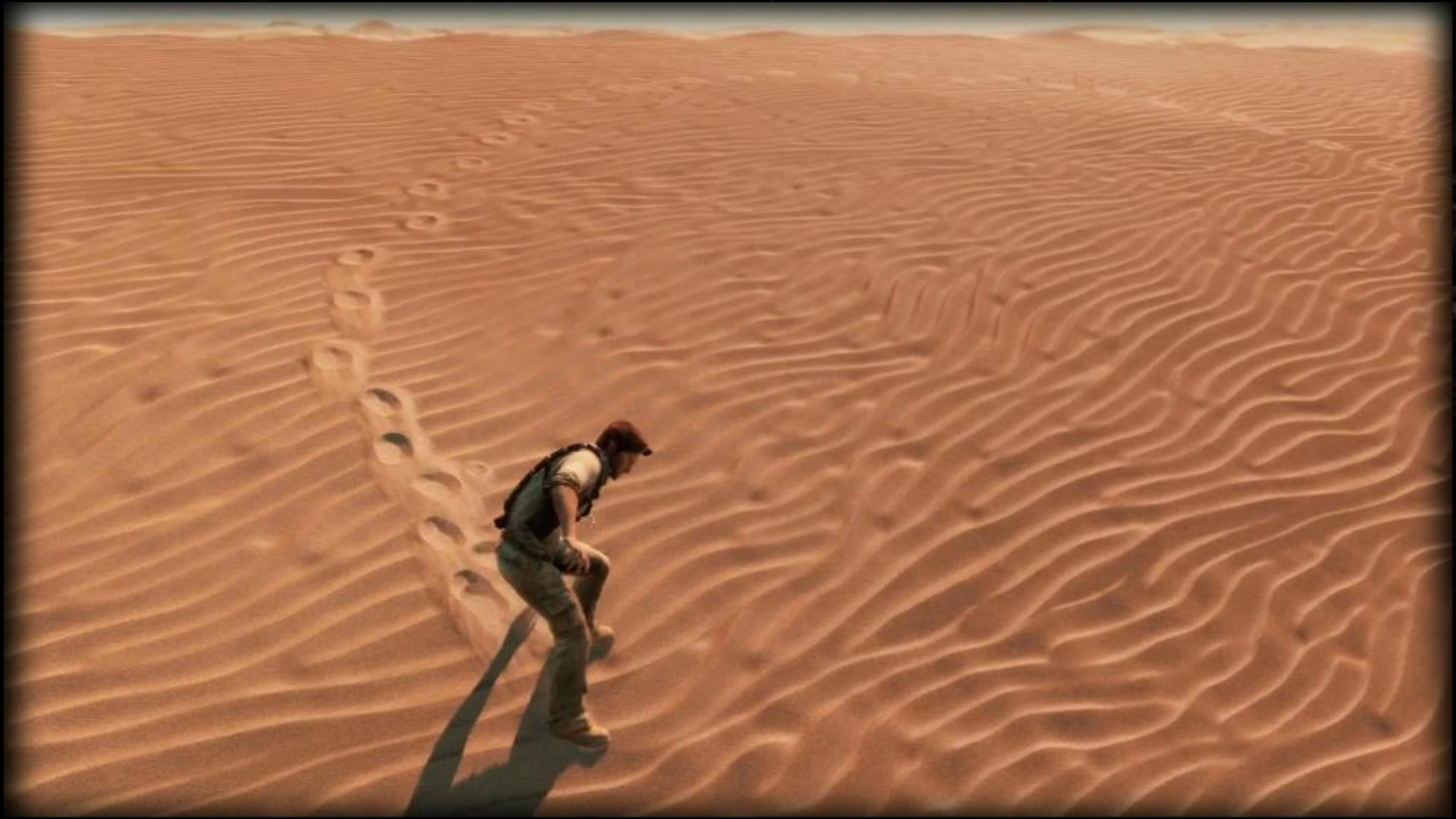
Interacting with the Desert

Sand Foot Prints

Interacting with the Desert

Sand Foot Prints

Interacting with the Desert



Interacting with the Desert

- Pros:
  - @#$%ing awesome
- Cons:
  - @#$%ing expensive

Retrospective

Retrospective

- Pros:
  - Incredible power and variety of controls and tools to experiment with for new solutions
  - Open communication, experimentation, and teamwork are our absolute greatest assets

Retrospective

- Cons
  - "With great power comes great responsibility"
    - We break the game all the time…locally.
  - Slow workflow from the vast amount of control

Naughty Dog is Hiring!

Company Email:

JOBS@NAUGHTYDOG.COM

Recruiter Email:

CANDACE_WALKER@NAUGHTYDOG.COM

Twitter:

@Candace_Walker

The Tricks Up Our Sleeves

A Walkthrough of the Special FX of Uncharted 3: Drake's Deception

Keith Guerrette
Lead Visual Effects Artist, Naughty Dog
www.keithguerrette.com

GAME DEVELOPERS CONFERENCE 2012
SAN FRANCISCO, CA
MARCH 5-9, 2012
EXPO DATES: MARCH 7-9