- Vinay D - 23371UA009
- CSGD IV Sem 2nd Year
- Unity Game Engine

Unity Thesis - Final Report

Thesis Topic

A Unity project demonstrating realistic fire VFX using VFX Graph

Table of Contents

1. Introduction

1.1 Background

Visual effects like fire and smoke play a vital role in making games and simulations more immersive. Static environments often lack this realism, which reduces player engagement.

1.2 Problem Statement

Many beginner Unity projects lack dynamic fire effects, relying on simple textures or animations. This reduces realism and fails to capture natural fire behavior.

1.3 Objectives

- Create a fire and smoke effect using Unity’s VFX Graph.
- Simulate realistic particle behavior with color gradients, noise, and lighting.
- Optimize the effect for smooth real-time performance.

1.4 Scope of the Project

The project focuses on designing fire VFX within Unity. It does not cover advanced fluid simulation but aims to create visually convincing fire and smoke suitable for real-time games.

2. Literature Review

2.1 Visual Effects in Games

Modern games rely heavily on visual effects like fire, smoke, and explosions to improve immersion and realism.

2.2 Unity VFX Graph

Unity's VFX Graph enables GPU-driven particle effects, allowing complex visuals such as fire and smoke with high performance.

2.3 Particle Systems

Particle systems simulate natural phenomena by controlling properties like lifetime, velocity, color, and noise. They are essential for fire and smoke representation.

2.4 Related Work

Prior studies and game projects show fire VFX is often achieved by combining glowing particles with smoke trails, optimized for real-time rendering.

---

3. System Design

3.1 Project Workflow

The system is designed in stages:

1. Researching Unity's VFX Graph and particle systems.
2. Setting up a Unity project with necessary VFX Graph packages.
3. Creating a fire effect using GPU-based particles.
4. Adding smoke trails that fade and disperse naturally.

5. Optimizing particle counts for real-time use.

3.2 Tools and Technologies Used

- **Unity 2021.3+** for project development.
- **VFX Graph** for GPU-driven particle effects.
- **Shader Graph** (optional) for additional glow and distortion.
- **Lighting System** in Unity for realistic illumination.

3.3 VFX Graph Node Design

The fire and smoke effects are created using a node-based workflow:

- **Spawner Node** – controls emission rate.
- **Initialize Particle** – sets lifetime, velocity, and size.
- **Update Particle** – applies forces like turbulence and gravity.
- **Output Particle (Quad)** – renders fire with gradient textures.
- **Noise Module** – gives smoke random dispersion.

3.4 Integration with Unity Scene

The fire VFX is placed inside a sample Unity scene to test behavior in different environments. The system responds to lighting changes and blends with the environment. Multiple fire variations are created to demonstrate scalability.

4. Implementation

4.1 Unity Project Setup

The project was created in Unity 2021.3 with the **Visual Effect Graph (VFX Graph)** package installed. A new VFX asset was created and attached to a GameObject in the scene. This served as the base for both fire and smoke effects.

4.2 Fire Particle System Creation

- A **Spawner Node** was used to control emission rate.
- The **Initialize Particle** node set particle lifetime, size, and velocity.
- **Color over Lifetime** was applied using a gradient (yellow → orange → red → black).
- Fire glow was simulated with **additive blending**.

```
if (fireEffect.aliveParticleCount > 1000) {
        fireEffect.Stop(); // Prevent overload
    }
```

4.3 Smoke Trail Design

- A secondary particle system was added for smoke.
- **Noise & Turbulence** modules created random dispersion.
- The smoke particles faded from dark gray to transparent.
- Longer lifetime values gave a natural trailing effect.

4.4 Color Gradients and Lighting Integration

- Fire particles were given a **hot-to-cool gradient**.
- Smoke used **alpha fading** for realism.
- A **Point Light** was attached near the fire to simulate flickering glow.

4.5 Optimizing Performance

- Limited maximum particle count to ensure smooth FPS.
- Enabled **GPU simulation** for efficient rendering.
- Used **Level of Detail (LOD)** to reduce particle density at distance.

```
if (fireEffect.aliveParticleCount > 1000) {
    fireEffect.Stop(); // Prevent overload
}
```

---

5. Results and Analysis

5.1 Fire VFX Variations (Color & Intensity)

Different fire effects were created by adjusting emission rate, particle size, and color gradients.

- **High intensity fire**: brighter, faster emission, short particle lifetime.
- **Low intensity fire**: slower emission, longer lifetime, dimmer glow.

5.2 Smoke Behavior and Realism

The smoke particles dispersed naturally with turbulence applied.

- **Denser smoke for larger fires.**
- **Faded smoke for smaller flames.**
 The alpha fading created a realistic trail effect as smoke disappeared over time.

5.4 Visual Comparisons

- **Without VFX Graph: fire appeared flat and less immersive.**
- **With VFX Graph: fire and smoke had depth, glow, and natural movement.**

- **Adding lighting effects further improved scene realism.**

---

6. Discussion

6.1 Achievements of the Project

- Successfully created a realistic **fire and smoke effect** using Unity's VFX Graph.
- Implemented **color gradients, turbulence, and fading** for natural visuals.
- Achieved **real-time performance** with GPU particle simulation.
- Demonstrated **variations in fire intensity** and smoke density.

6.2 Challenges Faced

- Balancing **realism vs performance** was difficult when using large particle counts.
- Achieving natural smoke dispersion required **fine-tuning noise and lifetime settings**.
- Lighting integration was tricky, as flickering lights sometimes caused **performance drops**.

6.3 Solutions Applied

- Optimized particle counts and used **GPU-based rendering** to improve FPS.
- Tweaked turbulence and lifetime values until smoke trails felt natural.
- Attached a **point light with controlled intensity** to simulate fire glow without overload.

---

7. Future Work

7.1 Enhancements in Realism

Future versions can add **heat distortion effects** using shaders to simulate air refraction near fire.

- Dynamic flickering glow on nearby objects.
- More advanced smoke simulation with volumetric rendering.

7.2 Integration into Larger Game Environments

- Fire VFX can be combined with **destructible objects** (wood, walls).
- Can be used as part of **gameplay mechanics** (damage zones, torches, explosions).
- Scalable fire variations for **different levels or weather conditions.**

7.3 Performance Scaling for Complex Scenes

- Implement **Level of Detail (LOD)** to reduce particle count at distance.
- Use **object pooling** for reusable fire effects in large maps.
- Explore **GPU instancing** for handling multiple fire sources efficiently.

---

8. Conclusion

8.1 Summary of Work

This project demonstrated the creation of a **realistic fire and smoke system** in Unity using the VFX Graph. Particle systems, gradients, turbulence, and lighting were combined to simulate natural fire behavior. The implementation showed how such effects can improve immersion in real-time applications.

8.2 Key Learnings

- Gained hands-on experience with **Unity's VFX Graph** and GPU-driven particles.
- Learned the importance of **balancing realism and performance.**
- Discovered how **lighting and color blending** affect the perception of fire.

9. Appendix

Image 1

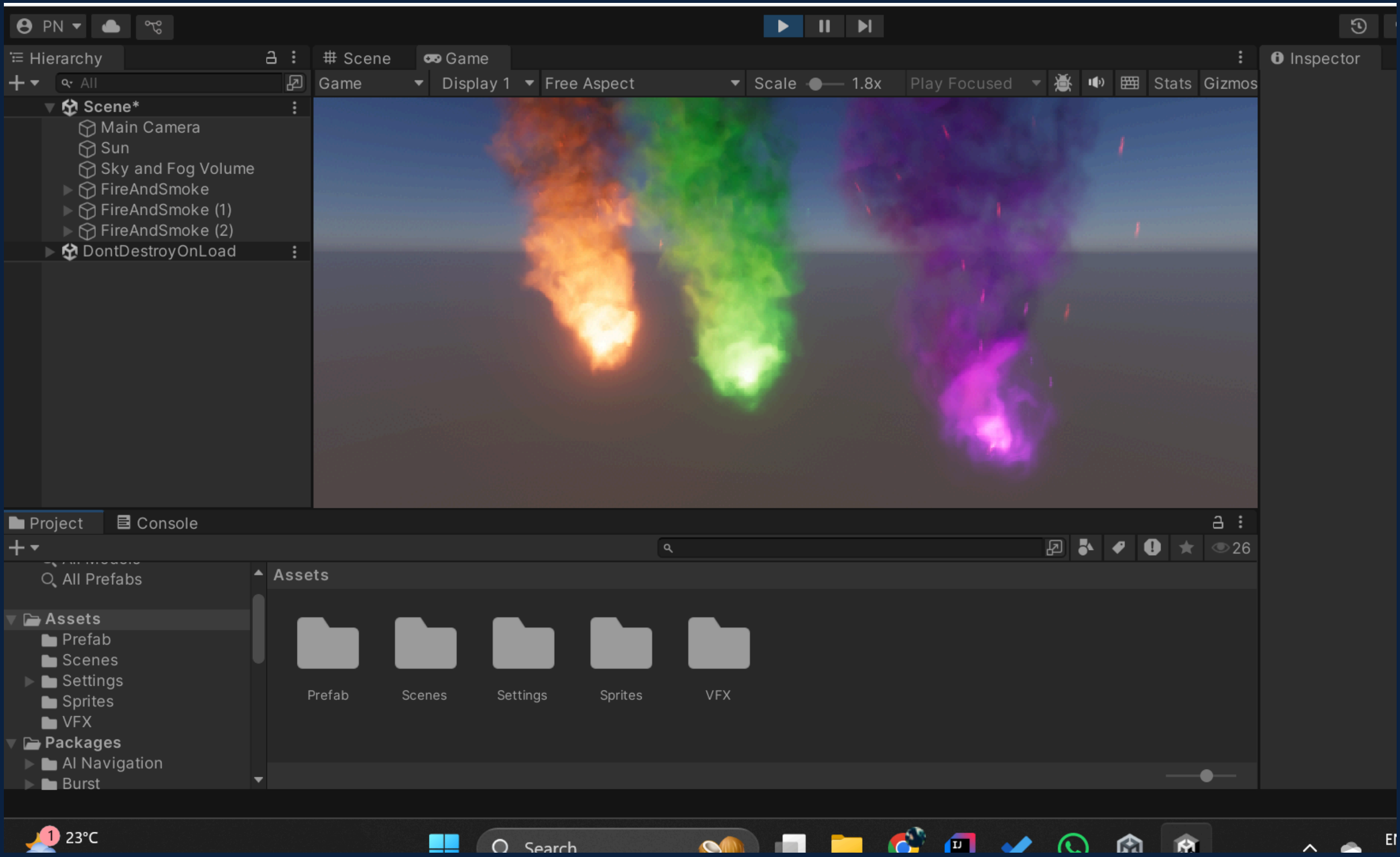


Image 2

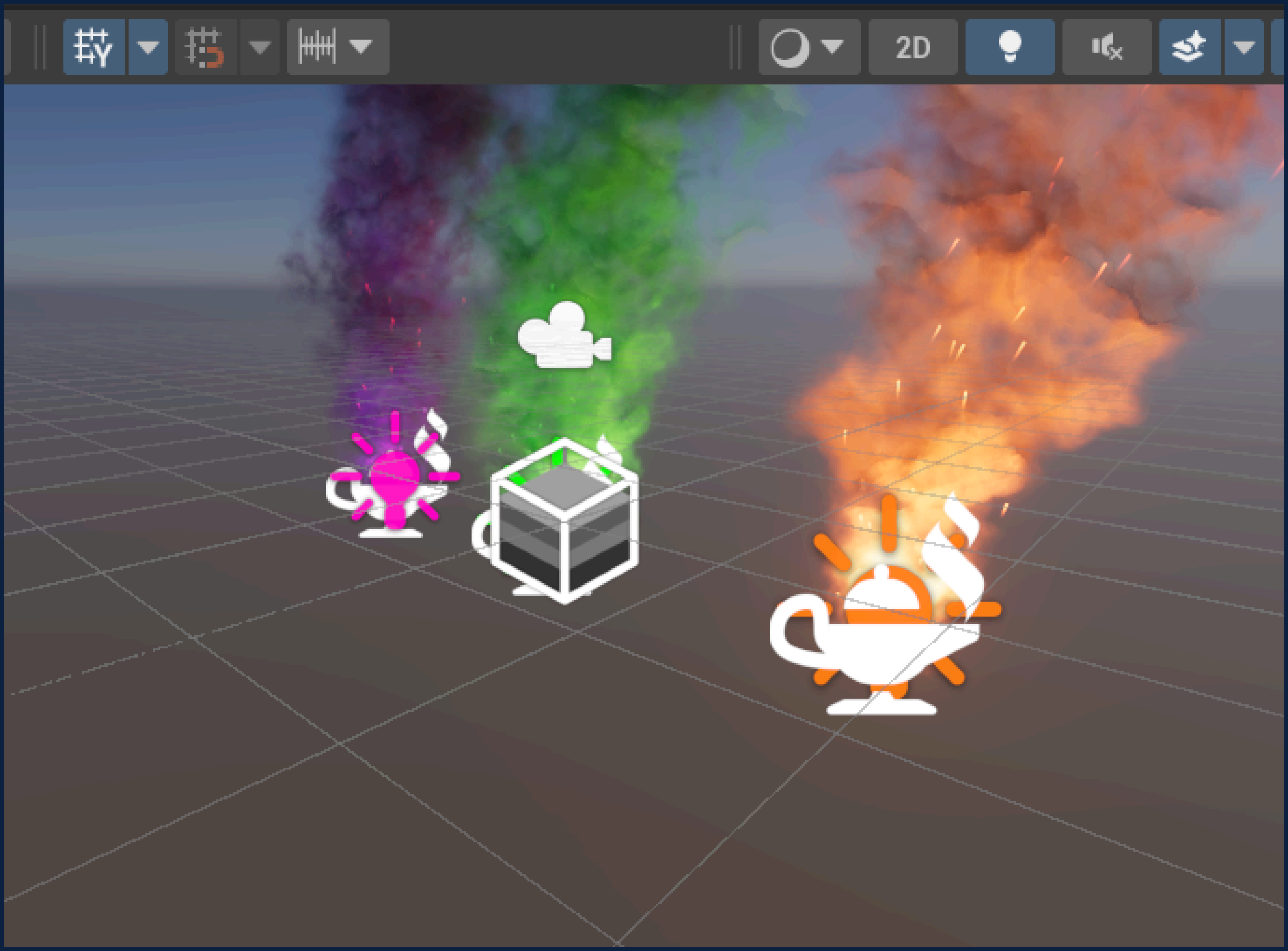
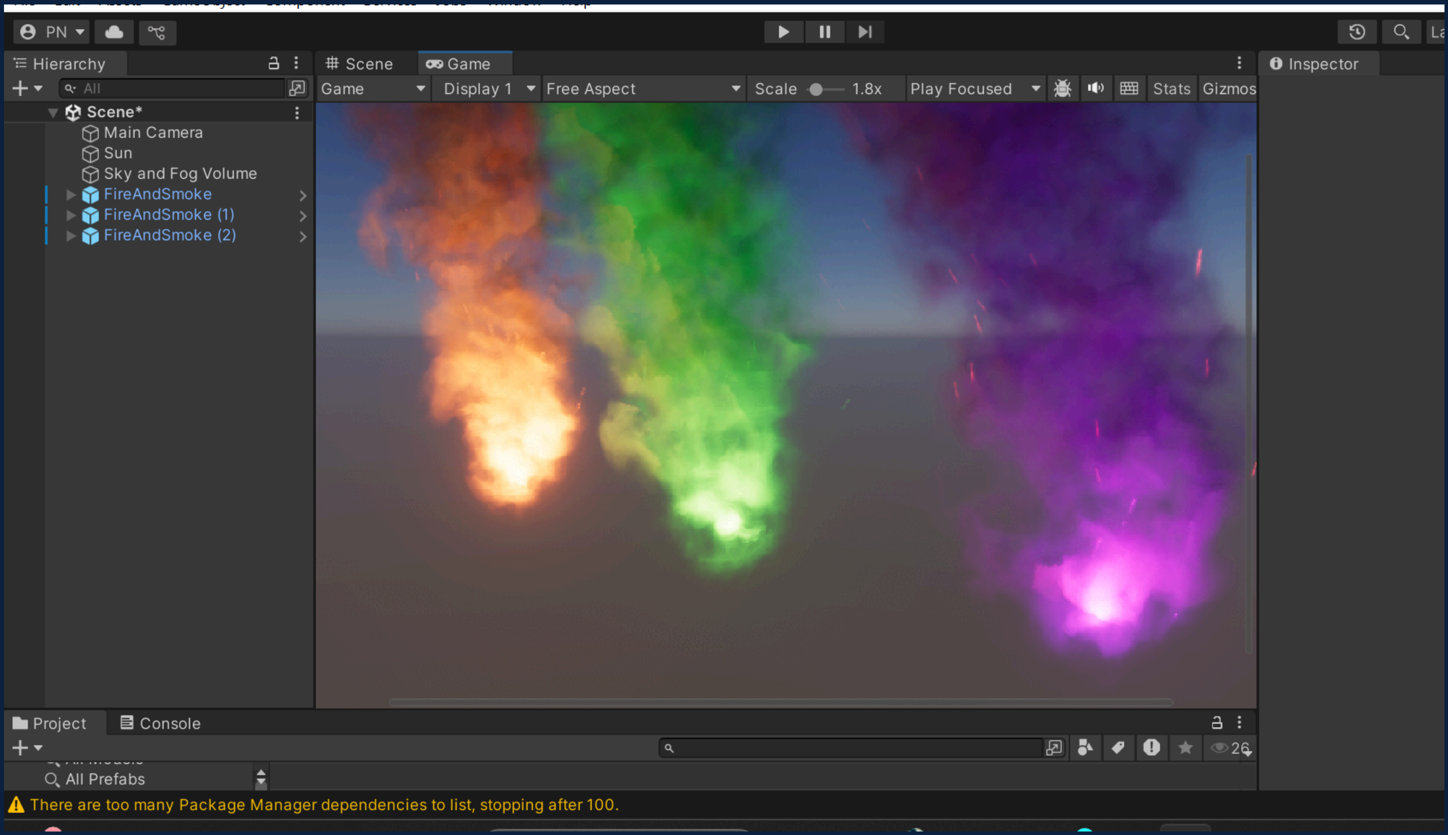


Image 3



There are too many Package Manager dependencies to list, stopping after 100.