

---

# Introduction to Compute Shaders

Francis Joseph Serina

---

## About me

Former high school computer science teacher

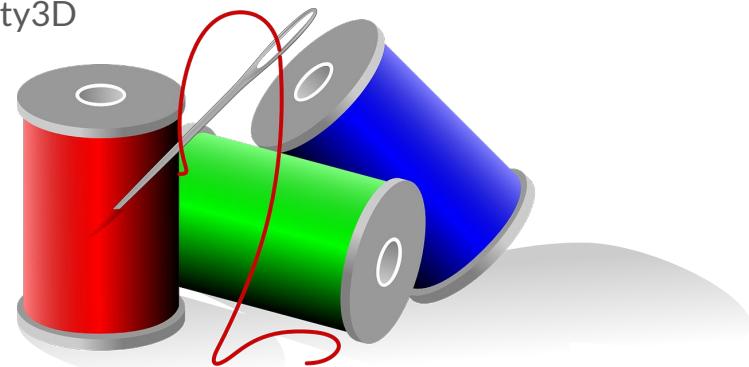
Unity3D Developer



---

# Goals

- Difference between CPU and GPU Threads
- Graphics Pipeline and Shaders
- When to use Compute Shaders
- Overview on how to use Compute Shaders in Unity3D



Imagine this...



Imagine this...



Imagine this...



Imagine this...



Programmers call this **Parallel Processing**



# Parallel Processing

Concurrency is when 2 or more activities are happening simultaneously

---

## Parallel Processing - Purpose

- Separation of Concerns





## Parallel Processing - Purpose

- Separation of Concerns
- Performance



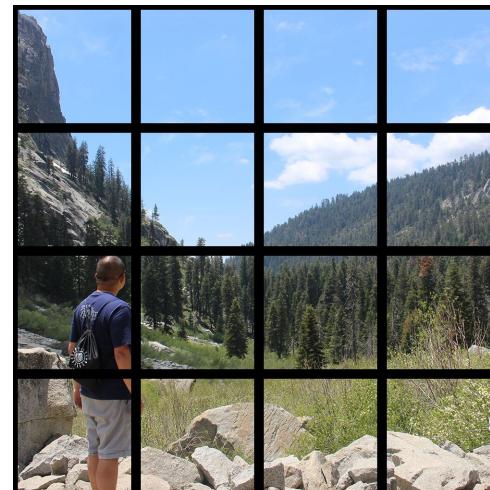
## Parallel Processing - Purpose

- Separation of Concerns
- Performance
  - Task Parallelism

---

# Parallel Processing - Purpose

- Separation of Concerns
- Performance
  - Task Parallelism
  - Data Parallelism



---

# Parallel Processing - Purpose

- Separation of Concerns
- Performance *<- this is what Shaders are meant for*
  - Task Parallelism
  - Data Parallelism



# What are Shaders?

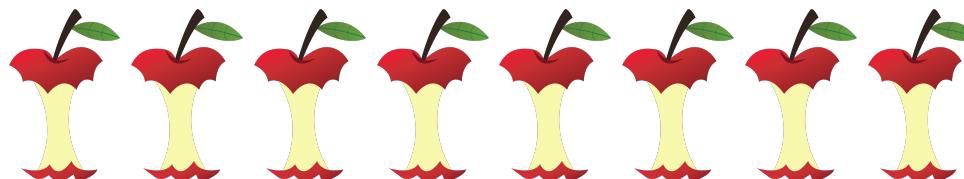
Programs running on the GPU instead of the CPU



---

# What are Shaders?

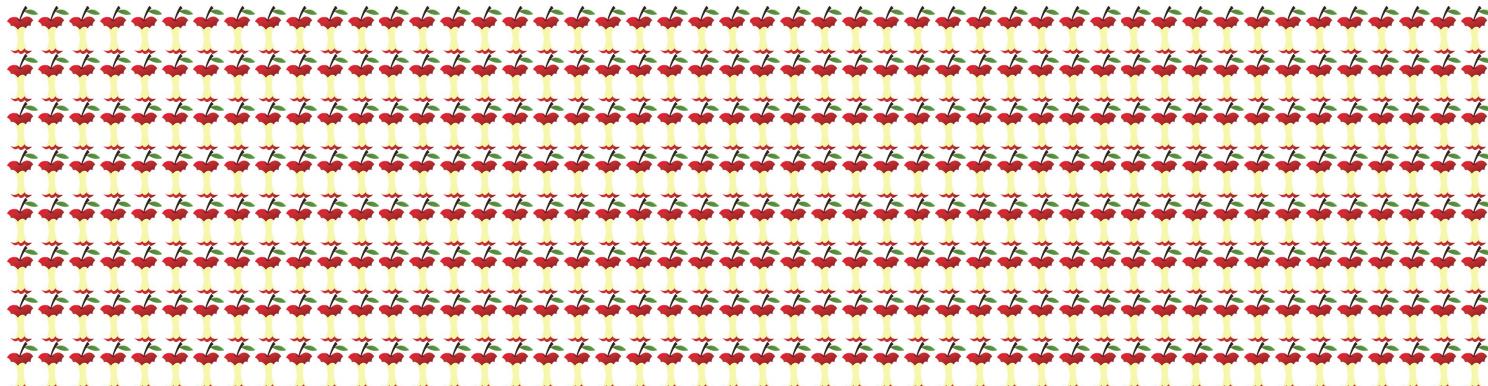
Programs running on the GPU instead of the CPU



---

# What are Shaders?

Programs running on the GPU instead of the CPU





# What are Shaders?

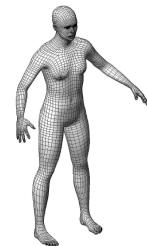
Originally intended for Graphics

---

# What are Shaders?

Originally intended for Graphics

Application





# What are Shaders?

Originally intended for Graphics



Application

Geometry



# What are Shaders?

Originally intended for Graphics



Application

Geometry

Rasterization



# What are Shaders?

Originally intended for Graphics



Application

Geometry

Rasterization

Screen



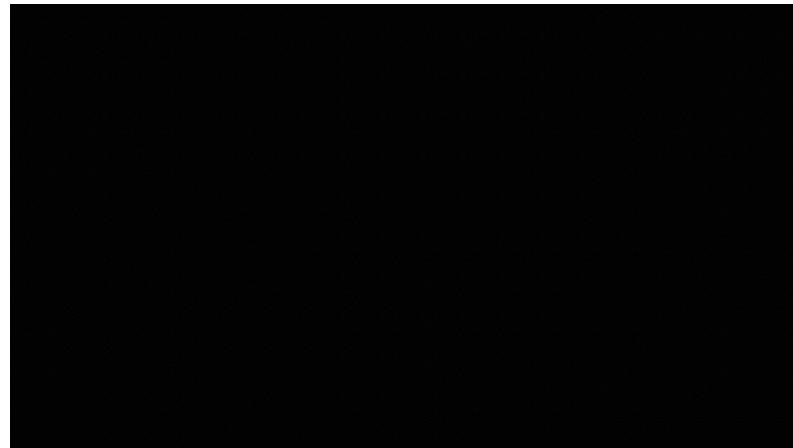
# What are Compute Shaders?

Programs run on the GPU but not necessarily for Graphics

---

# Sample Applications

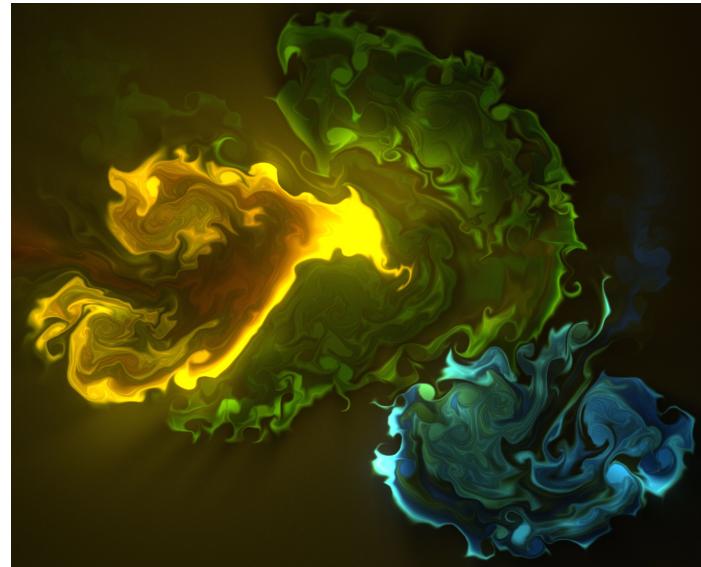
[WebGL Fluid Simulation\(github.io\)](#) by Pavel Dobryakov



---

## Sample Applications

[WebGL Fluid Simulation\(github.io\)](#) by Pavel Dobryakov



---

# Sample Applications

[Conway's Game of Life](#)





# Sample Applications

## Computer Vision

- Convolutions
- Fast Fourier Transform
- Feature Detection
- Post-processing Image Effects

## Simulations

- Weathering and Erosion (Terrain Generation)
- Particle Systems
- AI Pathfinding
- Diffusion and Propagation

---

# Sample Applications

Embarrassingly Parallel or Pleasantly Parallel or Conveniently Concurrent

- Little to no effort to divide the problem into parallel tasks
- Highly scalable algorithms

---

# Quick Guide: Compute Shaders in Unity3D

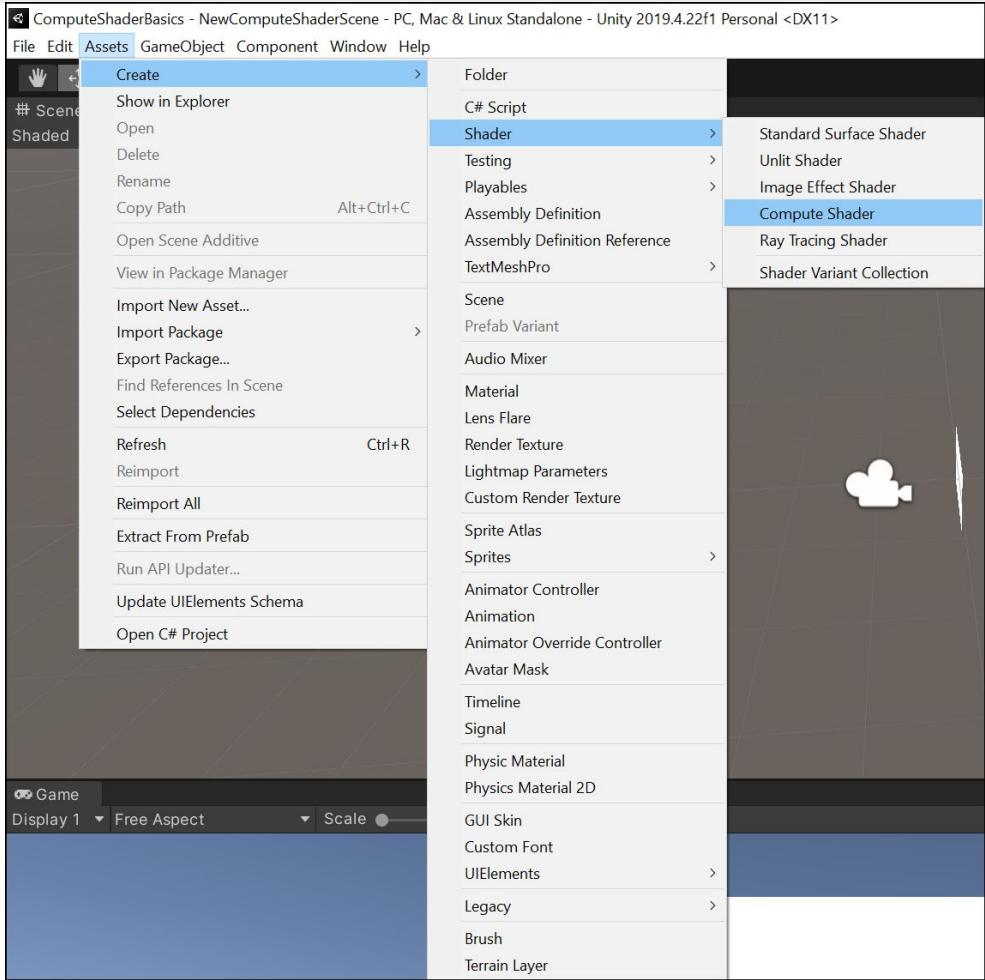


# The Plan

- Create a compute shader
- Create a MonoBehaviour to use the compute shader
- Set up the scene to show the output

# Creating a new Compute Shader

- Assets Menu
- Create
- Shader
- Compute Shader



## NewComputeShader.compute

```
1 //·Each·#kernel·tells·which·function·to·compile;·you·can·have·many·kernels
2 #pragma·kernel·CSMain
3
4 //·Create·a·RenderTarget·with·enableRandomWrite·flag·and·set·it
5 //·with·cs.SetTexture
6 RWTexture2D<float4>·Result;
7
8 [numthreads(8,8,1)]
9 void·CSMain·(uint3·id··:·SV_DispatchThreadID)
10 {
11     ...//·TODO:·insert·actual·code·here!
12
13     ...Result[id.xy]·=·float4(id.x·&·id.y,·(id.x·&·15)/15.0,·(id.y·&·15)/15.0,·0.0);
14 }
```

## NewComputeShader.compute

```
1 //·Each·#kernel·tells·which·function·to·compile;·you·can·have·many·kernels
2 #pragma·kernel·CSMain
3
4 //·Create·a·RenderTarget·with·enableRandomWrite·flag·and·set·it
5 //·with·cs.SetTexture
6 RWTexture2D<float4>·Result;
7
8 [numthreads(8,8,1)]
9 void·CSMain·(uint3·id··:SV_DispatchThreadID)
10 {
11     ...//·TODO:·insert·actual·code·here!
12
13     ...Result[id.xy]·=·float4(id.x·&·id.y,·(id.x·&·15)/15.0,·(id.y·&·15)/15.0,·0.0);
14 }
```

## NewComputeShader.compute

```
1 //·Each·#kernel·tells·which·function·to·compile;·you·can·have·many·kernels
2 #pragma·kernel·CSMain
3
4 //·Create·a·RenderTexture·with·enableRandomWrite·flag·and·set·it
5 //·with·cs.SetTexture
6 RWTexture2D<float4>·Result;
7
8 [numthreads(8,8,1)]
9 void·CSMain·(uint3·id··:·SV_DispatchThreadID)
10 {
11     ....//·TODO:·insert·actual·code·here!
12
13     ....Result[id.xy]·=·float4(id.x&·id.y,·(id.x&·15)/15.0,·(id.y&·15)/15.0,·0.0);
14 }
```



## NewComputeShader.compute

```
1 //·Each·#kernel·tells·which·function·to·compile;·you·can·have·many·kernels
2 #pragma·kernel·CSMain
3
4 //·Create·a·RenderTarget·with·enableRandomWrite·flag·and·set·it
5 //·with·cs.SetTexture
6 RWTexture2D<float4>·Result;
7
8 [numthreads(8,8,1)]
9 void·CSMain·(uint3·id·:·SV_DispatchThreadID)
10 {
11     //·TODO:·insert·actual·code·here!
12
13     Result[id.xy]·=·float4(id.x&id.y,·(id.x&15)/15.0,·(id.y&15)/15.0,·0.0);
14 }
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class NewComputeShaderController : MonoBehaviour
6  {
7      [SerializeField]
8      private ComputeShader _shader = null;
9
10     [SerializeField]
11     private Material _material = null;
12
13     private RenderTexture _texture = null;
14
15     private void Start()
16     {
17         _texture = new RenderTexture(32, 32, 24);
18         _texture.filterMode = FilterMode.Point;
19         _texture.enableRandomWrite = true; // IMPORTANT!
20         _texture.Create();
21
22         _material.mainTexture = _texture;
23
24         var kernel = _shader.FindKernel("CSMain");
25         _shader.SetTexture(kernel, "Result", _texture);
26         _shader.Dispatch(kernel, 4, 4, 1);
27     }
28
29     private void OnDestroy()
30     {
31         _texture.Release();
32         _texture = null;
33     }
34 }
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class NewComputeShaderController : MonoBehaviour
6  {
7      [SerializeField]
8      private ComputeShader _shader = null;
9
10     [SerializeField]
11     private Material _material = null;
12
13     private RenderTexture _texture = null;
14
15     private void Start()
16     {
17         _texture = new RenderTexture(32, 32, 24);
18         _texture.filterMode = FilterMode.Point;
19         _texture.enableRandomWrite = true; // IMPORTANT!
20         _texture.Create();
21
22         _material.mainTexture = _texture;
23
24         var kernel = _shader.FindKernel("CSMain");
25         _shader.SetTexture(kernel, "Result", _texture);
26         _shader.Dispatch(kernel, 4, 4, 1);
27     }
28
29     private void OnDestroy()
30     {
31         _texture.Release();
32         _texture = null;
33     }
34 }
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class NewComputeShaderController : MonoBehaviour
6  {
7      [SerializeField]
8      private ComputeShader _shader = null;
9
10     [SerializeField]
11     private Material _material = null;
12
13     private RenderTexture _texture = null;
14
15     private void Start()
16     {
17         _texture = new RenderTexture(32, 32, 24);
18         _texture.filterMode = FilterMode.Point;
19         _texture.enableRandomWrite = true; // IMPORTANT!
20         _texture.Create();
21
22         _material.mainTexture = _texture;
23
24         var kernel = _shader.FindKernel("CSMain");
25         _shader.SetTexture(kernel, "Result", _texture);
26         _shader.Dispatch(kernel, 4, 4, 1);
27     }
28
29     private void OnDestroy()
30     {
31         _texture.Release();
32         _texture = null;
33     }
34 }
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class NewComputeShaderController : MonoBehaviour
6  {
7      [SerializeField]
8      private ComputeShader _shader = null;
9
10     [SerializeField]
11     private Material _material = null;
12
13     private RenderTexture _texture = null;
14
15     private void Start()
16     {
17         _texture = new RenderTexture(32, 32, 24);
18         _texture.filterMode = FilterMode.Point;
19         _texture.enableRandomWrite = true; // IMPORTANT!
20         _texture.Create();
21
22         _material.mainTexture = _texture;
23
24         var kernel = _shader.FindKernel("CSMain");
25         _shader.SetTexture(kernel, "Result", _texture);
26         _shader.Dispatch(kernel, 4, 4, 1);
27     }
28
29     private void OnDestroy()
30     {
31         _texture.Release();
32         _texture = null;
33     }
34 }
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class NewComputeShaderController : MonoBehaviour
6  {
7      [SerializeField]
8      private ComputeShader _shader = null;
9
10     [SerializeField]
11     private Material _material = null;
12
13     private RenderTexture _texture = null;
14
15     private void Start()
16     {
17         _texture = new RenderTexture(32, 32, 24);
18         _texture.filterMode = FilterMode.Point;
19         _texture.enableRandomWrite = true; // IMPORTANT!
20         _texture.Create();
21
22         _material.mainTexture = _texture;
23
24         var kernel = _shader.FindKernel("CSMain");
25         _shader.SetTexture(kernel, "Result", _texture);
26         _shader.Dispatch(kernel, 4, 4, 1);
27     }
28
29     private void OnDestroy()
30     {
31         _texture.Release();
32         _texture = null;
33     }
34 }
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class NewComputeShaderController : MonoBehaviour
6  {
7      [SerializeField]
8      private ComputeShader _shader = null;
9
10     [SerializeField]
11     private Material _material = null;
12
13     private RenderTexture _texture = null;
14
15     private void Start()
16     {
17         _texture = new RenderTexture(32, 32, 24);
18         _texture.filterMode = FilterMode.Point;
19         _texture.enableRandomWrite = true; // IMPORTANT!
20         _texture.Create();
21
22         _material.mainTexture = _texture;
23
24         var kernel = _shader.FindKernel("CSMain");
25         _shader.SetTexture(kernel, "Result", _texture);
26         _shader.Dispatch(kernel, 4, 4, 1);
27     }
28
29     private void OnDestroy()
30     {
31         _texture.Release();
32         _texture = null;
33     }
34 }
```



## NewComputeShaderController.cs

```
var kernel = _shader.FindKernel("CSMain");
```

- Remember the `#pragma kernel` before?



# NewComputeShaderController.cs

```
var kernel = _shader.FindKernel("CSMain");
```

- Remember the `#pragma kernel` before?

```
_shader.SetTexture(kernel, "Result", _texture);
```

- Set the texture to the buffer



# NewComputeShaderController.cs

```
var kernel = _shader.FindKernel("CSMain");
```

- Remember the `#pragma kernel` before?

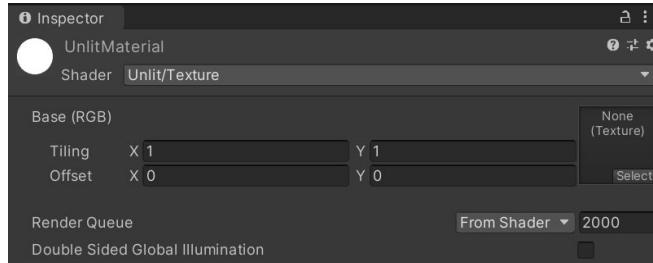
```
_shader.SetTexture(kernel, "Result", _texture);
```

- Set the texture to the buffer

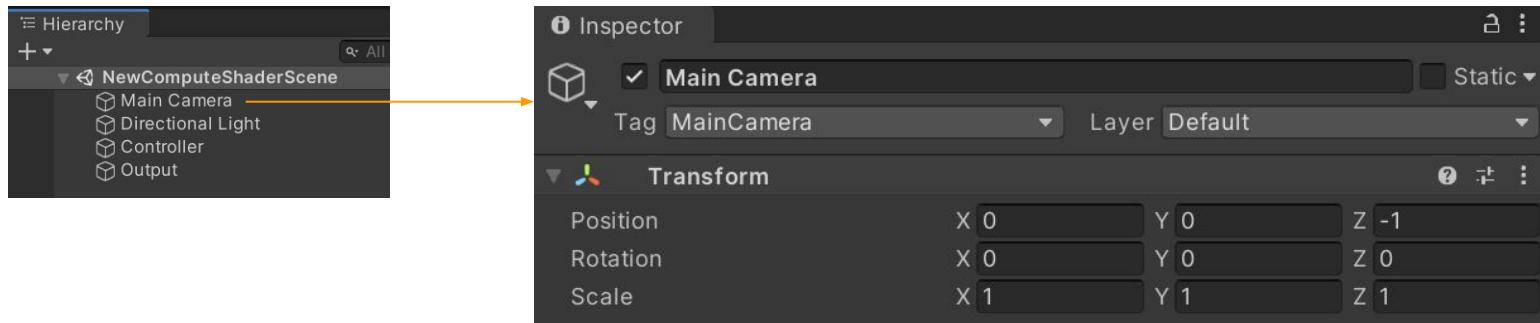
```
_shader.Dispatch(kernel, 4, 4, 1);
```

- Execute the kernel with the number of **thread groups**
- Each **thread group has 64 threads** from [numthreads (8,8,1) ]

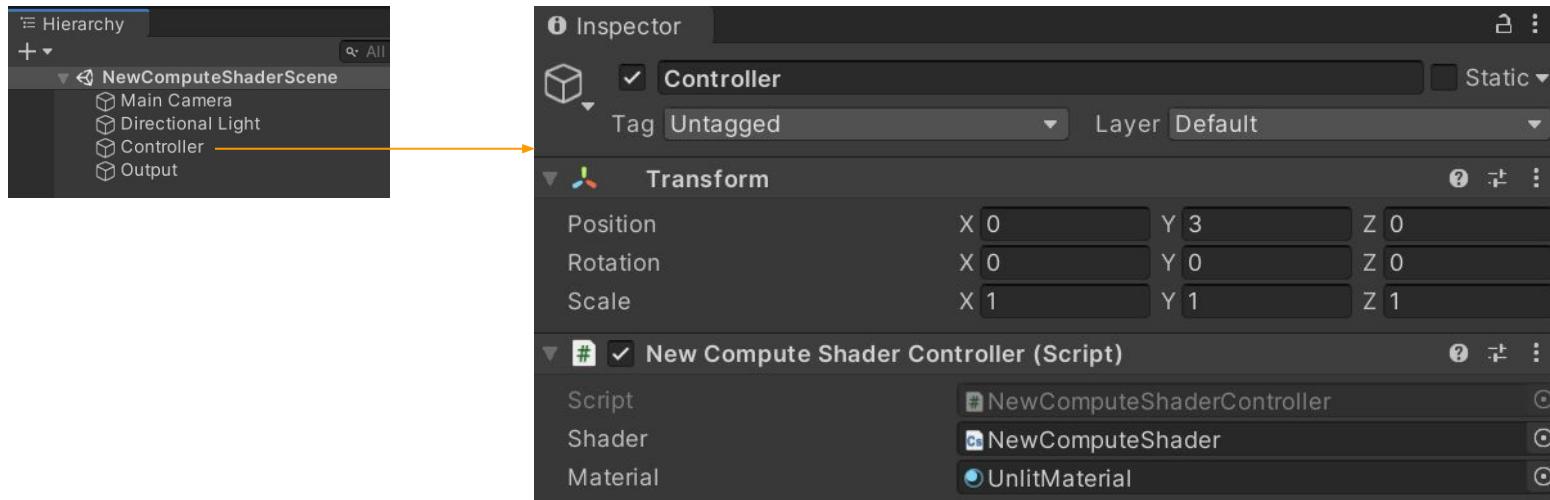
# Scene Setup



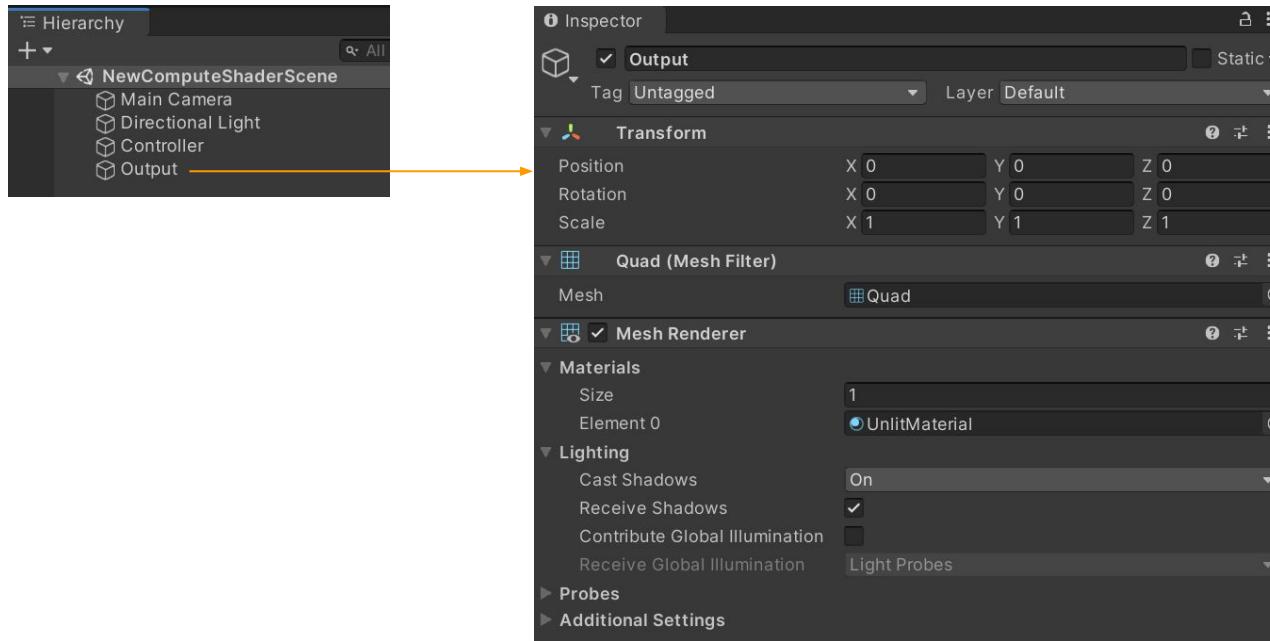
# Scene Setup



# Scene Setup

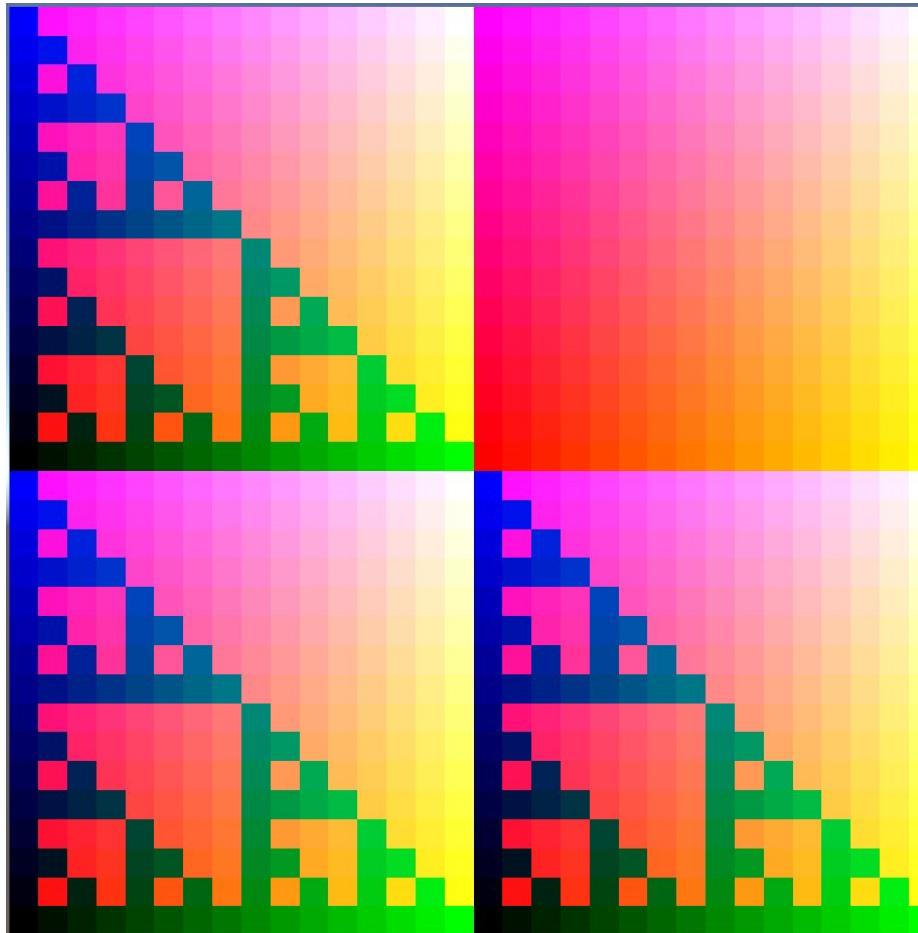


# Scene Setup



---

**Output**





# Source Code

- Source Code is at <https://github.com/xeratol/ComputeShaderBasics>
  - 01-NewComputeShader
    - How to use default compute shader boilerplate code
  - 02-Color
    - Set the color of a texture
  - 03-GrayScale
    - post-processing effect using Compute Shaders
    - [Cg Programming/Unity/Computing Image Effects](#)
  - 04-ThreadIDDemo
    - Thread ID and other parameters
  - 05-Life
    - Conway's Game of Life

# Q&A

<https://github.com/xeratol/ComputeShaderBasics>



---

# Introduction to Compute Shaders

Francis Joseph Serina

---

# Image Licenses

- [Meeting Business Brainstorming - Free vector graphic on Pixabay](#) Pixabay License
- [Thread Rolls Needle - Free vector graphic on Pixabay](#) Pixabay License
- [Child Joke Childhood](#) Pixabay License
- [Steps Running Kids - Free photo on Pixabay](#) Pixabay License
- [Free Images : play, color, paint, drawing, mural, finger painting, child art, kid painting, kids art 2352x1568 -- 958981 - Free stock photos](#) CC0 Public Domain
- [Camera Logo Symbol](#) Pixabay License
- [3d Model Free Stock Photo](#) CC0 Public Domain
- [Clip Art Of A Bitten Apple Bite Clipart Clipartxtras - Snow White Apple Clip Art PNG Image | Transparent PNG Free Download on SeekPNG](#) Personal Use
- [File:The OpenGL - DirectX graphics pipeline.png](#) CC BY-SA 3.0
- [Conway's Game of Life - Wikipedia](#) CC BY-SA 3.0
- [Man Jogging Cartoon.svg](#) from [Wikimedia Commons](#) by [Videoplasty.com](#), CC-BY-SA 4.0



# References

[CPU vs GPU? What's the Difference? Which Is Better?](#)